О.І. Черняк, П.В. Захарченко

ІНТЕЛЕКТУАЛЬНИЙ АНАЛІЗ ДАНИХ

Підручник

Затверджено
Міністерством освіти і науки України

Київ
2010

ЗМІСТ

ПЕРЕДМОВА

Сьогодні ми є свідками активного розвитку технологій інтелектуального аналізу даних (Data Mining), поява яких пов'язана, в першу чергу, з необхідністю аналітичної обробки великих об'ємів інформації, що нагромаджуються в сучасних базах даних. Більшість компаній накопичують під час своєї діяльності величезні об'єми даних, але головне, що вони хочуть від них отримати - це корисну інформацію. Як можна дізнатися з даних про те, що є вигіднішим для клієнтів компанії, як розмістити ресурси ефективним чином або як мінімізувати втрати? Для вирішення цих проблем і призначені новітні технології інтелектуального аналізу, які використовуються для знаходження моделей і відносин, прихованих в середовищі даних, - моделей, які не можуть бути знайдені звичайними методами.

Суть і мету технологій Data Mining можна охарактеризувати так: це технології, які призначені для пошуку у великих об'ємах даних неочевидних, об'єктивних і корисних на практиці закономірностей. Сфера застосування Data Mining нічим не обмежена - вона скрізь, де є які-небудь дані. Але в першу чергу методи Data Mining сьогодні заінтригували компанії, що розгортають проекти на основі сучасних інформаційних технологій. Досвід багатьох таких компаній показує, що віддача від використання Data Mining може досягати 1000%.

Технології інтелектуального аналізу даних потрібні в першу чергу спеціалістам, що ухвалюють важливі рішення, - керівникам, аналітикам, експертам, консультантам. Дохід компанії більшою мірою визначається якістю цих рішень - точністю прогнозів, оптимальністю вибраних стратегій. І від якості цих рішень залежить розвиток компанії. Слід також зазначити, що для реальних задач бізнесу і виробництва не існує чітких алгоритмів рішення. Тому керівники і експерти вирішують такі задачі тільки на основі особистого досвіду. Часто класичні методики виявляються малоефективними для багатьох практичних завдань, оскільки неможливо точно описати реальність за

допомогою невеликого числа параметрів моделі, або розрахунок моделі займає дуже багато часу і обчислювальних ресурсів. Аналітичні технології дозволяють створювати моделі, що істотним чином підвищують ефективність рішень.

Запропонований підручник містить загальну інформацію про сутність, створення та практичне застосування технологій інтелектуального аналізу даних. Основні етапи розробки та застосування таких систем супроводжуються використанням різноманітного програмного забезпечення, яке досить широко і детально представлено в кожній главі підручника. Велика кількість ілюстративного матеріалу, структурованість розділів та підрозділів дають підстави сподіватися на швидке оволодіння запропонованими технологіями.

Структуру підручника складають 8 розділів. Автори зробили все можливе, щоб ознайомлення з матеріалом можна було починати з будь-якої частини, тому досвідчений спеціаліст може використовувати підручник як довідник з системи технологій інтелектуального аналізу даних. Початківцям радимо опановувати матеріал у запропонованому порядку, що дозволятиме послідовно аналізувати сутність та основні проблеми, які виникають при створенні та застосуванні різних технологій інтелектуального аналізу даних. Кожен розділ підручника містить тести з відповідної теми, виконання яких дозволить самостійно закріпити опанований теоретичний матеріал. Розділ 8 містить опис виконання лабораторних робіт, що дозволить оволодіти практикою застосування різноманітних технологій інтелектуального аналізу даних для широкого класу задач з реальної діяльності українських підприємств і організацій.

Перший розділ підручника присвячений сутності, принципам і особливостям аналітичних технологій, які суттєво відрізняють їх від інших інформаційних технологій. Еволюція таких технологій зумовила появу нового класу систем – систем інтелектуального аналізу даних (Data Mining). В матеріалі розділу розглядається їх сутність, основні моделі інтелектуальних обчислень, засоби комп’ютерної підтримки, сучасна практика застосування та перспективні напрями розвитку.

Застосування технологій інтелектуального аналізу даних, як правило, базується на обробці великих об'ємів інформації, що нагромаджуються в сучасних сховищах даних. Існують різні концепції, технології та практичні підходи до побудови таких сховищ. Також розроблена досить потужна технологія їх використання – OLAP – технологія, яка має різноманітні архітектури, особливості і широкі практичні можливості. Детальне вивчення цих технологій розглядається у другому розділі підручника.

В діяльності багатьох компаній досить часто доводиться вирішувати задачі, постановка яких неформальна, а рішення неоднозначне. Навіть якщо строгий алгоритмічний підхід неможливий, а отримати точне рішення принципово не можна, існують інші ефективні способи рішення. Важливе місце серед них займають нейрокомп’ютерні технології та нейронні мережі. В матеріалі третього розділу розглянуто сутність, архітектури, практика побудови та програмні засоби реалізації нейрокомп’ютерних технологій, а також нейронні мережі Хопфілда і Кохонена та сучасна практика і перспективні напрямки їх застосування.

У четвертому розділі представлен матеріал, присвячений пошуку асоціативних правил та побудови дерев рішень. Зокрема розглядаються питання теорії асоціативних правил, програмних засобів пошуку асоціативних правил та практичний аспект застосування цієї технології. В цьому розділі також розглянута технологія дерев рішень, яка є однією з найбільш популярних методів вирішення задач класифікації і прогнозування, а саме, подані загальні принципи технології, комп’ютерні системи та напрямки застосування дерев рішень.

Еволюційна теорія довела свою ефективність як при вирішенні складноформалізуємих задач кластеризації, асоціативного пошуку, так і при вирішенні трудомістких задач оптимізації, апроксимації, інтелектуальної обробки даних. Концепції еволюційних обчислень включає генетичні алгоритми, генетичне програмування, еволюційні стратегії і еволюційне програмування. Еволюційні технології інтелектуального аналізу сьогодні

успішно застосовуються для вирішення ряду великих і економічно значущих задач в бізнесі та інших важливих проектах. В п’ятому розділі подані матеріали про концептуальні засади еволюційної теорії, основні положення теорії генетичних алгоритмів та їх моделі, а також інформація про програмне забезпечення та сфері застосування генетичних алгоритмів. Мурашині алгоритми та генетичне програмування є самими сучасними напрямками еволюційної технології і майже не розглядалися раніше в підручниках по технологіям інтелектуального аналізу даних.

Важливим напрямком розвитку інтелектуального аналізу даних є широке застосування теорії нечітких обчислень, яка в сучасному світі розглядається як консорціум обчислювальних методологій, що колективно забезпечують основи для розуміння, конструювання і розвитку інтелектуальних систем, зокрема, систем інтелектуального аналіза даних. Тому в шостому розділі підручника розглянута концепція нечітких обчислень, а також її складові – теорія нечітких множин та нечітка логіка. Представлені матеріали по програмному забезпечення нечітких методів та сучасна практика застосування таких методів.

Сьомий розділ підручника присвячено класичним технологія інтелектуального аналізу, зокрема, технологіям класифікації та кластеризації, які є досить поширеним і потужним засобом аналізу даних. В представленому матеріалі розглянуті як традиційні, так і сучасні методи та алгоритми класифікації, а також сучасні програмні засоби їх підтримки, стан та перспективні напрямки застосування. Задачі кластеризації схожі із задачами класифікації і є їх логічним продовженням. Сучасною наукою розроблено велику кількість методів та алгоритмів цього виду аналізу, створені потужні програмні засоби їх реалізації та методології практичного впровадження, що також описано в даному розділі.

Розділ вісім має практичний напрям і націлений на придбання навичок використовувати теоретичний матеріал і сучасні комп'ютерні засоби для вирішення аналітичних задач з практики діяльності українських підприємств,

фірм і організацій. В цьому розділі запропонован перелік лабораторних робіт, які на думку авторів, дозволять отримати перший практичний досвід і поєднати знання з системи технологій інтелектуального аналізу з їх практичним втіленням в реальну дійсність.

Основним завданням підручника залишається надання студентам економічних спеціальностей цілісного уявлення про процес інтелектуального аналізу даних, зміст його етапів, технологію залучення інструментальних засобів тощо.

Для полегшення подання та сприйняття інформації у пропонованому виданні прийняті такі позначення. Найбільш важливі терміни, означення та назви методів, алгоритмів, програмних засобів тощо виділені напівжирним шрифтом. Всі пункти меню, назви піктограм меню, назви кнопок управління виділені курсивом та лапками.

У підручнику можна знайти теоретичне підґрунтя та практичні приклади застосування різноманітних технологій інтелектуального аналізу даних, що дає змогу оптимізувати час впровадження та кошторис витрат.

Підручник може бути корисним для студентів та аспірантів економічних спеціальностей вищих навчальних закладів. Крім того, це видання слід рекомендувати як довідник для всіх економістів-вчених та практиків, які використовують у своїй роботі технології інтелектуального аналізу даних.

Автори з великою вдячністю ознайомляться з Вашими коментарями та зауваженнями щодо підручника за адресами:

- chernyak@univ.kiev.ua
- pvzz1957@gmail.com

«Ми стільки можемо, скільки знаємо»
Френсис Бэкон

Розділ 1.

ОСНОВНІ ПОНЯТТЯ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

1.1. Сутність аналітичних технологій.

Аналітичні технології почали застосовуватися людством досить давно. Простим прикладом аналітичної технології є теорема Піфагора, яка дозволяє визначити довжину гіпотенузи маючи відомі довжини катетів по відомій формулі $c^2 = a^2 + b^2$. Іншим прикладом аналітичної технології можна назвати алгоритм обробки інформації людським мозком. Навіть мозок дитини може виконувати задачі, непідвладні сучасним комп'ютерам, наприклад, розпізнавання знайомих облич в юрбі або ефективне управління декількома десятками м'язів при грі у футбол. Унікальністю мозку є здібність до розв’язання нових задач - гри в шахи, водінню автомобіля і т.д. Але при цьому, мозок погано пристосований до обробки великих об'ємів числової інформації - людина не може перемножити два багатозначні числа, не використовуючи калькулятора або алгоритму обчислення в стовпчик. Реальні завдання з числами, набагато складніше, ніж множення і людині для вирішення таких завдань необхідні додаткові методики і інструменти.

Під *аналітичною технологією* будемо розуміти методики, які на основі певних моделей, алгоритмів, математичних теорем дозволяють за відомими даними оцінити значення невідомих характеристик і параметрів.

Аналітичні технології потрібні в першу чергу людям, що ухвалюють важливі рішення, - керівникам, аналітикам, експертам, консультантам. Дохід компанії більшою мірою визначається якістю цих рішень - точністю прогнозів, оптимальністю вибраних стратегій. І від якості цих рішень залежить розвиток компанії. За допомогою аналітичних технологій можна вирішувати проблеми прогнозування, наприклад, курсів валют, цін на сировині, попиту, доходу

компанії, рівня безробіття і оптимізації, наприклад, плану закупівель, плану інвестицій, стратегії розвитку. Слід також зазначити, що для реальних задач бізнесу і виробництва не існує чітких алгоритмів їх розв’язання. Тому керівники і експерти знаходять рішення таких задачі тільки на основі особистого досвіду. Часто класичні методики виявляються малоефективними для багатьох практичних завдань, оскільки неможливо точно описати реальність за допомогою невеликого числа параметрів моделі, або розрахунок моделі займає дуже багато часу і обчислювальних ресурсів. Аналітичні технології дозволяють створювати моделі, що істотним чином підвищують ефективність рішень [3, 24, 30].

Серед класичних підходів до аналізу даних на практиці найбільш поширеними виявилися детерміновані технології та імовірнісні технології.

Детерміновані аналітичні технології типа теореми Піфагора використовуються людиною вже багато сторіч. За цей час було створено величезну кількість формул, теорем і алгоритмів для вирішення класичних завдань - визначення об'ємів, знаходження рішень систем лінійних рівнянь, пошуку коренів багаточленів. Розроблені складні і ефективні методи аналізу задач оптимального управління, розв’язання диференціальних рівнянь і т.д. Всі ці методи діють по одній і тій же схемі (рис. 1.1.).

Рис.1.1. Схема детермінованої аналітичної технології.

Для застосування алгоритму необхідно, щоб дане завдання цілком описувалося певною детермінованою моделлю (деяким набором відомих функцій і параметрів). У такому разі алгоритм дає точна відповідь. Наприклад, для застосування теореми Піфагора потрібно перевірити, що трикутник - прямокутний.

На практиці часто зустрічаються завдання, пов'язані із спостереженням випадкових величин, наприклад, задача прогнозування курсу акцій. Для подібних проблем не можна будувати детерміновані моделі, тому застосовується принципово інший, імовірнісний підхід. Параметри імовірнісних моделей - це розподіли випадкових величин, їх середні значення, дисперсії і т.д. Як правило, ці параметри наперед невідомі, а для їх оцінки використовуються статистичні методи, вживані до вибірок зафіксованих значень (історичних даних) (рис. 1.2.).

Рис.1.2. Схема імовірнісної аналітичної технології.

Такого роду методи припускають, що відома деяка імовірнісна модель задачі. Наприклад, в задачі прогнозування курсу можна припустити, що завтрашній курс акцій залежить тільки від курсу за останні 2 дні (авторегресійна модель). Якщо це вірно, то спостереження курсу впродовж декількох місяців дозволяють досить точно оцінити коефіцієнти цієї залежності і прогнозувати курс в майбутньому (рис. 1.3.).

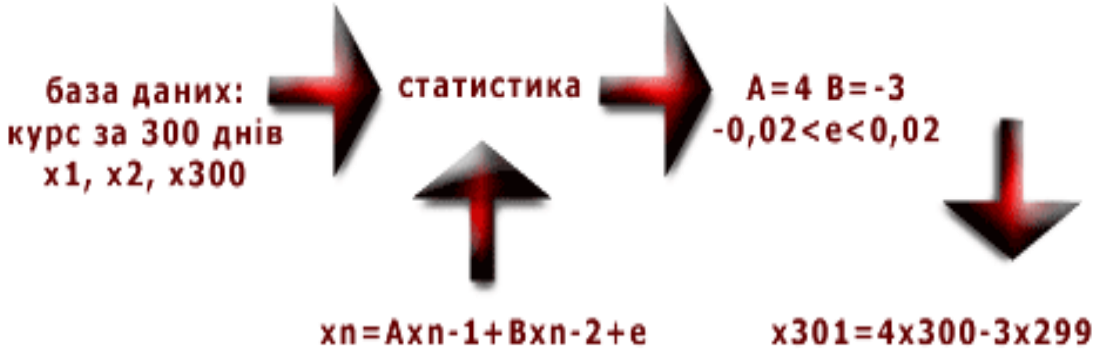


Рис.1.3. Схема прогнозування курсу акцій.

На жаль, класичні методики виявляються малоефективними в багатьох практичних задачах. Це пов'язано з тим, що неможливе достатньо повно описати реальність за допомогою невеликого числа параметрів моделі, або розрахунок моделі вимагає дуже багато час і обчислювальних ресурсів. Зокрема, розглянемо проблеми, що виникають при пошуку рішення задачі оптимального розподілу інвестицій [19]:

- у реальній задачі жодна з функцій не відома точно, а відомі лише приблизні або очікувані значення прибутку. Для того, щоб позбавитися від невизначеності, ми вимушені зафіксувати функції, втрачаючи при цьому в точності опису задачі;
- детермінований алгоритм для пошуку оптимального рішення може бути застосовний тільки в тому випадку, якщо всі дані функції лінійні. У реальних задачах бізнесу ця умова не виконується. Хоча дані функції можна апроксимувати лінійними, рішення в цьому випадку буде далеким від оптимального;
- якщо одна з функцій нелінійна, то симплекс-метод непридатний, і залишається два традиційні шляхи пошуку рішення цієї задачі. Перший шлях - використовувати метод градієнтного спуску для пошуку максимуму прибутку. В випадку, коли область визначення функції прибутку має складну форму, а сама функція - декілька локальних максимумів, градієнтний метод може привести до неоптимального

рішення. Другий шлях - провести повний перебір варіантів інвестування. Якщо кожна з 10 функцій задана в 100 крапках, то доведеться перевірити близько 1020 варіантів, що зажадає не менші декілька місяців роботи сучасного комп'ютера.

Імовірнісні технології також володіють істотними недоліками при вирішенні практичних задач. Ми проілюстрували роботу імовірнісного підходу на прикладі простої лінійній авторегресійній моделі, проте залежності, що зустрічаються на практиці, часто нелінійні. Навіть якщо і існує проста залежність, то її вигляд наперед невідомий. Якщо ж ми хочемо враховувати для прогнозування курсу акцій декілька взаємозв'язаних чинників (наприклад, кількість операцій, курс долара і т.д.), то доведеться звернутися до побудови багатовимірної статистичної моделі. Проте, такі моделі або припускають гауссовський розподіл спостережень (що не виконується на практиці), або не обґрунтовані теоретично. У багатовимірній статистиці за відсутністю кращого нерідко застосовують малообгрунтовані евристичні методи, які за своєю суттю дуже близькі до технології нейронних мереж.

У останні роки відбувається бурхливий розвиток аналітичних систем нового типу. У їх основі - технології штучного інтелекту, що імітують природні процеси, наприклад, діяльність нейронів мозку або процес природного відбору. При розробці сучасних аналітичних технологій враховується їх здатність:

- розуміння задачі, загального процесу і знання можливостей інших систем і людей, що беруть участь у взаємодії;
- зв'язок з користувачами за допомогою розуміння природної мови, малюнків, зображень, і знаків;
- знання, засновані на здоровому глузді;
- координування ухвалення рішень, планування і дії;
- навчання на попередньому досвіді і адаптація поведінки.

Розуміння цих можливостей в людях і втілення їх при розробці програм є центральним в створеннях новітніх аналітичних технологій, здатних набувати і використовувати знання. Від зростання потужностей для проведення

інформаційного аналізу, ухвалення рішення, гнучкого проектування і виробництва залежить національна конкурентоспроможність. При додаванні інтелекту до комп'ютерних систем усуваються багато обмежень в рішенні реальних задач.

1.2. Поняття інтелектуального аналізу даних.

Більшість організацій накопичують під час своєї діяльності величезні об'єми даних, але головне, що вони хочуть від них отримати - це корисну інформацію. Як можна дізнатися з даних про те, що є вигіднішим для клієнтів організації, як розмістити ресурси ефективним чином або як мінімізувати втрати? Для вирішення цих проблем призначені новітні технології інтелектуального аналізу, які використовують для знаходження моделей і відносин, прихованих в середовищі даних, - моделей, які не можуть бути знайдені звичайними методами [9, 30, 45, 66].

Модель, як і карта - це абстрактне представлення реальності. Карта може указувати на шлях від аеропорту до будинку, але вона не може показати аварію, яка створила пробку, або ремонтні роботи, які ведуться в даний момент і вимагають об'їзду. До тих пір, поки модель не відповідає існуючим реальним відносинам, неможливо отримати сприятливий результат. Існують два види моделей: прогнозуючі і описові. Перші використовують один набір даних з відомими результатами для побудови моделей, які явним чином прогнозують результати для інших наборів, а другі описують залежності в існуючих даних, які у свою чергу використовуються для ухвалення рішень або дій. Звичайно ж, компанія, що давно знаходиться на ринку і знає своїх клієнтів користується безліччю моделей. Технології інтелектуального аналізу можуть не тільки підтвердити ці емпіричні спостереження, але і знайти нові, невідомі раніше моделі. Спочатку це може дати користувачеві лише невелику перевагу, але

якщо його об'єднати по кожному товару і кожному клієнтові, дає великий відрив від тих, хто не використовує таких технологій. З іншого боку, за допомогою методів інтелектуального аналізу можна знайти таку модель, яка приведе до радикального поліпшення у фінансовому і ринковому положенні компанії.

Термін Data Mining отримав свою назву з двох понять: пошуку цінної інформації у великій базі даних (Data) і видобутку гірської руди (Mining). Обидва процеси вимагають або просіювання величезної кількості “сирого” матеріалу, або розумного дослідження і пошуку корисних цінностей. Найчастіше Data Mining переводиться як видобуток даних, витягання інформації, розкопка даних, інтелектуальний аналіз даних, засоби пошуку закономірностей, витягання знань, аналіз шаблонів, "витягання зерен знань з гір даних", розкопка знань в базах даних, інформаційна проходка даних, "промивання" даних. Поняття "виявлення знань в базах даних" (Knowledge Discovery in Databases, KDD) можна вважати синонімом Data Mining.

Поняття Data Mining вперше з'явилося в 1978 році та придбало високу популярність в сучасному трактуванні приблизно з першої половини 1990-х років. Донині обробка і аналіз даних здійснювався в рамках прикладної статистики, при цьому в основному вирішувалися завдання обробки невеликих баз даних.

У основу сучасної технології Data Mining покладена концепція шаблонів (паттернів), що відображають фрагменти багатоаспектних взаємин в даних. Ці шаблони є закономірностями, властивими підвибіркам даних, які можуть бути компактно виражені в зрозумілій людині формі. Пошук шаблонів проводиться методами, не обмеженими рамками апріорних припущень про структуру вибірки і виду розподілів значень аналізованих показників.

Важливе положення Data Mining - нетривіальність розшукуваних шаблонів. Це означає, що знайдені шаблони повинні відображати неочевидні, несподівані (unexpected) регулярності в даних, такі, що становлять так звані приховані знання (hidden Knowledge). До суспільства прийшло розуміння, що

неякісні дані (raw Data) містять глибинний пласт знань, при грамотній розкопці якого можуть бути виявлені справжні самородки (рис.1.4.).

В цілому технологію Data Mining достатньо точно визначає Григорій Піатецький-Шапіро (Gregory Piatetsky-Shapiro) - один із засновників цього напряму: «*Data Mining* - це процес виявлення в "сирих" даних раніше невідомих, нетривіальних, практично корисних і доступних інтерпретації знань, необхідних для ухвалення рішень в різних сферах людської діяльності» [57].

Рис. 1.4. Рівні знань, які вилучаються зданих.

Суть і мету технології Data Mining можна охарактеризувати так: це технологія, яка призначена для пошуку у великих об'ємах даних неочевидних, об'єктивних і корисних на практиці закономірностей. *Неочевидних* - це означає, що знайдені закономірності не виявляються стандартними методами обробки інформації або експертним шляхом. *Об'єктивних* - це означає, що виявлені закономірності повністю відповідатимуть дійсності, на відміну від експертної думки, яка завжди є суб'єктивною. *Практично корисних* - це означає, що висновки мають конкретне значення, якому можна знайти практичне застосування [4, 9, 45, 58].

Приведемо ще декілька визначень поняття Data Mining.

Data Mining - це процес виділення, дослідження і моделювання великих об'ємів даних для виявлення невідомих до цього структур (patterns) з метою досягнення переваг в бізнесі (визначення SAS Institute) [73].

Data Mining - це процес, мета якого - виявити нові значущі кореляції, зразки і тенденції в результаті просіювання великого об'єму даних, що зберігаються, з використанням методик розпізнавання зразків плюс застосування статистичних і математичних методів (визначення Gartner Group).

Data Mining - мультидисциплінарна область, що виникла і розвивалася на базі таких наук як прикладна статистика, розпізнавання образів, штучний інтелект, теорія баз даних та інше (рис. 1.5.).

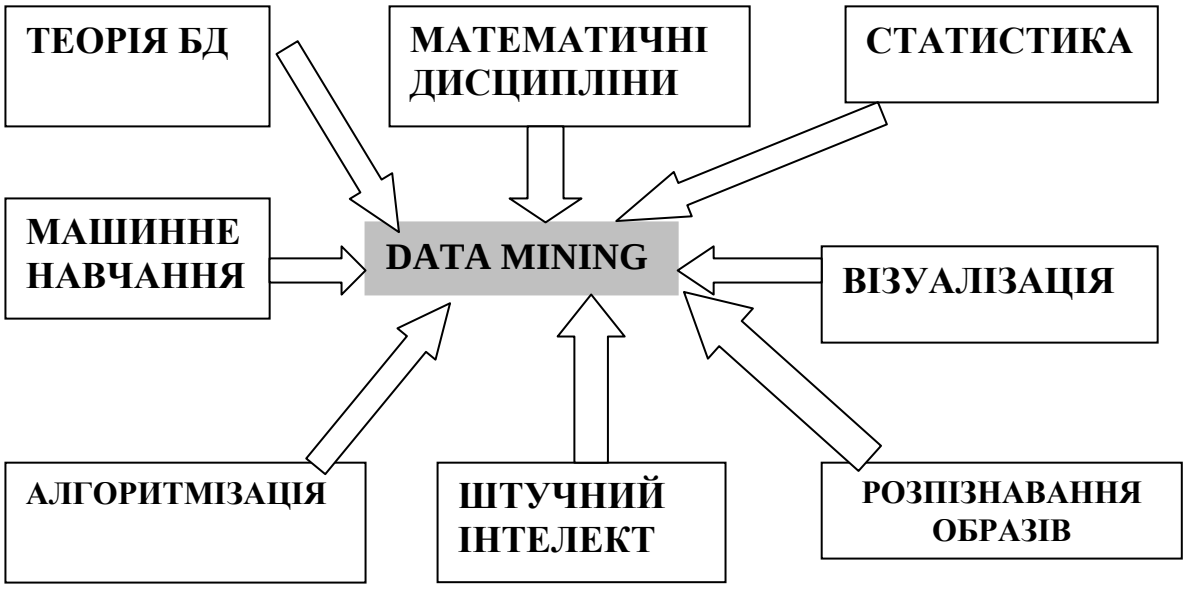


Рис. 1.5. Data Mining як мультидисциплінарна область.

Розглянемо сутність деяких дисциплін, на стику яких з'явилася технологія Data Mining:

- *статистика* - це наука про методи збору даних, їх обробки і аналізу для виявлення закономірностей, властивих явищу, що вивчається. Статистика є сукупністю методів планування експерименту, збору даних, їх уявлення і

узагальнення, а також аналізу і отримання висновків на підставі цих даних. Вона оперує даними, отриманими в результаті спостережень або експериментів;

- *машинне навчання* можна охарактеризувати як процес отримання програмою нових знань. Мітчелл в 1996 році дав таке визначення: "Машинне навчання - це наука, яка вивчає комп'ютерні алгоритми, що автоматично поліпшуються під час роботи". Одним з найбільш популярних прикладів алгоритму машинного навчання є нейронні мережі [52];
- *штучний інтелект* - науковий напрям, в рамках якого ставляться і вирішуються завдання апаратного або програмного моделювання видів людської діяльності, що традиційно вважаються інтелектуальними. Термін інтелект (intelligence) походить від латинського intellectus, що означає розум, розумові здібності людини. Відповідно, штучний інтелект (AI, Artificial Intelligence) тлумачиться як властивість автоматичних систем брати на себе окремі функції інтелекту людини. Штучним інтелектом називають властивість інтелектуальних систем виконувати творчі функції, які традиційно вважаються прерогативою людини.

Поняття Data Mining також тісно пов'язане з *технологіями баз даних* і поняттям дані. Дослідження історичного аспекту цієї проблеми дозволяє виявити основні моменти, пов’язані з появленням систем інтелектуального аналізу даних. Так у 1968 році була введена в експлуатацію перша промислова СУБД система IMS фірми IBM. Вже у 1975 році з'явився перший стандарт асоціації по мовах систем обробки даних - Conference on Data System Languages (CODASYL), що визначив ряд фундаментальних понять в теорії систем баз даних, які до цих пір є основоположними для мережевої моделі даних. Подальший розвиток теорії баз даних пов'язаний з ім’ям американського математика Е.Ф. Кодда, який є творцем реляційної моделі даних. Протягом 80-х років багато дослідників експериментували з новим підходом в напрямах структуризації баз даних і забезпечення до них доступу. Метою цих пошуків було отримання реляційних прототипів для простішого моделювання даних. В

результаті, в 1985 році була створена мова, названий SQL. На сьогоднішній день практично всі СУБД забезпечують даний інтерфейс. Приблизно у цей час з'явилися специфічні типи даних - "графічний образ", "документ", "звук", "карта", типи даних для часу, інтервалів часу, символьних рядків з двобайтовим представленням символів були додані в мову SQL. Результати цих всіх зусиль зробили можливим появу технології Data Mining, сховищ даних, мультимедійних баз даних і web-баз даних [3, 9, 58, 66].

Для успішного проведення процесу знаходження нового знання необхідною умовою є наявність сховища даних. *Сховище даних* - це предметно-орієнтований, інтегрований, прив'язаний до часу, незмінний збір даних для підтримки процесу ухвалення рішень. *Предметна орієнтація* означає, що дані об'єднані в категорії і зберігаються відповідно областям, що вони описують, а не до застосувань, що їх використовують. *Інтегрованість* означає, що дані задовольняють вимогам всього підприємства, а не одній функції бізнесу. Цим, сховище даних гарантує, що однакові звіти, що згенерували для різних аналітиків, міститимуть однакові результати. *Прив'язка до часу* означає, що сховище можна розглядати як сукупність "історичних" даних тобто можна відновити картину на будь-який момент часу. Атрибут часу завжди явно присутній в структурах сховища даних. *Незмінність* означає, що, потрапивши один раз в сховище, дані там зберігаються і не змінюються. У сховищі дані лише додаються. Для організації і експлуатації інформаційного сховища створюється спеціалізоване програмне забезпечення, яке забезпечує ефективну взаємодію з користувачем.

Ключовою можливістю застосування новітніх технологій стало величезне падіння ціни за останні декілька років на пристрої зберігання інформації з десятків доларів за зберігання мегабайта інформації, до десятків центів. Це істотним чином здешевило і збільшило можливості збору і зберігання великих об'ємів інформації. Падіння цін на процесори з одночасним збільшенням їх швидкодії сприяло розвитку технологій, пов'язаних з обробкою величезних масивів інформації. В результаті цього була подолана безліч

бар'єрів, що зустрічаються при знаходженні нового знання в сховищах інформації. Клієнт-серверная архітектура також є необхідним атрибутом технології інтелектуального аналізу даних. Такий підхід надає можливості виконувати найбільш трудомісткі процедури обробки даних на високопродуктивному сервері як розробникам проектів, так і користувачам. На цьому ж сервері можуть зберігатися, і по запитах клієнтів, виконуватися корпоративні проекти.

Технології інтелектуального аналізу даних є важливою частиною ринку сучасних інформаційних технологій. Агентство Gartner Group, що займається аналізом ринків інформаційних технологій, в 1980-х роках ввело термін "Business Intelligence" (BI), діловий інтелект або бізнес-інтелект. Цей термін запропонований для опису різних концепцій і методів, які покращують бізнес рішення шляхом використання систем підтримки ухвалення рішень. У 1996 році агентство уточнило визначення даного терміну: «Business Intelligence - програмні засоби, функціонуючі в рамках підприємства і забезпечуючи функції доступу і аналізу інформації, яка знаходиться в сховищі даних, а також що забезпечують ухвалення правильних і обґрунтованих управлінських рішень». Поняття BI об'єднує в собі різні засоби і технології аналізу і обробки даних масштабу підприємства. На основі цих засобів створюються BI-системи, мета яких - підвищити якість інформації для ухвалення управлінських рішень. BI-системи також відомі під назвою Систем Підтримки Ухвалення Рішень (СППР, DSS, Decision Support System). Ці системи перетворюють дані в інформацію, на основі якої можна ухвалювати рішення, тобто що підтримує ухвалення рішень. Gartner Group визначає склад ринку систем Business Intelligence як набір програмних продуктів наступних класів [36, 40, 58]:

- засоби побудови сховищ даних (Data Warehousing, ХД);
- системи оперативної аналітичної обробки (OLAP);
- інформаційно-аналітичні системи (Enterprise Information Systems, EIS);
- засоби інтелектуального аналізу даних (Data Mining);

- інструменти для виконання запитів і побудови звітів (Query and Reporting tools).

Класифікація Gartner базується на методі функціональних задач, де програмні продукти кожного класу виконують певний набір функцій або операцій з використанням спеціальних технологій.

Приведемо декілька коротких цитат найбільш впливових членів бізнес-організацій, які є експертами в цій відносно новій технології. Керівництво з придбання продуктів Data Mining (Enterprise Data Mining Buying Guide) компанії Aberdeen Group: "Data Mining - технологія здобичі корисної інформації з баз даних. Проте у зв'язку з істотними відмінностями між інструментами, досвідом і фінансовим станом постачальників продуктів, підприємствам необхідно ретельно оцінювати передбачуваних розробників Data Mining і партнерів. Щоб максимально використовувати потужність інструментів Data Mining комерційного рівня, підприємству необхідно вибрати, очистити і перетворити дані, іноді інтегрувати інформацію, здобуту із зовнішніх джерел, і встановити спеціальне середовище для роботи Data Mining алгоритмів. Результати Data Mining великою мірою залежать від рівня підготовки даних, а не від "чудових можливостей" якогось алгоритму або набору алгоритмів. Близько 75% роботи над Data Mining полягає в зборі даних, який здійснюється ще до того, як запускаються самі інструменти. Безграмотно застосувавши деякі інструменти, підприємство може безглуздо розтратити свій потенціал, а іноді і мільйони доларів" [72].

Думка Херба Едельштайна (Herb Edelstein), відомого в світі експерта в області Data Mining, сховищ даних і CRM: "Недавнє дослідження компанії Two Crows показало, що Data Mining знаходиться все ще на ранній стадії розвитку. Багато організацій цікавляться цією технологією, але лише деякі активно упроваджують такі проекти. Вдалося з'ясувати ще один важливий момент: процес реалізації Data Mining на практиці виявляється складнішим, ніж очікується. IT-команди захопилися міфом про те, що засоби Data Mining прості у використанні. Передбачається, що досить запустити такий інструмент на

терабайтной базі даних, і вмить з'явиться корисна інформація. Насправді, успішний Data Mining-проект вимагає розуміння суті діяльності, знання даних і інструментів, а також процесу аналізу даних" [61].

Перш ніж використовувати технологію Data Mining, необхідно ретельно проаналізувати її проблеми, обмеження і критичні питання, з нею зв'язані, а також зрозуміти, чого ця технологія не може. Зокрема, Data Mining не може замінити аналітика. Також технологія не може дати відповіді на ті питання, які не були задані. Вона не може замінити аналітика, а всього лише дає йому могутній інструмент для полегшення і поліпшення його роботи. Оскільки дана технологія є мультидисциплінарною областю, для розробки додатку, включаючого Data Mining, необхідно задіювати фахівців з різних областей, а також забезпечити їх якісну взаємодію.

Різні інструменти Data Mining мають різний ступінь "доброзичливості" інтерфейсу і вимагають певної кваліфікації користувача. Тому програмне забезпечення повинне відповідати рівню підготовки користувача. Використання Data Mining повинне бути нерозривно пов'язане з підвищенням кваліфікації користувача. Проте фахівців з Data Mining, які б добре розбиралися в бізнесі, поки що мало [2, 25].

Успішний аналіз вимагає якісної передобробки даних. За ствердженням аналітиків і користувачів баз даних, процес передобробки може зайняти до 80% відсотків всього Data Mining-процесу. Тому, щоб технологія працювала на себе, буде потрібно багато зусиль і часу, які йдуть на попередній аналіз даних, вибір моделі і її коректування.

За допомогою Data Mining можна відшукувати дійсно дуже цінну інформацію, яка незабаром дасть великі дивіденди у вигляді фінансової і конкурентної вигоди. Проте Data Mining достатньо часто робить безліч помилкових відкриттів, що не мають сенсу. Багато фахівців стверджують, що засоби Data Mining можуть видавати величезну кількість статистично недостовірних результатів. Щоб цього уникнути, необхідна перевірка адекватності отриманих моделей на тестових даних.

Слід також зазначити, що якісна Data Mining-програма може коштувати достатньо дорого для компанії. Варіантом служить придбання вже готового рішення з попередньою перевіркою його використання, наприклад на демо-версії з невеликою вибіркою даних. Засоби Data Mining, на відміну від статистичних, теоретично не вимагають наявності строго певної кількості ретроспективних даних. Ця особливість може стати причиною виявлення недостовірних, помилкових моделей і, як результат, ухвалення на їх основі невірних рішень. Необхідно здійснювати контроль статистичної значущості виявлених знань.

Data Mining має досить суттєві відмінності від інших методів аналізу даних. Традиційні методи аналізу даних (статистичні методи) і OLAP в основному орієнтовані на перевірку наперед сформульованих гіпотез (verification-driven Data Mining) і на "грубий" розвідувальний аналіз, що становить основу оперативної аналітичної обробки даних (Online Analytical Processing, OLAP), тоді як одне з основних положень Data Mining - пошук неочевидних закономірностей. Інструменти Data Mining можуть знаходити такі закономірності самостійно і також самостійно будувати гіпотези про взаємозв'язки. Оскільки саме формулювання гіпотези щодо залежностей є найскладнішим завданням, перевага Data Mining в порівнянні з іншими методами аналізу є очевидною. Більшість статистичних методів для виявлення взаємозв'язків в даних використовують концепцію усереднювання по вибірці, що приводить до операцій над неіснуючими величинами, тоді як Data Mining оперує реальними значеннями. OLAP більше підходить для розуміння ретроспективних даних, Data Mining спирається на ретроспективні дані для отримання відповідей на питання про майбутньому.

Потенціал Data Mining дає "зелене світло" для розширення меж застосування технології. Щодо перспектив Data Mining можливі наступні напрями розвитку:

- виділення типів предметних областей з відповідними ним евристиками, формалізація яких полегшить рішення відповідних задач Data Mining, що відносяться до цих областей;
- створення формальних мов і логічних засобів, за допомогою яких буде формалізовані міркування і автоматизація яких стане інструментом рішення задач Data Mining в конкретних наочних областях;
- створення методів Data Mining, здатних не тільки витягувати з даних закономірності, але і формувати якісь теорії, що спираються на емпіричні дані;
- подолання істотного відставання можливостей інструментальних засобів Data Mining від теоретичних досягнень в цій області.

Якщо розглядати майбутнє Data Mining в короткостроковій перспективі, то очевидно, що розвиток цієї технології найбільш направлений до областей, пов'язаних з бізнесом. Продукти Data Mining можуть стати такими ж звичайними і необхідними, як електронна пошта, і, наприклад, використовуватися користувачами для пошуку найнижчих цін на певний товар або найбільш дешевих квитків.

У довгостроковій перспективі майбутнє Data Mining є таким, що дійсно захоплює - це може бути пошук інтелектуальними агентами як нового вигляду лікування різних захворювань, так і нового розуміння природи всесвіту.

Проте Data Mining таїть в собі і потенційну небезпеку - адже все більша кількість інформації стає доступною через всесвітню мережу, у тому числі і відомості приватного характеру, і все більше знань можливо добути з неї. Не так давно найбільший онлайновий магазин "Amazon" опинився в центрі скандалу з приводу отриманого ним патенту "Методи і системи допомоги користувачам при покупці товарів", який є не що інше як черговий продукт Data Mining, призначений для збору персональних даних про відвідувачів магазина. Нова методика дозволяє прогнозувати майбутні запити на підставі фактів покупок, а також робити висновки про їх призначення. Мета даної методики - те, про що мовилося вище - отримання як можна більшої кількості

інформації про клієнтів, у тому числі і приватного характеру (стан, вік, переваги і т.д.). Таким чином, збираються дані про приватне життя покупців магазина, а також членів їх сімей, включаючи дітей. Останнє заборонене законодавством багатьох країн - збір інформації про неповнолітніх можливий там тільки з дозволу батьків.

Дослідження відзначають, що існують як успішні рішення, які використовують Data Mining, так і невдалий досвід застосування цієї технології. Напрями, де застосування технологій Data Mining, швидше за все, будуть успішними, мають такі особливості:

- вимагають рішень, заснованих на знаннях;
- мають навколишнє середовище, що змінюється;
- мають доступні, достатні і значущі дані;
- забезпечують високі дивіденди від правильних рішень.

Сфера застосування Data Mining нічим не обмежена - вона скрізь, де є які-небудь дані. Але в першу чергу методи Data Mining сьогодні, м'яко кажучи, заінтригували комерційні підприємства, що розгортають проекти на основі інформаційних сховищ даних (Data Warehousing). Досвід багато таких підприємств показує, що віддача від використання Data Mining може досягати 1000%. Наприклад, відомі повідомлення про економічний ефект, в 10-70 разів що перевищив первинні витрати від 350 до 750 тис. дол.; про проект в 20 млн. дол., який окупився всього за 4 місяці. Інший приклад - річна економія 700 тис. дол. за рахунок впровадження Data Mining в мережі універсамів у Великобританії. Data Mining представляють велику цінність для керівників і аналітиків в їх повсякденній діяльності. Ділові люди усвідомили, що за допомогою методів Data Mining вони можуть отримати відчутні переваги в конкурентній боротьбі [19, 30, 58,66].

Розглянемо деякі бізнес-приклади Data Mining.

Роздрібна торгівля. Підприємства роздрібної торгівлі сьогодні збирають докладну інформацію про кожну окрему покупку, використовуючи кредитні картки з маркою магазина і комп'ютеризовані системи контролю. Ось

типові задачі, які можна вирішувати за допомогою Data Mining у сфері роздрібної торгівлі:

- аналіз купівельної корзини (аналіз схожості) призначений для виявлення товарів, яких покупці прагнуть придбати разом. Знання купівельної корзини необхідне для поліпшення реклами, вироблення стратегії створення запасів товарів і способів їх розкладки в торгових залах;
- дослідження часових шаблонів допомагає торговим підприємствам приймати рішення про створення товарних запасів. Воно дає відповіді на питання типу "Якщо сьогодні покупець придбав відеокамеру, то через який час він найімовірніше купить нові батареї і плівку?";
- створення прогнозуючих моделей дає можливість торговим підприємствам дізнаватися характер потреб різних категорій клієнтів з певною поведінкою, наприклад, таких, що купують товари відомих дизайнерів або таких, що відвідують розпродажі. Ці знання потрібні для розробки точно направлених, економічних заходів щодо просування товарів.

Банківська справа. Досягнення технології Data Mining використовуються в банківській справі для вирішення наступних поширених завдань:

- виявлення шахрайства з кредитними картками. Шляхом аналізу минулих транзакцій, які згодом виявилися шахрайськими, банк виявляє деякі стереотипи такого шахрайства;
- сегментація клієнтів. Розбиваючи клієнтів на різні категорії, банки роблять свою маркетингову політику більш цілеспрямованою і результативною, пропонуючи різні види послуг різним групам клієнтів**;**
- прогнозування змін клієнтури. Data Mining допомагає банкам будувати прогнозні моделі цінності своїх клієнтів, і відповідним чином обслуговувати кожну категорію.

Телекомунікації. В області телекомунікацій методи Data Mining допомагають компаніям енергійніше просувати свої програми маркетингу і

ціноутворення, щоб утримувати існуючих клієнтів і привертати нових. Серед типових заходів відзначимо наступні:

- аналіз записів про докладні характеристики викликів. Призначення такого аналізу - виявлення категорій клієнтів з схожими стереотипами користування їх послугами і розробка привабливих наборів цін і послуг;
- виявлення лояльності клієнтів. Data Mining можна використовувати для визначення характеристик клієнтів, які, один раз скориставшись послугами даної компанії, з великою часткою вірогідність залишиться їй вірними. У результаті засоби, що виділяються на маркетинг, можна витрачати там, де віддача більше всього.

Страхування. Страхові компанії протягом ряду років накопичують великі об'єми даних. Тут обширне поле діяльності для методів Data Mining:

- виявлення шахрайства. Страхові компанії можуть понизити рівень шахрайства, відшукуючи певні стереотипи в заявах про виплату страхового відшкодування, що характеризують взаємини між юристами, лікарями і заявниками;
- аналіз ризику. Шляхом виявлення поєднань чинників, пов'язаних із сплаченими заявами, страховики можуть зменшити свої втрати за зобов'язаннями. Відомий випадок, коли в США крупна страхова компанія виявила, що суми, виплачені за заявами людей, що є в браку, удвічі перевищує суми за заявами самотніх людей. Компанія відреагувала на це нове знання переглядом своєї загальної політики надання знижок сімейним клієнтам.

Управління виробництвом, менеджмент якості. Шляхом аналізу даних автоматизованого виробництва і відхилень від нього можна ідентифікувати проблеми на етапах виробництва як з погляду якості, так і з погляду збереження темпу виробництва. На підставі такої встановленої інформації можна, наприклад, у виробничий процес ввести етап додаткового контролю, завдяки якому вже в процесі виробництва будуть виявлені розроблені вироби, які після закінчення виробничого процесу не пройдуть вихідний контроль.

Молекулярна генетика і генна інженерія. Мабуть найгостріше завдання виявлення закономірностей в експериментальних даних стоїть в молекулярній генетиці і генній інженерії. Тут вона формулюється як визначення так званих маркерів, під якими розуміють генетичні коди, контролюючі ті або інші фенотипічні ознаки живого організму. Такі коди можуть містити сотні, тисячі і більш зв'язаних елементів. На розвиток генетичних досліджень виділяються великі кошти. Останнім часом в даній області виник особливий інтерес до застосування методів Data Mining. Відомо декілька крупних фірм, що спеціалізуються на застосуванні цих методів для розшифровки генома людини і рослин.

Медицина. Відомо багато експертних систем для постановки медичних діагнозів. Вони побудовані головним чином на основі правил, що описують поєднання різних симптомів різних захворювань. За допомогою таких правил дізнаються не тільки, на що хворий пацієнт, але і як потрібно його лікувати. Правила допомагають вибирати засоби медикаментозної дії, визначати свідчення - протипоказання, орієнтуватися в лікувальних процедурах, створювати умови найбільш ефективного лікування, передбачати результати призначеного курсу лікування і т.п. Технології Data Mining дозволяють виявляти в медичних даних шаблони, які складають основу вказаних правил.

Прикладна хімія. Методи Data Mining знаходять широке застосування в прикладній хімії (органічної і неорганічної). Тут нерідко виникає питання про з'ясування особливостей хімічної будови тих або інших з'єднань, що визначають їх властивості. Особливо актуальне таке завдання при аналізі складних хімічних сполук, опис яких включає сотні і тисячі структурних елементів і їх зв'язків.

Можна привести ще багато прикладів різних областей знання, де методи Data Mining грають провідну роль. Особливість цих областей полягає в їх складній системній організації. Вони відносяться головним чином до суперскладного рівня організації систем, закономірності якого не можуть бути достатньо точно описані на мові статистичних або інших аналітичних

математичних моделей. Дані у вказаних областях неоднорідні, гетерогенні, нестаціонарні і часто відрізняються високою розмірністю.

1.3. Етапи та методи знаходження нових знань.

Важливо розуміти, що побудова моделі інтелектуального аналізу даних є складовою частиною масштабнішого процесу, який включає всі етапи, починаючи з визначення базової проблеми, яку модель вирішуватиме, до розгортання моделі в робочому середовищі. Даний процес може бути заданий за допомогою наступних шести базових кроків (рис. 1. 6.):

1. Постановка задачі.
2. Підготовка та огляд даних:

- оцінювання даних;
- об'єднання і очищення даних;
- відбір даних;
- перетворення.

3. Побудова моделей:

- оцінка і інтерпретація;
- зовнішня перевірка.

4. Використання моделей.
5. Нагляд за моделлю.

Хоча процес, що ілюструється за допомогою рисунку 1.6, носить циклічний характер, кожен крок не обов'язково веде безпосередньо до наступного кроку. Створення моделі інтелектуального аналізу даних є динамічний ітеративний процес. Виконавши огляд даних, користувач може виявити, що існуючих даних недостатньо для створення необхідних моделей інтелектуального аналізу даних, що, відповідно, веде до необхідності пошуку додаткових даних. Можна розробити декілька моделей і зрозуміти, що вони не

вирішують сформульованої задачі. Отже, потрібна зміна характеристик завдання.

Рис 1.6. Етапи інтелектуального аналізу даних.

Може виникнути необхідність в оновленні вже розгорнених моделей за рахунок нових даних, що поступили. Таким чином, важливо розуміти, що створення моделі інтелектуального аналізу даних є процесом і що кожен крок такого процесу може бути повторений стільки раз, скільки необхідно для створення ефективної моделі. Розглянемо докладніше кожний з цих етапів:

*Визначення проблеми***.** Для того, щоб повніше використовувати всі переваги інтелектуальних технологій необхідно ясно представити мету майбутнього аналізу. Побудова моделі проводиться залежно від мети. Якщо необхідно збільшити прибуток торгової організації, то для цілей: "збільшення кількості продажів" і "збільшення ефективності реклами" необхідно будувати різні моделі. На цьому ж етапі визначаються способи оцінювання результатів майбутнього проекту і можливі витрати на його реалізацію.

Підготовка та огляд даних. Це є найтриваліший етап, який може займати від 50% до 85% часу всього процесу знаходження нового знання. На цьому етапі необхідно визначити джерела отримання даних. Це можуть бути дані, накоплені самою організацією або зовнішні дані від загальнодоступних

джерел (відомості про погоду або перепис населення) або приватних джерел (різні архівні дані, бази нотаріальних контор і ін.).

Оцінювання даних. При побудові моделі необхідно пам'ятати одне правило, що стосується коректності вхідних даних: "Якщо на вхід задачі поступає "сміття", то і результатом теж буде "сміття". Вхідні дані можуть знаходитися в одній базі або в декількох. Перед "завантаженням" даних в сховище необхідно врахувати, що різні джерела даних можуть бути спроектовані під певні задачі і, відповідно, виникають проблеми, пов'язані з об'єднанням даних: різні формати представлення числових даних (наприклад, цілі або вещественні); різне кодування даних (наприклад, різний формат дат); різні способи зберігання даних; різні одиниці вимірювання (дюйми і сантиметри); а також частота збору даних і дата останнього оновлення. Навіть, якщо дані знаходяться в одній базі, то все одно треба звертати увагу на пропущені значення і значення, нереальної величини, так звані "викиди". Аналітик винен завжди знати, як, де і за яких умов збираються дані, і бути упевненим, що всі дані, які використовуються для проведення аналізу зміряні однаковим способом.

Об'єднання і очищення даних. На цьому етапі відбувається побудова сховища даних для подальшої обробки, тобто, відбувається наповнення сховища або додавання до нього даних, відібраних на попередніх етапах. В цей же час відбувається очищення, тобто виправлення всіх виявлених помилок. Існують різні аспекти очищення даних. Всі вони направлені на знаходження і виправлення помилок, допущених на етапі збору інформації. Помилкою в даних можуть вважатися: пропущене значення, неможлива подія (невірно набране значення - "викид"). Корекція відбувається на основі здорового глузду, використання правил або із залученням експерта, добре обізнаного з предметною областю. Запис в базі даних, в якому є помилка, повинен бути виправлений або, в спірних випадках, виключений з подальшого розгляду. Після перевірки даних, вони перетворюються і форматуються відповідно результатам оцінювання. Це робиться для більшої зручності спостереження за

даними. Дані дискретних подій перетворюються на спеціально розроблену або стандартну форму, в якій відбивається час і опис подій. Якщо користувачі легко розбиратимуться в цій формі, вони зможуть швидко вивчити події, які були в основі побудови цієї форми. Може здатися, що цей крок дублює етап збору даних, але насправді це два зовсім різних етапа. На першому з них відбувається відбір даних для прискорення машинної обробки інформації без втрати якості, на другому дані доводяться до вигляду, зручному для візуального контролю користувача. Людина, яка проводить аналіз, може повніше уявити собі вхідні дані. Це необхідно для різного роду звітів, коли потрібно коротко охарактеризувати вхідні дані, вживані для аналізу.

Відбір даних. Якщо сховище сформоване і визначені типи моделей, які будуть побудовані для вирішення задач, відбувається відбір даних необхідних саме для цих моделей. Мається на увазі не тільки зменшення кількості записів в базі по певній умові, але також і зміну кількості полів, злиття різних таблиць в одну, або, навпаки, створення на основі однієї таблиці декілька. Тобто, перетворення відбувається в "трьох вимірюваннях": по кількості записів, по кількості полів і по структурі.

Перетворення даних. Служить для збагачення отриманої бази, тобто додавання різних відносин на основі існуючих полів (не просто "ціна" і "кількість", а їх твір - "загальна сума", не борг і дохід, а відношення борга до доходу), додавання інтервалів (по номеру місяця можна поставити номер кварталу, а відсоток виконання плану можна доповнити характеристиками "добре", "задовільно"), додавання критичних значень (максимум, середнє, мінімум).

Побудова моделі є ітераційний процес, тобто, необхідно побудувати ряд моделей для знаходження однієї, що найбільш задовольняє поставленим цілям. Моделі можна розділити на дві групи:

- контрольовані (моделі класифікації, регресії, прогнозування часових послідовностей);
- неконтрольовані (кластеризація, асоціація і послідовність).

Після того, як визначений тип моделі, необхідно вибрати алгоритм побудови моделі або технологію знаходження знання.

Суть процесу побудови контрольованої моделі зводиться до знаходження залежностей на одній частині даних ("навчання моделі") і перевірки цих залежностей на іншій частині даних (оцінка точності). Модель вважається побудованою, якщо завершується цикл "навчання" і перевірок. Якщо точність моделі при чергових ітераціях не поліпшується, то це говорить про завершення побудови моделі. Оскільки "навчальні" і тестові дані знаходяться в одній базі даних, то часто виникає необхідність в третьому наборі даних - контрольному, який вибирається з таких даних, що не перетинаються з "навчальними" і тестовими. Він потрібний для незалежного оцінювання точності моделі. Як правило, всі три набори даних належать множині даних, необхідній для реалізації певного проекту[2, 9, 45].

Найбільш відомий тестовий метод - називається простою оцінкою. В цьому випадку розподіл даних на два набори відбувається випадковим чином. Відношення кількості тестових даних до кількості даних, на яких відбувається побудова моделі повинно бути в межах від 5% до 33%. Після побудови моделі, її використовують для передбачення значень на тестовому наборі. Мірою точності моделі вважають відношення кількості вдалих результатів до загальної кількості прикладів в тестовому наборі (можна використовувати таку змінну, як міра неточності, яка дорівнює 1, - "міра точності") .

Якщо для побудови моделі використовується не дуже велика база даних, то застосовується так звана перехресна оцінка точності. В цьому випадку дані випадковим чином діляться на дві приблизно рівні частини. Після цього модель будуватиметься на одній з них, а інша використовується для визначення точності. Потім частини бази міняються ролями. Отримані дві незалежні оцінки точності об'єднуються (як середнє арифметичне або іншим способом) для якнайкращої оцінки точності моделі, побудованої на всій базі.

Для ще менших баз, в декілька тисяч записів, використовується n-перехресна оцінка точності. В цьому випадку база ділиться на n приблизно

рівних непересічних груп. Далі перша з цих груп стає тестовим набором, а інші групи об'єднуються, і на їх основі відбувається побудова моделі. Отримана модель використовується для передбачення значень для тестового набору і таким чином виходить перше значення точності. Аналогічним чином набувають всі n незалежних значень точності. Середнє з них є точністю всієї моделі.

Ще один спосіб використовується для знаходження точності в маленьких базах даних. В цьому випадку модель будуватиметься на основі даних всієї бази. Після цього випадковим чином із записів бази створюється безліч тестових наборів (мінімум 200, а іноді навіть більше 1000). Один запис може бути присутнім в різних тестових наборах. Для будь-якої з них визначається точність. Середнє з них є точністю всієї моделі.

Після того, як побудова моделі завершена, можна корегувати модель, використовуючи інші параметри або навіть змінити алгоритм побудови моделі, оскільки ніколи не можна сказати, який алгоритм, яка технологія знаходження знання дасть кращі результати. Не можна бути упевненим, що певна технологія працюватиме краще всього. Часто доводиться будувати велику кількість моделей і оцінювати кожну для знаходження кращої. Окрім цього, для різних моделей необхідна різна підготовка даних і неминуче повторення кроків. Все це збільшує час знаходження кращої моделі, тому необхідно застосовувати технології паралельних обчислень.

*Оцінка і інтерпретація***.** Після побудови моделі необхідно оцінити результати і пояснити (інтерпретувати) їх значущість. При оцінці моделі обчислюється точність, але треба пам'ятати, що це значення вірно лише для даних, на яких модель побудована і бути готовим, що нові дані, до яких надалі застосовуватиметься модель, можуть відрізнятися від результатних невідомим чином.

Зовнішня перевірка. Висока точність моделі не є гарантією того, що модель правильно відображає реальне середовище. Однією з причин, є існування так званих неявних припущень в моделі. Тобто, сам по собі

коефіцієнт інфляції не може бути частиною моделі, що пояснює схильність покупців до покупки чи того іншого товару, але різка зміна цього коефіцієнта з 3% до 20% вже, напевно, може пояснити таку поведінку. Інша причина - це існування неминучих проблем з даними, що приводять до некоректності моделі, тому дуже важливо перевірити модель в реальному середовищі. Наприклад, якщо модель використовується для відбору кандидатів для цільової реклами, то можна зробити тестову розсилку для перевірки моделі на невеликому об'ємі даних. Якщо модель використовується для передбачення ризику неповернення кредиту, то слід піддати випробуванню цю модель на невеликій кількості претендентів на позику. Чим більше ризик, пов'язаний з некоректністю моделі, тим більше важливо провести попередні експерименти для перевірки моделі перед її експлуатацією.

Використання моделі. Після побудови і оцінки моделі, її можна використовувати різними способами. Наприклад, аналітик може подивитися групи, яка визначила модель кластеризації, графіки ефективності моделі або отримані правила. Іноді аналітик може використовувати модель для вибору деяких записів з бази даних для проведення додаткового аналізу. Базуючись на результатах використання моделі, аналітик може рекомендувати дії, які можна починати в діловій сфері. Проте, часто технології інтелектуальних обчислень - це частина автоматизованої системи (наприклад, знаходження кредитних ризиків, визначення можливості втрати клієнтів і ін.), тобто модель вбудовується в систему, яку аналітик або менеджер можуть застосовувати для ухвалення рішення. З іншого боку, модель можна включати в систему, що генерує деяку дію (наказ), коли прогнозована величина починає виходити за межі якихось значень. Загалом, методи інтелектуальних обчислень, це невелика, хоч і важлива частина кінцевого програмного продукту. Процедура знаходження знання за допомогою цих методів може об'єднуватися із знаннями експертів і застосовуватися до даних в базах.

Спостереження за моделлю. Коли модель починає працювати в реальному середовищі, то необхідно вимірювати точність моделі на реальних

даних. Проте, навіть якщо модель працює добре, і можна вважати, що робота на цьому закінчується, то все одно необхідно продовжувати спостереження за моделлю. Всі системи мають властивість розвиватися, і отримані дані (їх структура, точність, періодичність) теж міняються. Зовнішня змінна, така як коефіцієнт інфляції, своєю зміною теж можуть впливати на поведінку людей і на чинники, що впливають на цю зміну. Час від часу модель необхідно піддавати повторному тестуванню, і навіть перебудові.

Простим способом спостереження діяльності моделі є графіки розбіжностей між величинами, що передбачаються, і реальними значеннями. Вони прості для побудови і розуміння і можуть вбудовуватися в програмні продукти. Така автоматизована система може стежити сама за собою і сповіщати користувача, коли величина цих розбіжностей починає виходити за певний граничний рівень [9, 45, 58, 61].

Можна виділити принаймні шість методів виявлення і аналізу знань:

- класифікація;
- регресія;
- прогнозування часових послідовностей (рядів);
- кластеризація;
- асоціація;
- послідовність.

Перші три використовуються головним чином для передбачення, тоді як останні зручні для опису існуючих закономірностей в даних.

Класифікація - найпоширеніша модель інтелектуального аналізу даних. З її допомогою виявляються ознаки, що характеризують групу, до якої належить той або інший об'єкт. Це робиться за допомогою аналізу вже класифікованих об'єктів і формулювання деякого набору правил. Наприклад, в багатьох видах бізнесу проблемою є втрата постійних клієнтів. Класифікація допомагає виявити характеристики "нестійких" покупців і створити модель, яка передбачає, хто саме схильний піти до іншого постачальника. Використовуючи її, можна визначити ефективні види знижок і інші вигідні пропозиції, що діють

для разних покупців. Завдяки цьому, можна утримати клієнтів, витративши стільки грошей, скільки необхідно.

Певний ефективний класифікатор може використовуватися для класифікації нових записів в базі даних у вже існуючі класи і в цьому випадку він набуває характеру прогнозу. Наприклад, класифікатор, що уміє ідентифікувати ризик віддачі позики, може бути використаний для ухвалення рішення, чи великий ризик надання позики певному клієнтові. Тобто, класифікатор використовується для прогнозування вірогідності повернення позики.

Регресійний аналіз використовується, коли стосунки між змінними можуть бути виражені кількісно у вигляді деякої комбінації цих змінних. Отримана комбінація використовується для передбачення значення, яке може приймати цільова (залежна) змінна, що обчислюється на заданому наборі значень вхідних (незалежних) змінних. У простому випадку, для цього використовуються стандартні статистичні методи, такі як лінійна регресія, але більшість реальних моделей не укладаються в її рамки. Наприклад, розміри продажів або фондові ціни складні для передбачення, оскільки можуть залежати від комплексу взаємин змінних.

Прогнозування часових послідовностей. Основою для будь-яких систем прогнозування служить історична інформація, що зберігається в інформаційних сховищах у вигляді часових рядів. Якщо можна побудувати математичну модель і знайти шаблони, що адекватно відображують цю динаміку, є вірогідність, що з їх допомогою можна передбачати і поведінку системи в майбутньому. Прогнозування часових послідовностей дозволяє на основі аналізу поведінки часових рядів оцінювати майбутні значення прогнозованих змінних. Ці моделі повинні включати особливі ознаки часу: ієрархію періодів (місяць-квартал-рік), особливі відрізки часу (пяти- шести або семиденний робочий тиждень), сезонність, свята та ін.

Кластеризація відрізняється від класифікації тим, що класи заздалегідь не задані і за допомогою моделі кластеризації засоби інтелектуальних обчислень самостійно створюють однорідні групи даних.

Асоціація відноситься до аналізу структури і застосовується, коли декілька подій зв'язано між собою. Класичний приклад аналізу структури покупок відноситься до представлення придбання якої-небудь кількості товарів як одиночної економічної операції (транзакції). Оскільки велика кількість покупок відбувається в супермаркетах, а покупці для зручності використовують корзини, куди і складається весь товар, то для знаходження асоціацій служить аналіз вмісту корзини. Метою підходу є знаходження трендів (однакових ділянок) серед великого числа транзакцій, які можна використовувати для пояснення поведінки покупців. Така інформація може бути використана для регулювання запасів, зміни розміщення товарів на території магазина і ухвалення рішення по проведенню рекламної кампанії для збільшення продажів або для просування певного вигляду продукції. Наприклад, дослідження, проведене в супермаркеті, може показати, що 65% людей купують картопляні чипси, беруть також і "кока-колу", а за наявності знижки за такий комплект "кока-колу" купують в 85% випадків. Маючи такі дані, менеджерам легко оцінити, наскільки дієва надана знижка.

Хоча цей підхід прийшов виключно з роздрібної торгівлі, він може також застосовуватися у фінансовій сфері для аналізу портфеля коштовних паперів і знаходження наборів фінансових послуг, які клієнти часто купують разом. Це може використовуватися для створення деякого набору послуг, як частини кампанії по стимулюванню продажів.

Послідовність має місце, якщо існує ланцюжок зв'язаних в часі подій. Традиційний аналіз структури покупок має справу з набором товарів, які представляють одну транзакцію. Варіант такого аналізу зустрічається, якщо існує додаткова інформація (номер кредитної карти клієнта або номер його банківського рахунку) для скріплення різних покупок в єдину тимчасову серію. У такій ситуації важливе не лише співіснування даних усередині однієї

транзакції, але і порядок, в якому ці дані з'являються в різних транзакціях і час між цими транзакціями. Правила, що встановлюють ці стосунки, можуть бути використані для визначення типового набору попередніх продажів, які можуть повести за собою наступні продажі певного товару. Після покупки будинку в 45% випадків протягом місяця купується і нова кухонна плита, а в перебігу наступних двох тижнів 60% новоселів обзаводяться холодильником.

1.4. Основні моделі інтелектуальних обчислювань.

Розглянемо основні види моделей, які використовуються для знаходження нового знання на основі даних інформаційного сховища. Метою інтелектуальних технологій є знаходження нового знання, яке користувач може надалі застосувати для поліпшення результатів своєї діяльності. Результат моделювання - це виявлення відносин в даних [4, 30, 57, 70].

На практиці широке застосування знайшли такі види (алгоритми) інтелектуальних обчислень:

- нейронні мережі;
- дерева рішень;
- системи роздумів на основі аналогічних випадків;
- алгоритми визначення асоціацій і послідовностей;
- нечітка логіка;
- генетичні алгоритми;
- еволюційне програмування;
- візуалізація даних;
- комбіновані методи.

Нейронні мережі це системи з архітектурою, що умовно імітують роботу нейронів. Математична модель нейрона є деяким універсальним нелінійним елементом з можливістю широкої зміни і настроювання його характеристик. Нейронні мережі є сукупністю зв'язаних між собою прошарків

нейронів, які отримують вхідні дані, здійснюють їх обробку і генерують на виході результат. Між вузлами видимих вхідного і вихідного прошарків може знаходитися певне число прихованих прошарків. Нейронні мережі реалізують непрозорий процес. Це означає, що побудована модель, як правило, не має чіткої інтерпретації. Багато пакетів, які реалізують алгоритми нейронних мереж, застосовуються у сфері обробки комерційної інформації, при розпізнаванні образів, розшифровки рукописного тексту, інтерпретації кардіограм. Апаратні або програмні реалізації алгоритмів нейромереж називаються нейрокомп'ютером. Його основними особливостями є:

- нейрокомп'ютери дають стандартний спосіб рішення багатьох нестандартних задач. І неважливо, що спеціалізована машина краще вирішить один клас завдань. Важливіше, що один нейрокомп'ютер вирішить і цю задачу, і іншу, і третю і не треба кожного разу проектувати спеціалізовану ЕОМ, нейрокомп'ютер зробить все сам і майже не гірше;
- замість програмування застосовується навчання. Нейрокомп'ютер навчається, потрібно лише формувати навчальну множину. Таким чином, робота програміста замінюється новою роботою вчителя. Краще це чи гірше? Не те, ні друге. Програміст наказує машині виконати всі деталі роботи, вчитель створює "навчальне середовище", до якого пристосовується нейрокомп'ютер. З'являються нові можливості для роботи;
- нейрокомп'ютери ефективні там, де потрібний аналог людської інтуїції, зокрема, для розпізнавання образів, читання рукописних текстів, підготовки аналітичних прогнозів, перекладу з однієї мови на іншій і т.п. Саме для таких завдань зазвичай важко скласти явний алгоритм;
- нейронні мережі дозволяють створити ефективне програмне і математичне забезпечення для комп'ютерів з високим ступенем розпаралелювання обробки;
- нейрокомп'ютери "демократичні", вони прості, як текстові процесори, тому з ними може працювати будь-якій, навіть зовсім недосвідчений користувач (рис 1.7.).

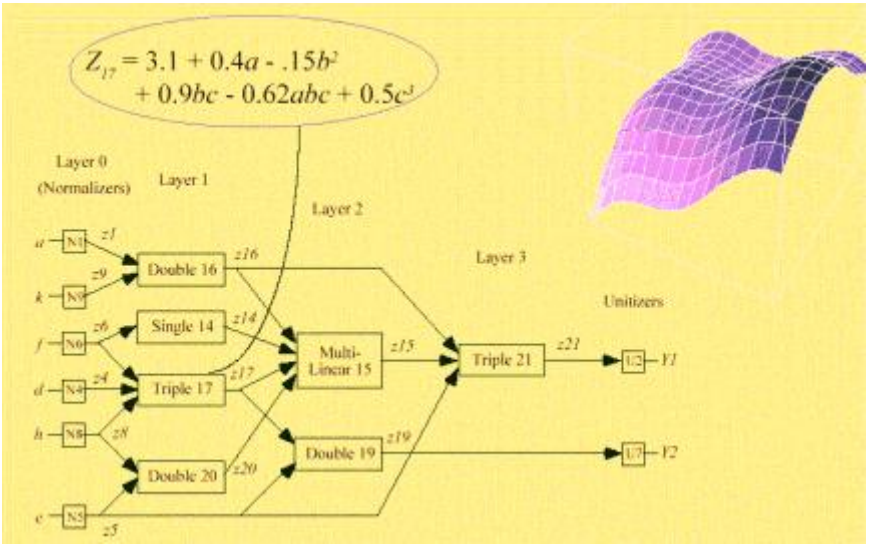


Рис.1.7. Поліномінальна нейромережа.

Дерева рішень - метод, широко вживаний в області фінансів і бізнесу, де частіше зустрічаються задачі числового прогнозу. В результаті застосування цього методу, для навчальної вибірки даних створюється ієрархічна структура правил класифікації типу, "ЯКЩО... ТОДІ...", що мають вид дерева. Для того, щоб вирішити, до якого класу віднести деякий об'єкт або ситуацію, треба відповісти на питання, що стоїть у вузлах цього дерева, починаючи з його кореня. Питання можуть мати вигляд "Значення параметра А більше Х ? " або вигляду "Значення змінної В належить підмножині ознак З ? ". Якщо відповідь позитивна, перехід до правого вузла наступного рівня, якщо негативний - то до лівого вузла; потім знову відповідь на питання, пов'язане з відповідним вузлом. Таким чином, врешті-решт, можна дійти до одного з кінцевих вузлів, де визначений клас об'єкту. Цей метод гарантує предметне уявлення правил і його легко зрозуміти (рис. 1.8.).

Сьогодні спостерігається підйом інтересу до продуктів, що застосовують дерева рішень. В основному це пояснюється тим, що більшість комерційних проблем вирішуються ними швидше, ніж алгоритмами нейтронних мереж, вони простіше і зрозуміліше для користувачів. В той же час не можна сказати, що дерева рішень завжди діють безвідмовно: для певних типів даних вони можуть виявитися неприйнятними.

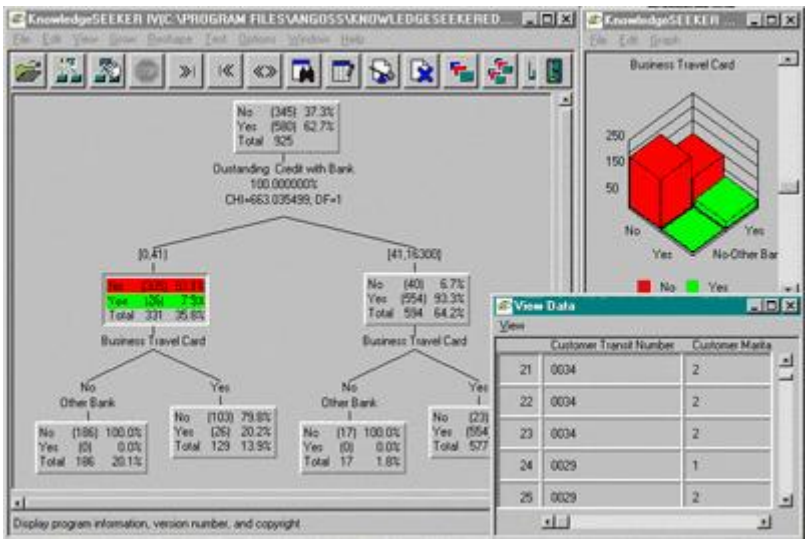


Рис.1.8. Система веде обробку банківської інформації.

Річ у тому, що окремим вузлам на кожній гілці відводиться менше число записів даних - дерево може сегментувати дані на велику кількість окремих випадків. Чим більше таких окремих випадків, тим менше навчальних прикладів потрапляє в кожен такий окремий випадок, і їх класифікація стає менш надійною. Якщо дерево дуже "гіллясте" - складається з невиправдано великого числа дрібних гілок - воно не даватиме статистично обґрунтованих відповідей. Як показує практика, у більшості систем, що використовують дерева рішень, ця проблема не знаходить задовільного рішення.

Системи роздумів на основі аналогічних випадків. Ідея алгоритму проста. Для того, щоб зробити прогноз майбутнього або вибрати правильне рішення, системи знаходять у минулому близькі аналоги наявної ситуації і вибирають ту ж відповідь, що була для них правильною. Тому, цей метод ще називають методом "найближчого сусіда". Системи роздумів на основі аналогічних випадків дають добрі результати в різних завданнях. Головний їх мінус в тому, що вони взагалі не створюють яких-небудь моделей або правил, узагальнювальних попередній досвід. У виборі рішення вони базуються на всьому масиві доступних історичних даних, тому неможливо сказати, на основі яких конкретно чинників ці системи будують свої відповіді.

Алгоритми виявлення асоціацій знаходять правила про окремі предмети, які з'являються разом в одній транзакції, наприклад в одній покупці. Послідовність - ця теж асоціація, але залежна від часу. Асоціація записується як А->Б, де А називається передумовою, Б - наслідком. Частота появи кожного окремого предмету або групи предметів, визначається дуже просто - підраховується кількість появи цього предмету у всіх подіях (покупках) і ділиться на загальну кількість подій. Ця величина вимірюється у відсотках і носить назву "поширеність". Низький рівень поширеності (менш одного тисячною відсотка) говорить про неістотність асоціації.

Для визначення важливості кожного отриманого асоціативного правила необхідно отримати величину, яка носить назву "довірчість А до Б" (взаємозв'язок А і Б). Ця величина показує, як часто з появою А з'являється Б і розраховується як відношення частоти появи (поширеності) А і Б разом до поширеності А. Тобто, якщо довірчість А до Б дорівнює 20%, то це означає, що при покупці товару А в кожному п'ятому випадку купують і товар Б. Якщо поширеність А не рівна поширеності Б, то і довірчість А до Б не дорівнює довірчості Б до А. Насправді, покупка комп'ютера частіше веде до покупки дискет, чим покупка дискети до покупки комп'ютера.

Ще однією важливою характеристикою асоціації є потужність асоціації. Чим більше потужність, то сильніше вплив, який поява А робить на появу Б. Потужність розраховується по формулі: (довірчість А до Б) / (поширеність Б).

Деякі алгоритми пошуку асоціацій спочатку сортують дані і лише після цього визначають взаємозв'язок і поширеність. Єдиною розбіжністю таких алгоритмів є швидкість або ефективність знаходження асоціацій. Це важливо, у зв'язку з величезною кількістю комбінацій, що необхідно перебрати для знаходження більш значущих правил. Алгоритми пошуку асоціацій можуть створювати свої бази даних поширеності, довірчості і потужності, до яких можна звертатися при запиті. Наприклад: "Знайти всі асоціації, в яких для товару Х довірчість більше 50% і поширеність не меншого 2,5%". При знаходженні послідовностей додається змінна часу, що дозволяє працювати з

серією подій для знаходження послідовних асоціацій впродовж деякого періоду часу (рис.1.9.).

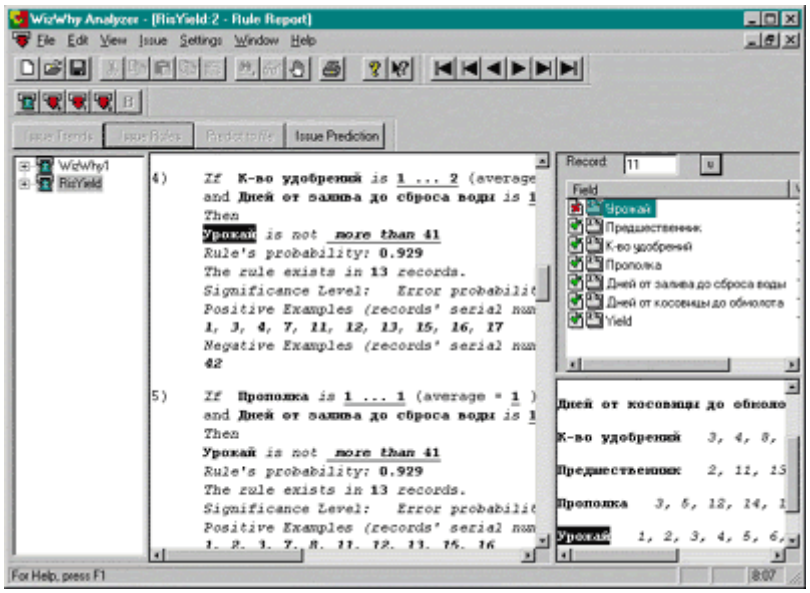


Рис.1.9. Система знайшла правила, які пояснюють низьку врожайність деяких сільськогосподарських культур.

Підводячи підсумки цьому методу аналізу, необхідно сказати, що випадково може виникнути така ситуація, коли товари в супермаркеті будуть згруповані за допомогою знайдених моделей, але це, замість очікуваного прибутку, дасть зворотний ефект. Це може відбутися через те, що клієнт довго не ходитиме по магазину у пошуках бажаного товару, купуючи при цьому ще щось, що попадається на очі, і те, що він ніколи не планував купувати.

Нечітка логіка застосовується для наборів даних, де приналежність даних до якої-небудь групи є вірогідністю в інтервалі від 0 до 1. Чітка логіка маніпулює результатами, які можуть бути або істиною, або ложью. Нечітка логіка застосовується в тих випадках, коли існує "може бути" в доповненні до "та" чи ні". Областю впровадження алгоритмів нечіткої логіки є будь-які аналітичні системи, зокрема :

- нелінійний контроль за процесами (виробництво);
- удосконалення стратегій управління і координації дій, наприклад складне промислове виробництво;
- самонавчальні системи (або класифікатори);

- дослідження ризикованих і критичних ситуацій;
- розпізнавання образів;
- фінансовий аналіз (ринки цінних паперів);
- дослідження даних (корпоративні сховища).

У Японії цей напрям переживає бум. Тут функціонує спеціально створена лабораторія Laboratory for International Fuzzy Engineering Research (LIFE). Програмою організації є створення ближчих до людини обчислювальних пристроїв. LIFE об'єднує 48 компаній в числі яких знаходяться: Hitachi, Mitsubishi, NEC, Sharp, Sony, Honda, Mazda, Toyota. З іноземних учасників LIFE можна виділити: IBM, Fuji Xerox, до діяльності LIFE також виявляє цікавість NASA [75].

Потужність і інтуїтивна простота нечіткої логіки як методології вирішення проблем гарантує її успішне використання у вбудованих системах контролю і аналізу інформації. При цьому відбувається підключення людської інтуїції і досвіду оператора. На відміну від традиційної математики, яка вимагає на кожному кроці моделювання точних і однозначних формулювань закономірностей, нечітка логіка пропонує зовсім інший рівень мислення, завдяки чому творчий процес моделювання відбувається на високому рівні абстракцій, при якому постулірується лише мінімальний набір закономірностей.

Недоліками нечітких систем є:

- відсутність стандартної методики конструювання нечітких систем;
- неможливість математичного аналізу нечітких систем існуючими методами.

Генетичні алгоритми є могутнім засобом рішення різних комбінаторних задач і проблем оптимізації. Проте, генетичні алгоритми увійшли зараз до стандартного інструментарію методів інтелектуальних обчислень. Цей метод названий так тому, що якоюсь мірою імітує процес природного відбору в природі. Хай нам треба знайти рішення задач, найбільш оптимальні з погляду деякого критерію, де кожне рішення цілком описується певним набором чисел або величин нечислової природи. Скажімо, якщо нам

треба вибрати сукупність фіксованого числа параметрів ринку, що істотно впливають на його динаміку, це буде набір імен цих параметрів. Про цей набір можна говорити як про сукупність хромосом, що визначають якості індивіда, - даного рішення поставленої задачі. Значення параметрів, що визначають рішення, називаються генами. Пошук оптимального рішення при цьому схожий на еволюцію популяції індивідів, представлених наборами хромосом.

У еволюції діють три механізми: по-перше, відбір найсильніших - наборів хромосом, яким відповідають найбільш оптимальні рішення; по-друге, схрещування - виробництво нових індивідів за допомогою змішування хромосомних наборів відібраних індивідів; і, по-третє, мутації - випадкові зміни генів у деяких індивідів популяції. В результаті зміни поколінь виробляється рішення поставленої задачі, яке вже не може бути далі покращене.

Генетичні алгоритми мають два слабкі місця. По-перше, постановка задачі не дає можливості проаналізувати статистичну значущість отримуваного з їх допомогою рішення і, по-друге, ефективно сформулювати завдання, визначити критерій відбору хромосом під силу тільки фахівцеві. Через ці чинники, генетичні алгоритми треба розглядати скоріше як інструмент наукового дослідження, чим засіб аналізу даних для практичного застосування в бізнесі і фінансах.

Еволюційне програмування наймолодша область інтелектуальних обчислень. Гіпотези про вид залежності цільової змінної від інших змінних формулюються системою у вигляді програм на деякій внутрішній мові програмування. Якщо це універсальна мова, то теоретично на ній можна виразити залежність будь-якого вигляду. Процес побудови таких програм будується як еволюція в світі програм (цим метод трохи схожий на генетичні алгоритми). Якщо система знаходить програму, яка точно виражає залежність, яка шукається, вона починає вносити до неї невеликі модифікації і відбирає серед побудованих таким чином дочірніх програм ті, які підвищують точність. Система "вирощує" декілька генетичних ліній програм, що конкурують між собою в точності знаходження шуканої залежності. Спеціальний транслюючий

модуль, перекладає знайдені залежності з внутрішньої мови системи на зрозумілий користувачеві мова (математичні формули, таблиці і ін.), роблячи їх досяжними. Для того, щоб зробити отримані результати зрозумілішими для користувача-нематематіка, існує великий арсенал різноманітних засобів візуалізації виявлених залежностей.

Пошук залежності цільових змінних від інших проводиться у формі функцій якого-небудь певного вигляду. Наприклад, в одному з найбільш вдалих алгоритмів цього типу - методі групового обліку аргументів (МГУА) залежність шукають у формі поліномів. Причому складні поліноми замінюються декількома простими, що враховують лише деякі ознаки (групи аргументів). Зазвичай використовуються попарні об'єднання ознак. Цей метод не має великих переваг в порівнянні з нейронними мережами з готовим набором стандартних нелінійних функцій, але, отримані формули залежності, в принципі, піддаються аналізу і інтерпретації (хоча на практиці це все-таки складно).

Програми візуалізації даних в певному значенні не є засобом аналізу інформації, оскільки вони тільки представляють її користувачеві. Проте, візуальне уявлення, скажімо, відразу чотирьох змінних наочно узагальнює величезні об'єми даних (рис. 1.10.).

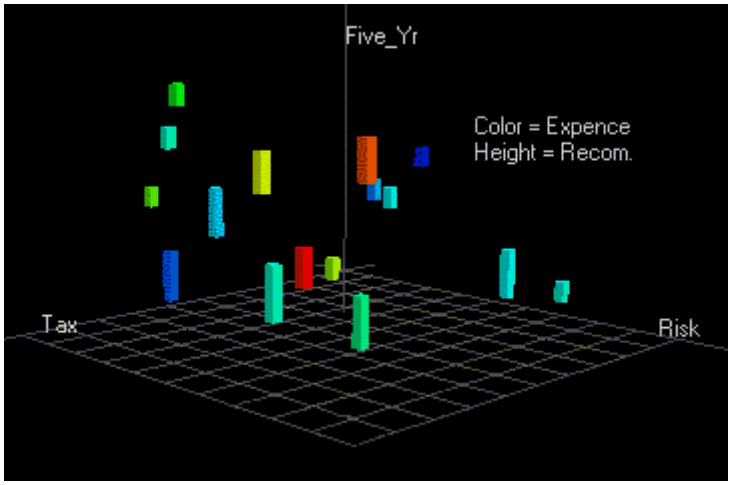


Рис. 1.10. Візуалізація показників діяльності інвестиційних фондів.

Комбіновані методи. Часто виробники об’єднують вказані підходи. Об'єднання алгоритмів нейронних мереж і технології дерев рішень сприяє побудові точнішої моделі і підвищенню швидкості. Для вирішення кожної проблеми слід шукати свій оптимальний метод.

1.5. Засоби програмної підтримки інтелектуального аналізу даних.

Сьогодні ми є свідками активного розвитку технологій інтелектуального аналізу даних, поява яких зв'язана, в першу чергу, з необхідністю аналітичної обробки надвеликих об'ємів інформації, що накопичується в сучасних сховищах даних. Можливість використання добре відомих методів математичної статистики і машинного навчання для вирішення завдань подібного роду відкрило нові можливості перед аналітиками, дослідниками, а також тими, хто ухвалює рішення - менеджерами і керівниками компаній. Аналітики передбачають, що з 2005 року такі технології стали дуже привабливою областю для IT-інвестицій. Згідно з результатами компанії Forrester Research, майже 30% європейських фірм розглядатимуть можливість впровадження цього програмного забезпечення в 2007 році. За даними іншої компанії, International Data Corporation (IDC), в країнах східної і південно-східної Азії (за винятком Японії) в 2005-2010 рр. середньорічний темп зростання рішень складе 21% [2, 17, 40, 61, 71].

На ринку програмного забезпечення Data Mining існує величезна різноманітність продуктів, що відносяться до цієї категорії. І не розгубитися в них достатньо складно. Для вибору продукту слід ретельно вивчити поставлені задачі і позначити ті результати, які необхідно отримати.

На початку 90-х років минулого сторіччя ринок Data Mining налічував близько десяти постачальників. У середині 90-х число постачальників, представлених компаніями малого, середнього і великого розміру, налічувало більше 50 фірм. Зараз до аналітичних технологій, зокрема до Data Mining,

виявляється величезний інтерес. На цьому ринку працює безліч фірм, орієнтованих на створення інструментів Data Mining, а також комплексного впровадження Data Mining, OLAP і сховищ даних. Інструменти Data Mining у багатьох випадках розглядаються як складова частина BI-платформ, до складу яких також входять засоби побудови сховищ і вітрин даних, засоби обробки несподіваних запитів (ad-hoc query), засоби звітності (reporting), а також інструменти OLAP. Розробкою в секторі Data Mining всесвітнього ринку програмного забезпечення зайняті як всесвітньо відомі лідери (IBM, Microsoft), так і нові компанії, що розвиваються.

Інструменти Data Mining можуть бути представлені або як самостійне рішення, або як доповнення до основного продукту. Останній варіант реалізується багатьма лідерами ринку програмного забезпечення. Так, вже стало традицією, що розробники універсальних статистичних пакетів, на додаток до традиційних методів статистичного аналізу, включають в пакет певний набір методів Data Mining. Це такі пакети як SPSS (SPSS, Clementine), Statistica (Statsoft), SAS Institute (SAS Enterprise Miner). Деякі розробники OLAP-рішень також пропонують набір методів Data Mining, наприклад, сімейство продуктів Cognos. Є постачальники, які включають Data Mining рішення у функціональність СУБД: це Microsoft (Microsoft SQL Server), Oracle, IBM (IBM Intelligent Miner for Data) [70, 73, 74, 76, 77].

Інструменти Data Mining можна оцінювати по різних критеріях. Оцінка програмних засобів Data Mining з погляду кінцевого користувача визначається шляхом оцінки набору його характеристик. Їх можна поділити на дві групи: бізнес-характеристики і технічні характеристики. Цей розподіл є достатньо умовним, і деякі характеристики можуть потрапляти одночасно в обидві категорії.

Інтуїтивний інтерфейс. Інтерфейс - середовище передачі інформації між програмним середовищем і користувачем, діалогова система, яка дозволяє передати людині всі необхідні дані, отримані на етапі формалізації і обчислення. Він припускає розташування різних елементів, в т.ч. блоків меню,

інформаційних полів, графічних блоків, блоків форм, на екранних формах. Для зручності роботи користувача необхідно, щоб інтерфейс був інтуїтивним.

Інтуїтивний інтерфейс дозволяє користувачеві легко і швидко сприймати елементи інтерфейсу, завдяки чому діалог "програмне середовище-користувач" простіше і доступніше. Поняття інтуїтивного інтерфейсу включає також поняття знайомого навколишнього середовища і наявність виразної нетехнічної термінології (наприклад, для повідомлення користувача про скоєну помилку).

Зручність експорту - імпорту даних. При роботі з інструментом Data Mining користувач часто застосовує різноманітні набори даних, працює з різними джерелами даних. Це можуть бути текстові файли, файли електронних таблиць, файли баз даних. Інструмент Data Mining повинен мати зручний спосіб завантаження (імпорту) даних. Після закінчення роботи користувач також повинен мати зручний спосіб вивантаження (експорту) даних в зручне для нього середовище. Програма повинна підтримувати найбільш поширені формати даних. Додаткова зручність для користувача створюється при нагоді завантаження і вивантаження певної частини (по вибору користувача) полів, що імпортуються або експортуються.

Наочність і різноманітність отримуваної звітності. Ця характеристика припускає отримання звітності в термінах предметної області, а також в якісно спроектованих вихідних формах в тій кількості, яка може надати користувачеві всю необхідну результативну інформацію.

Зручність і простота використання. Істотно полегшує роботу користувача можливість використовувати програми Майстер.

Можливості візуалізації. Наявність графічного представлення інформації істотно полегшує інтерпритованість отриманих результатів.

Наявність значень параметрів, заданих за умовчанням. Для користувачів, що починають, - це достатньо істотна характеристика, оскільки при виконанні багатьох алгоритмів від користувача потрібне завдання або вибір великого числа параметрів. Особливо багато їх в інструментах, що

реалізовують метод нейронних мереж. У нейросимуляторах найчастіше наперед задані значення основних параметрів, інший раз недосвідченим користувачам навіть не рекомендується змінювати ці значення. Якщо ж такі значення відсутні, користувачеві доводиться перепробувати безліч варіантів, перш ніж отримати прийнятний результат.

Кількість методів і алгоритмів. У багатьох інструментах Data Mining реалізоване відразу декілька методів, що дозволяють вирішувати одну або декілька задач. Якщо для вирішення одного завдання (класифікації) передбачена можливість використання декількох методів (дерев рішень і нейронних мереж), користувач дістає можливість порівнювати характеристики моделей, побудованих за допомогою цих методів.

Можливості пошуку, сортування, фільтрації. Така можливість корисна як для вхідних даних, так і для вихідної інформації. Застосовується сортування по різних критеріях (полям), з можливістю накладення умов. За умови фільтрації вхідних даних з'являється можливість побудови моделі Data Mining на одній з вибірок набору даних. Фільтрація вихідної інформації корисна з погляду інтерпретації результатів. Так, наприклад, іноді при побудові дерев рішень результати виходять дуже громіздкими, і тут можуть виявитися корисними функція як фільтрації, так і пошуку і сортування. Додаткова зручність для користувача - колірне підсвічування деяких категорій записів.

Захист, пароль. Дуже часто за допомогою Data Mining аналізується конфіденційна інформація, тому наявність пароля доступу в систему є бажаною характеристикою для інструменту.

Описані характеристики є критеріями функціональності, зручності, безпеки інструменту Data Mining. При виборі інструменту слід керуватися потребами, а також задачами, які необхідно вирішити. Так, наприклад, якщо точно відомо, що фірмі необхідно вирішувати виключно задачі класифікації, то можливість рішення інструментом інших завдань зовсім не є критичною. Проте, слід враховувати, що впровадження Data Mining при серйозному підході

вимагає серйозних фінансових вкладень, тому необхідно враховувати всі можливі задачі, які можуть виникнути в перспективі.

Ринок інструментів Data Mining визначається широтою цієї технології і внаслідок цього - величезним різноманіттям програмного забезпечення. Найбільш популярна група інструментів містить наступні категорії:

- набори інструментів;
- класифікація даних;
- кластеризація і сегментація;
- інструменти статистичного аналізу;
- аналіз текстів (Text Mining), витягання відхилень (Information Retrieval);
- інструменти візуалізації.

Набори інструментів. До цієї категорії відносяться універсальні інструменти, які включають методи класифікації, кластеризації і попередньої підготовки даних. До цієї групи відносяться такі відомі комерційні інструменти як:

- Clementine. Data Mining з використанням Clementine є бізнес-процесом, розробленим для мінімізації часу рішення задач. Clementine підтримує процес Data Mining: доступ до даним, перетворення, моделювання, оцінювання і впровадження. За допомогою Clementine Data Mining виконується з методологією CRISP-DM.
- IBM Intelligent Miner for Data. Інструмент пропонує останні Data Mining-методи, підтримує повний Data Mining процес: від підготовки даних до презентації результатів. Підтримка мов XML і PMML.
- KXEN (Knowledge extraction Engines). Інструмент, що працює на основі теорії Вапника SVM. Вирішує задачі підготовки даних, сегментації, тимчасових рядів і SVM-класифікації.
- Oracle Data Mining (ODM). Інструмент забезпечує GUI, PL/SQL-інтерфейси, Java-інтерфейс. Використовувані методи: байесовська класифікація, алгоритми пошуку асоціативних правил, кластерні методи, SVM та інші.

- Polyanalyst. Набір, що забезпечує вселякий Data Mining, також включає аналіз текстів, ліс рішень, аналіз зв'язків. Підтримує OLE DB for Data Mining і DCOM-технологію.
- SPSS. Один з найбільш популярних інструментів, підтримується безліч методів Data Mining.
- Statistica Data Miner. Інструмент забезпечує вселякий, інтегрований статистичний аналіз даних, має могутні графічні можливості, управління базами даних, а також додатки розробки систем.

Друга група задач представлена інструментами, що реалізовують наступні рішення:

- Magnum Opus є швидким інструментом пошуку асоціативних правил в даних, підтримується операційними системами Windows, Linux і Solaris.
- Nuggets - це набір, що включає пошук асоціативних правил і інші алгоритми.
- Artool, інструмент містить набір алгоритмів для пошуку асоціативних правил в бінарних базах даних (binary databases).
- DM-II System, інструмент включає алгоритм CBA для виконання класифікації на основі асоціативних правил і деяких інших характеристик.

Для розв’язання задач *кластеризації та сегментації* широке застосування отримали:

- ClustanGraphics 3 - ієрархічний кластерний аналіз «зверху вниз», в якому підтримуються могутні графічні можливості.
- Starprobe, заснований на Web кросс-платформенній системі, включає методи кластеризації, нейронні мережі, дерева рішень, візуалізацію і т.д.
- Visipoint - кластеризація методом карт Кохонена (Self-organizing Map clustering) і візуалізація.
- Autoclass C - "навчання без вчителя" за допомогою байесовських мереж від NASA, працює на основі операційних систем Unix і Windows.
- Reckless є набором кластерних алгоритмів, заснованих на концепції к-ближніх сусідів. Інструмент перед проведенням кластеризації виконує пошук і

ідентифікацію шумів і викидів для зменшення їх впливу на результати кластеризації.

Для вирішення задач оцінювання і прогнозування застосовуються:

- Alyuda Forecaster XL - інструмент реалізований у вигляді Excel-надбудови і призначений для вирішення задач прогнозування і оцінювання з використанням нейронних мереж.
- Excelneuralpackage, в якому реалізовано дві базові парадигми нейронних мереж - багатопрошарковий персептрон і мережі Кохонена.

Як ми бачимо, ринок програмного забезпечення Data Mining представлений безліччю інструментів і на ньому йде постійна конкурентна боротьба за споживача. Така конкуренція породжує нові якісні рішення. Все більше число постачальників прагнуть об'єднати в своїх інструментах як можна більше число сучасних методів і технологій. Data Mining-інструменти найчастіше розглядаються як складова частина ринку Business Intelligence, який, не дивлячись на деякий загальний спад в індустрії інформаційних технологій, упевнено і постійно розвивається. В той же час деякі фахівці відзначають відставання існуючого програмного забезпечення від теоретичних розробок у зв'язку з складністю програмної реалізації деяких нових теоретичних розробок методів і алгоритмів Data Mining. В цілому, можна резюмувати, що ринок Business Intelligence, зокрема ринок інструментів Data Mining, настільки широкий і різноманітний, що будь-яка компанія може вибрати для себе інструмент, який підійде їй по функціональності і по можливостях бюджету.

Більш докладніше розглянемо деякі комп’ютерні системи Data Mining, які знайшли значне поширення в практиці діяльності підприємств та організацій.

Програмний продукт SAS Enterprise Miner (розробник SAS Institute Inc.,) є інтегрованою компонентою системи SAS, створеною спеціально для виявлення у величезних масивах даних інформації, яка необхідна для ухвалення рішень. Розроблений для пошуку і аналізу глибоко прихованих

закономірностей в даних SAS, Enterprise Miner включає методи статистичного аналізу, відповідну методологію виконання проектів Data Mining (SEMMA) і графічний інтерфейс користувача. Важливою особливістю SAS Enterprise Miner є його повна інтеграція з програмним продуктом SAS Warehouse Administrator, призначеним для розробки і експлуатації інформаційних сховищ, і іншими компонентами системи SAS. Розробка проектів Data Mining може виконуватися як локально, так і в архітектурі клієнт-сервер.

Пакет підтримує виконання всіх необхідних процедур в рамках єдиного інтегрованого рішення з можливостями колективної роботи і поставляється як розподілений клієнт-серверний додаток, що особливо зручно для здійснення аналізу даних в масштабах крупних організацій. Пакет SAS Enterprise Miner призначений для фахівців з аналізу даних, маркетингових аналітиків, маркетологів, фахівців з аналізу ризиків, фахівців з виявлення шахрайських дій, а також інженерів і учених, відповідальних за ухвалення ключових рішень в бізнесі або дослідницькій діяльності (рис. 1.11).

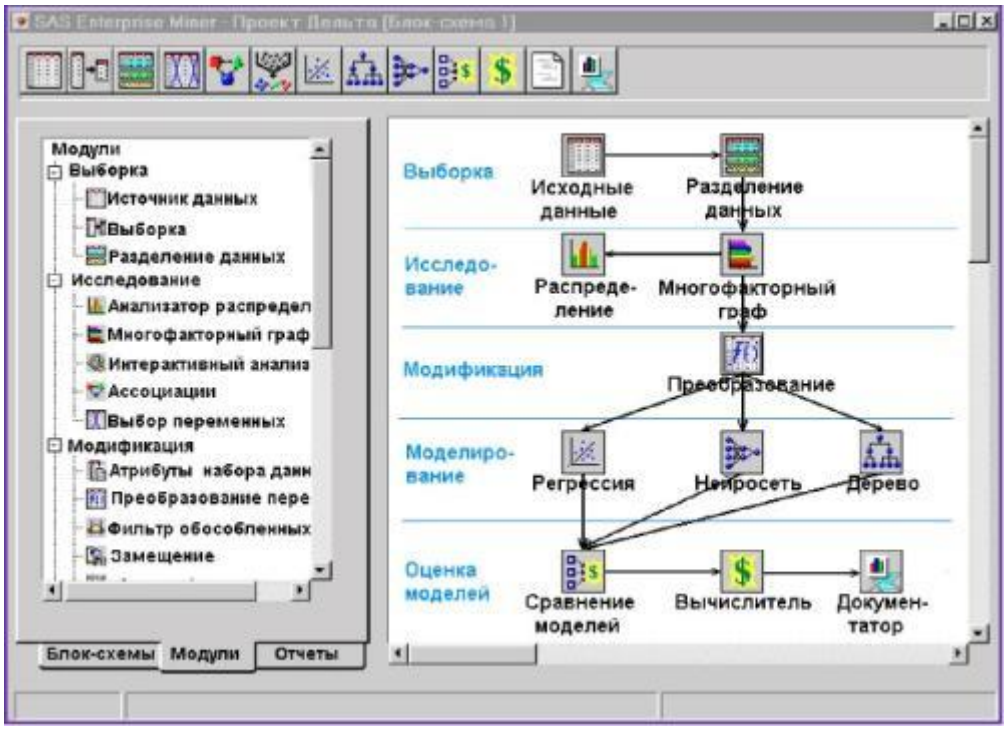


Рис. 1.11. Головне вікно SAS Enterprise Miner.

У пакеті SAS Enterprise Miner реалізований підхід, що заснований на створенні діаграм процесів обробки даних і дозволяє усунути необхідність ручного кодування і прискорити розробку моделей завдяки методиці Data Mining SEMMA. Графічний інтерфейс пакету є інтерфейсом типу "вказати і клацнути". З його допомогою користувачі можуть виконати всі стадії процесу Data Mining від вибору джерел даних, їх дослідження і модифікації до моделювання і оцінки якості моделей з подальшим застосуванням отриманих моделей, як для обробки нових даних, так і для підтримки процесів ухвалення рішень.

Пакет SAS Enterprise Miner пропонує різні інструменти для здійснення підготовки даних, які дають можливість, наприклад, зробити вибірку або розбиття даних, здійснити вставку неотриманих значень, провести кластеризацію, об'єднати джерела даних, усунути зайві змінні, виконати обробку на мові SAS за допомогою спеціалізованого вузла SAS code, здійснити перетворення змінних і фільтрацію недостовірних даних. Пакет оснащений функціями описової статистики, а також розширеними засобами візуалізація, яка дозволяє досліджувати надвеликі об'єми даних, представлених у вигляді багатовимірних графіків, і проводити графічне порівняння результатів моделювання. Також він надає набір інструментів і алгоритмів прогностичного і описового моделювання, що включають дерева рішень, нейронні мережі, нейронні мережі, що самоорганізовуються, методи міркування, засновані на механізмах пошуку в пам'яті (memorybased reasoning), лінійну і логістична регресії, кластеризацію, асоціації, тимчасові ряди і багато що інше. Інтеграція різних моделей і алгоритмів в пакеті Enterprise Miner дозволяє проводити послідовне порівняння моделей, створених на основі різних методів, і залишатися при цьому в рамках єдиного графічного інтерфейсу. Вбудовані засоби оцінки формують єдине середовище для порівняння різних методів моделювання, як з погляду статистики, так і з погляду бізнесу, дозволяючи виявити найбільш відповідні методи для наявних даних. Результатом є якісний

аналіз даних, виконаний з урахуванням специфічних проблем конкретного бізнесу.

Отримані моделі можна публікувати для сумісного використання в рамках підприємства за допомогою репозітарію моделей, що є першою на ринку системою управління моделями. Пакет надає ряд вбудованих оціночних функцій, що дозволяють порівняти результати різних методів моделювання, як в термінах бізнесу, так і з використанням статистичної діагностики. Це дає можливість вимірювати ефективність моделі в термінах її прибутковості. Підсумком робіт по інтелектуальному аналізу даних є розгортання створеної моделі - це завершальна стадія, на якій реалізується економічна віддача від проведених досліджень. Процес застосування моделі до нових даних, відомий як скорінг, часто вимагає ручного написання або перетворення програмного коду. Пакет SAS Enterprise Miner автоматизує процес підбору коефіцієнтів і надає готовий програмний код для скорінга на всіх стадіях створення моделі, підтримує створення різних програмних середовищ для розгортання моделі на мовах SAS, C, Java і PMML.

Система Polyanalyst призначена для автоматичного і напівавтоматичного аналізу числових баз даних і витягання з сирих даних практично корисних знань. Polyanalyst знаходить багатофакторні залежності між змінними в базі даних, автоматично будує і тестує багатомірні нелінійні моделі, що виражають знайдені залежності, виводить класифікаційні правила по навчальних прикладах, знаходить в даних багатомірні кластери, будує алгоритми рішень. Розробник системи Polyanalyst - російська компанія Megaputer Intelligence ("Мегапьютер"). За своєю природою Polyanalyst є клієнт-серверним додатком. Користувач працює з клієнтською програмою Polyanalyst Workplace. Математичні модулі виділені в серверну частину - Polyanalyst Knowledge Server. Така архітектура надає природну можливість для масштабування системи (рис. 1.12.).

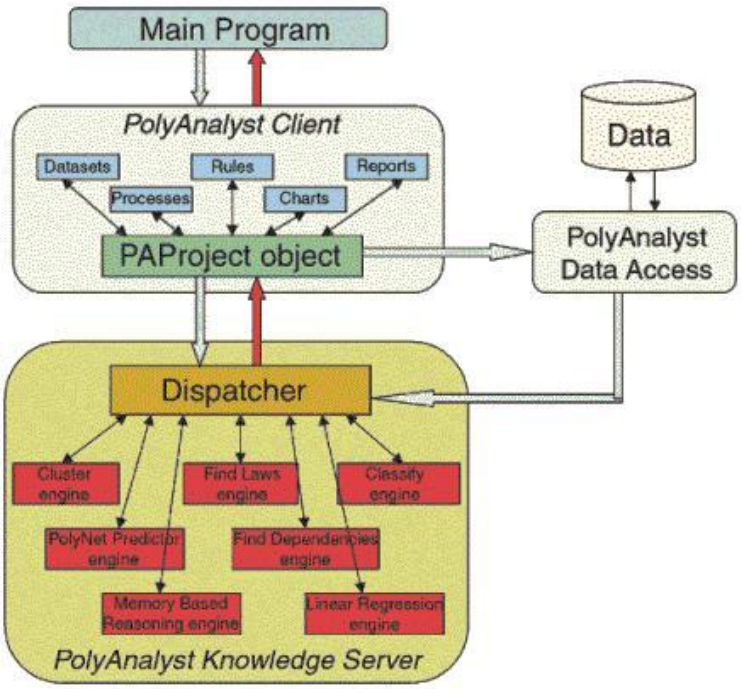


Рис. 1.12. Архітектура системи Polyanalyst.

Математичні модулі (Exploration Engines) і багато інших компонентів Polyanalyst виділені в окремі динамічні бібліотеки і доступні з інших додатків. Це дає можливість інтегрувати математику Polyanalyst в ті програмні ресурси, що існують в інформаційних системах, наприклад, в CRM- або ERP- системах.

Основні риси, призначеного для користувача, інтерфейсу програми: розвинені можливості маніпулювання з даними, графіка для представлення даних і візуалізації результатів, майстри створення об'єктів, сквозний логічний зв'язок між об'єктами, мова символьних правил, інтуїтивне управління через drop-down і pop-up меню, докладна контекстна довідка. Одиницею Data Mining дослідження в Polyanalyst є "проект". Проект об'єднує в собі всі об'єкти дослідження, дерево проекту, графіки, правила, звіти і т.д. Проект зберігається у файлі внутрішнього формату системи. Звіти досліджень представляються у форматі HTML і доступні через Інтернет (рис.1.13.).

Пакет Polyanalyst включає 18 математичних модулів, заснованих на різних алгоритмах Data і Text Mining. Більшість з цих алгоритмів є Know-How компанії Мегапьютер і не мають аналогів в інших системах. До них

відносяться: моделювання, прогнозування, кластеризація, класифікація, текстовий аналіз.

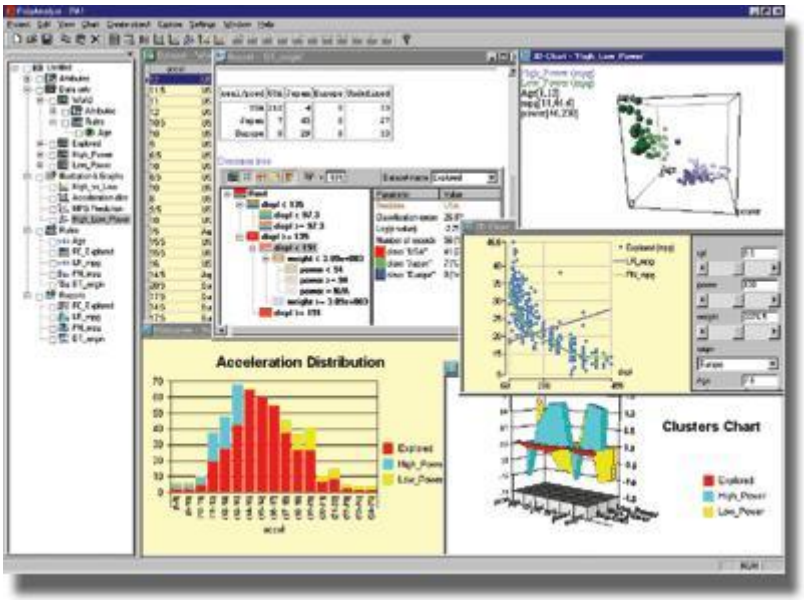


Рис.1.13. Головне меню пакету Polyanalyst.

Програмні продукти Cognos (розробник - компанія Cognos) - це інструменти інтелектуального або ділового аналізу даних (Business Intelligence Tools) або BI-інструменти. За їх допомогою вирішуються слідуючи задачі:

Робота із запитами і звітами. Рішення в області роботи із звітами орієнтовані на різні типи користувачів. Продукти відрізняються вимогами до рівня складності звітів і рівня навиків кінцевих користувачів:

- Decision Stream - засіб для створення вітрин даних (Data marts), оптимізованих на формування запитів і побудову звітів;
- Impromptu - засіб для роботи із запитами, а також із статичними звітами, що настроюються;
- Powerplay - засіб побудови багатомірних звітів;
- Impromptu Web Reports - засоби для роботи із статичними звітами через Web;
- Cognos Query - засіб для створення запитів, навігації і дослідження даних в т.ч. через Web;

- Visualizer - засіб для роботи з могутніми візуальними звітами.

Аналіз даних. Засоби аналізу даних призначені для аналізу критичної інформації і виявлення значущих чинників. Цей процес охоплює повний набір аналітичних задач і завдань по побудові звітів, включаючи роботу із звітами бізнес-уровня, можливість переходу до даним нижнього рівня, створення і проглядання уявлень з метою виявлення пріоритетів. Інтеграція засобів дозволяє зручно переходити від дослідження і аналізу даних за допомогою звітів бізнес-уровня до дослідження і аналізу даних по звітах нижнього рівня:

- Powerplay - засіб багатовимірного (OLAP) аналізу і побудови бізнес-звітів;
- Impromptu - засіб для проглядання звітів з детальною інформацією нижнього рівня (для Windows);
- Impromptu Web Reports - засіб для проглядання звітів з детальною інформацією нижнього рівня (для Web);
- Visualizer - засіб візуального представлення даних.

Візуалізація і виявлення пріоритетів. До розділу візуалізації інформації і виявлення пріоритетів можна віднести цілий спектр продуктів. З їх допомогою користувачеві стає доступна візуалізована інформація, представлена в зручному вигляді для виявлення критичних чинників на великих масивах даних. У цих продуктах за основу береться можливість аналізу ключових чинників, що впливають на дану галузь знань (бізнесу) за допомогою широких можливостей по візуалізації даних. Правильно виявлені пріоритети є основою для ухвалення ефективних рішень:

- Visualizer - засіб для представлення інформації у формі візуальних вистав з використанням візуальних елементів для виявлення пріоритетів;
- PowerPlay - засіб багатовимірного представлення інформації;
- Impromptu - засіб для роботи із звітами, що будуються;
- Cognos Query - засіб Web-користувачів для побудови запитів.

Розвідка даних (data mining). Засоби розвідки і добування даних пропонують цілий ряд можливостей по автоматизованому перегляду даних,

дозволяючи розкривати приховані тенденції, виявляти пріоритетні рішення і дії шляхом відображення тих чинників, які більш за інших впливають на досліджувані показники:

- Scenario - засіб сегментації і класифікації;
- 4Thought - засіб прогнозування (рис. 1.14.);
- Visualazer як засіб візуалізації.

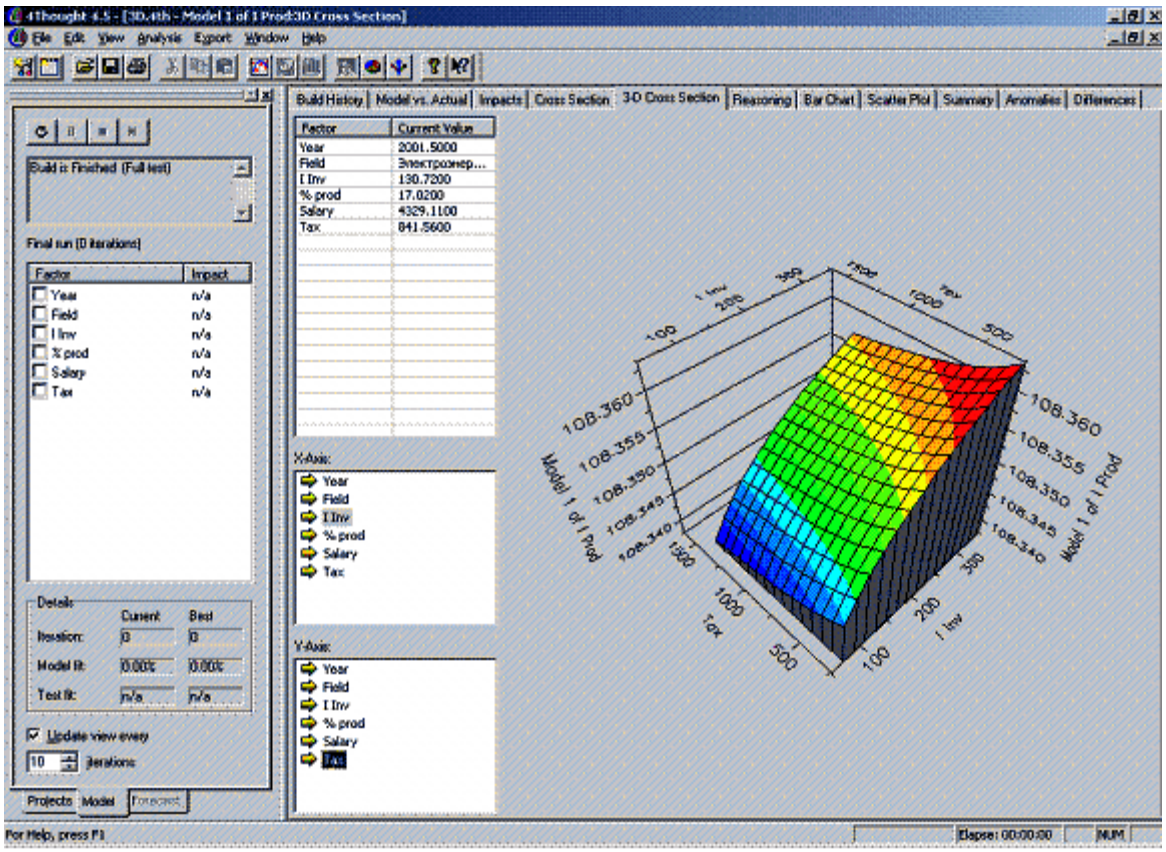


Рис. 1.14.**.** Інтерфейс програмного комплексу Cognos 4Thought.

Захист інформації. Захист інформації досягається за рахунок використання єдиного для всіх застосувань компонента Access Manager, що дозволяє описувати класи користувачів і управляти ними для всіх типів аналітичних додатків Cognos. На додаток до Access Manager, можуть бути використані також звичайні можливості забезпечення безпеки на рівні бази даних і операційної системи. На практиці можливе одночасне використання всіх трьох рівнів захисту інформації.

Опис метаданих. Як засіб опису метаданих може бути використаний єдиний для всіх Cognos BI продуктів компонент Cognos Architect. Привабливість використання єдиного для всіх засобів модуля полягає в можливості одноманітного представлення бізнес-інформації. Одного разу сформульовані метадані стають доступними в будь-якому аналітичному застосуванні Cognos.

Система STATISTICA Data Miner (розробник - компанія StatSoft) спроектована і реалізована як універсальний і всесторонній засіб аналізу даних - від взаємодії з різними базами даних до створення готових звітів, що реалізовують так званий графічно-орієнтований підхід. Серцем STATISTICA Data Miner є браузер процедур Data Mining, який містить більше 300 основних процедур, спеціально оптимізованих під задачі Data Mining, засоби логічного зв'язку між ними і управління потоками даних, що дозволить конструювати власні аналітичні методи (рис. 1.15.).

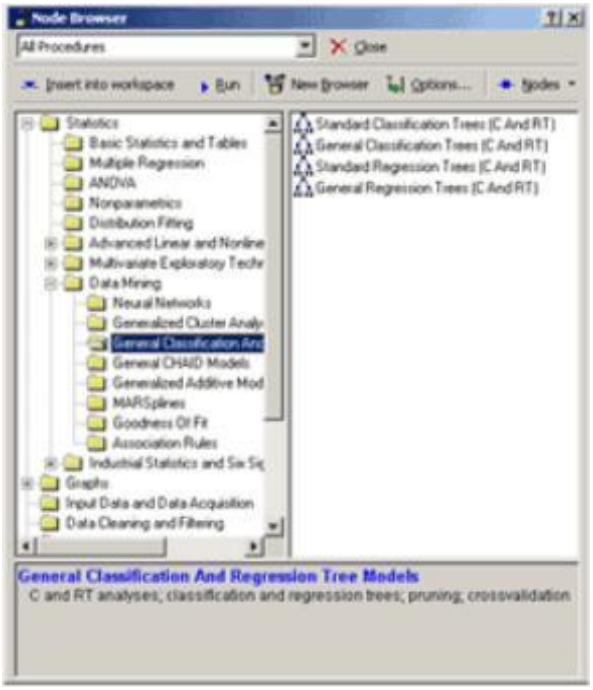


Рис. 1.15. Браузер процедур Data Mining.

Робочий простір STATISTICA Data Miner складається з чотирьох основних частин:

- Data Acquisition - збір даних. У даній частині користувач ідентифікує джерело даних для аналізу, будь то файл даних або запит з бази даних.
- Data Preparation, Cleaning, Transformation - підготовка, перетворення і очищення даних. Тут дані перетворяться, фільтруються, групуються.
- Data Analysis, Modeling, Classification, Forecasting - аналіз даних, моделювання, класифікація, прогнозування. Користувач може за допомогою браузеру або готових моделей задати необхідні види аналізу даних, таких як прогнозування, класифікація, моделювання.
- Reports - результати. У даній частині користувач може проглянути, задати вигляд і побудувати результати аналізу (наприклад, робоча книга, звіт або електронна таблиця).

Засоби аналізу STATISTICA Data Miner можна розділити на п'ять основних класів:

- General Slicer/Dicer and Drill-Down Explorer – розмітка - розподіл і поглиблений аналіз. Набор процедур, що дозволяє розбивати, групувати змінні, обчислювати описові статистики, будувати дослідницькі графіки.
- General Classifier - класифікація. STATISTICA Data Miner включає повний пакет процедур класифікації: узагальнені лінійні моделі, дерева класифікації, регресійні дерева, кластерний аналіз.
- General Modeler/Multivariate Explorer - узагальнені лінійні, нелінійні і регресійні моделі. Даний елемент містить лінійні, нелінійні, узагальнені регресійні моделі і елементи аналізу дерев класифікації.
- General Forecaster - прогнозування. Включає моделі АРПСС, сезонні моделі АРПСС, експоненціальне згладжування, спектральний аналіз Фур'є, сезонна декомпозиція, прогнозування за допомогою нейронних мереж.
- General Neural Networks Explorer - нейромережевий аналіз. У даній частині міститься якнайповніший пакет процедур нейромережевого аналізу.

Приведені вище елементи є комбінацією модулів інших продуктів StatSoft. Окрім них, STATISTICA Data Miner містить набір спеціалізованих процедур Data Mining, які доповнюють лінійку інструментів Data Mining:

- Feature Selection and Variable Filtering (for very large data sets) - спеціальна вибірка і фільтрація даних (для великих об'ємів даних). Даний модуль автоматично вибирає підмножини змінних із заданого файлу даних для подальшого аналізу.
- Association Rules - правила асоціації. Модуль є реалізацією так званого апріорного алгоритму виявлення правил асоціації. Наприклад, результат роботи цього алгоритму міг би бути наступним: клієнт після покупки продукт "A", в 95 випадках з 100 протягом наступних двох тижнів після цього замовляє продукт "B" або "C".
- Interactive Drill-Down Explorer - інтерактивний поглиблений аналіз. Є набором засобів для гнучкого дослідження великих наборів даних. На першому кроці ви задаєте набір змінних для поглибленого аналізу даних, на кожному подальшому кроці вибираєте необхідну підгрупу даних для подальшого аналізу.
- Generalized EM & к-Means Cluster Analysis - узагальнений метод максимуму середнього і кластеризація методом середніх. Даний модуль - це розширення методів кластерного аналізу. Він призначений для обробки великих наборів даних і дозволяє кластеризувати як безперервні, так і категоріальні змінні, забезпечує всі необхідні функціональні можливості для розпізнавання образів.
- Generalized Additive Models (GAM) - узагальнені аддитивні моделі (GAM). Набор методів, розроблених і популяризованных Hastie і Tibshirani.
- General Classification and Regression Trees (GTrees) - узагальнені класифікаційні і регресійні дерева (GTrees). Модуль є повною реалізацією методів, розроблених Breiman, Friedman, Olshen і Stone (1984). Окрім цього, модуль містить різного роду доопрацювання і доповнення, такі як

оптимізації алгоритмів для великих об'ємів даних. Модуль є набором методів узагальненої класифікації і регресійних дерев.

- General CHAID (Chi-square Automatic Interaction Detection) Models - узагальнені CHAID-модели (Хі-квадрат автоматичне виявлення взаємодії). Подібно до попереднього елементу, цей модуль є оптимізацією даної математичної моделі для великих об'ємів даних.
- Boosted Trees - розширювані прості дерева. Останні дослідження аналітичних алгоритмів показують, що для деяких завдань побудови "складних" оцінок, прогнозів і класифікацій використання послідовно збільшуваних простих дерев дає точніші результати, ніж нейронні мережі або складні цілісні дерева. Даний модуль реалізує алгоритм побудови простих збільшуваних (розширюваних) дерев.
- Multivariate Adaptive Regression Splines (Mar Splines) - багатовимірні адаптивні регресійні сплайни (Mar Splines). Даний модуль заснований на реалізації методики запропонованої Friedman (1991).

Нескладно відмітити, що система STATISTICA включає величезний набір різних аналітичних процедур, і це робить її недоступною для звичайних користувачів, які слабо знаються на методах аналізу даних. Компанією StatSoft запропонований варіант роботи для звичайних користувачів, що володіють невеликими досвідом і знаннями в аналізі даних і математичній статистиці. Для цього, окрім загальних методів аналізу, були вбудовані готові закінчені модулі аналізу даних, призначені для вирішення найбільш важливих і популярних завдань: прогнозування, класифікації, створення правил асоціації і т. д.

Oracle Data Mining. У березні 1998 компанія Oracle оголосила про спільну діяльність з 7 партнерами, які є постачальниками інструментів Data Mining. Далі послідувало включення в Oracle8i засобів підтримки алгоритмів Data mining. У червні 1999 року Oracle купує Darwin (Thinking Machines Corp.) і у 2000-2001 році виходять нові версії Darwin, Oracle Data Mining Suite. У червні 2001 року виходить Oracle9i Data Mining.

Oracle Data Mining є опцією або модулем в Oracle Enterprise Edition. Опція Oracle Data Mining (ODM) призначена для аналізу даних методами, що відносяться до технології витягання знань, або Data Mining. ODM підтримує всі етапи технології витягання знань, включаючи постановку задачі, підготовку даних, автоматичну побудову моделей, аналіз і тестування результатів, використання моделей в реальних застосуваннях. Важливо, що моделі будуються автоматично на основі аналізу наявних даних про об'єкти, спостереження і ситуації за допомогою спеціальних алгоритмів. Основу опції ODM складають процедури, що реалізовують різні алгоритми побудови моделей класифікації, регресії, кластеризації. На етапі підготовки даних забезпечується доступ до будь-яких реляційних баз, текстових файлів, файлів формату SAS. Додаткові засоби перетворення і очищення даних дозволяють змінювати вигляд сценарію, проводити нормалізацію значень, виявляти невизначені або відсутні значення. На основі підготовлених даних спеціальні процедури автоматично будують моделі для подальшого прогнозування, класифікації нових ситуацій, виявлення аналогій. ODM підтримує побудову п'яти різних типів моделей. Графічні засоби надають широкі можливості для аналізу отриманих результатів, верифікації моделей на тестових наборах даних, оцінки точності і стійкості результатів. Уточнені і перевірені моделі можна включати в існуючі додатки шляхом генерації їх описів на C++, Java, а також розробляти нові спеціалізовані застосування за допомогою того, що входить до складу середовища ODM засобу розробки Software Development Kit (SDK).

Аналітична платформа Deductor. Склад і призначення аналітичної платформи Deductor (розробник - компанія BaseGroup Labs). Deductor складається з двох компонентів: аналітичного додатка Deductor Studio і багатомірного сховища даних Deductor Warehouse. Архітектура системи Deductor представлена на рис. 1.16.

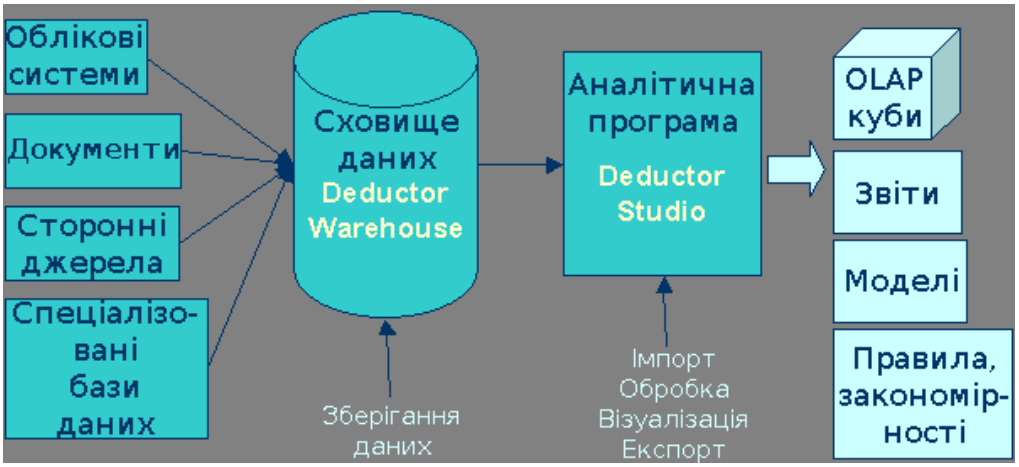


Рис. 1.16. Архітектура системи Deductor.

Deductor Warehouse - багатомірне сховище даних, що акумулює всю необхідну для аналізу наочної області інформацію. Використання єдиного сховища дозволяє забезпечити несуперечність даних, їх централізоване зберігання і автоматично створює всю необхідну підтримку процесу аналізу даних. Deductor Warehouse оптимізований для вирішення саме аналітичних задач, що позитивно позначається на швидкості доступу до даних.

Deductor Studio - це програма, призначена для аналізу інформації з різних джерел даних. Вона реалізує функції імпорту, обробки, візуалізації і експорту даних. Deductor Studio може функціонувати і без сховища даних, отримуючи інформацію з будь-яких інших джерел, але найбільш оптимальним є їх спільне використання.

Розглянемо цей процес детальніше. На початковому етапі в програму завантажуються або імпортуються дані з якого-небудь довільного джерела. Сховище даних Deductor Warehouse є одним з джерел даних. Підтримуються також інші сторонні джерела. Зазвичай в програму завантажуються не всі дані, а якась вибірка, необхідна для подальшого аналізу. Після здобуття вибірки можна отримати детальну статистику по ній, проглянути, як виглядають дані на діаграмах і гістограмах. Такий розвідувальний аналіз дає можливість приймати рішення про необхідність передобробки даних. Наприклад, якщо статистика показує, що у вибірці є порожні значення (пропуски даних), можна застосувати

фільтрацію для їх усунення. Передоброблені дані далі піддаються трансформації. Наприклад, нечислові дані перетворяться в числові, що необхідне для деяких алгоритмів. Безперервні дані можуть бути розбиті на інтервали, тобто виробляється їх дискретизація. До трансформованих даних застосовуються методи глибшого аналізу. На цьому етапі виявляються приховані залежності і закономірності в даних, на підставі яких будуються різні моделі. Модель є шаблоном, який містить формалізовані знання. Останній етап - інтерпретація – призначений для того, щоб з формалізованих знань отримати знання на мові наочної області.

Вся робота по аналізу даних в Deductor Studio базується на виконанні наступних дій: імпорт даних, обробка даних, візуалізація, експорт даних. Відправною точкою для аналізу завжди є процедура імпорту даних. Отриманий набір даних може бути оброблений будь-яким з доступних способів. Результатом обробки також є набір даних, який, у свою чергу, знову може бути оброблений. Імпортований набір даних, а також дані, отримані на кожному етапі обробки, можуть бути експортовані для подальшого використання в інших, наприклад, в облікових системах. Результати кожної дії можна відобразити різними способами: OLAP-кубы (крос-таблиця, крос-діаграма), плоска таблиця, діаграма, гістограма, статистика, аналіз за принципом "що-якщо", граф нейромережі, дерево - ієрархічна система правил, інше.

Послідовність дій, які необхідно провести для аналізу даних, називається сценарієм. Сценарій можна автоматично виконувати на будь-яких даних. Типовий сценарій змальований на рис. 1.17.

Deductor Warehouse - багатовимірне сховище даних, що акумулює всю необхідну для аналізу наочної області інформацію. Вся інформація в сховищі міститься в структурах типа "зірка", де в центрі розташовані таблиці фактів, а "променями" є виміри. Така архітектура сховища найбільш адекватна завданням аналізу даних. Кожна "зірка" називається процесом і описує певну дію. У Deductor Warehouse може одночасно зберігатися безліч процесів, що мають загальні виміри.

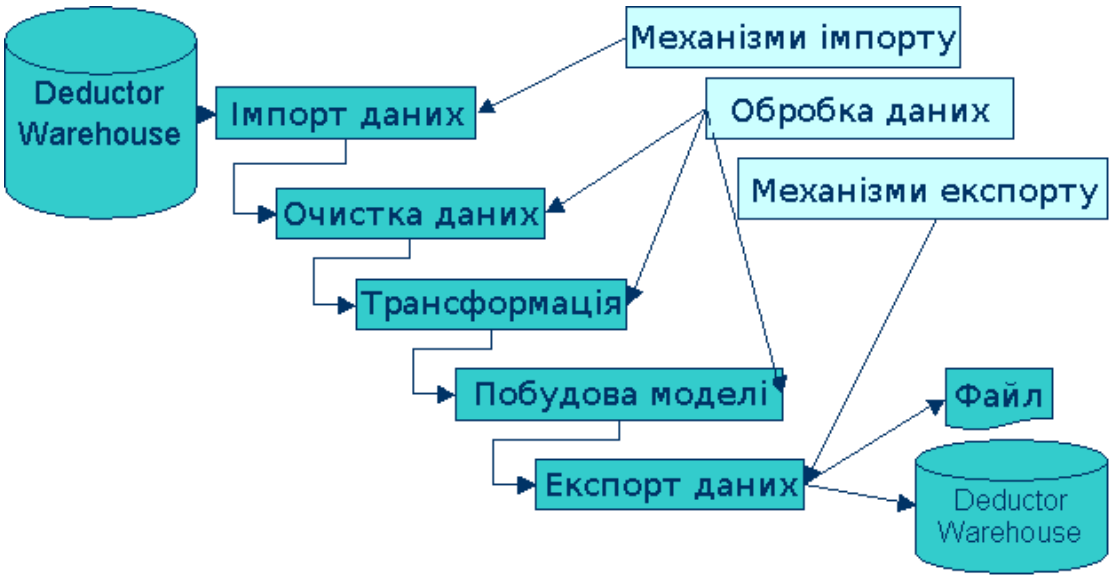


Рис. 1.17. Типовий сценарій Deductor Studio.

Що є сховищем Deductor Warehouse ? Фізично - це реляційна база даних, яка містить таблиці для зберігання інформації і таблиці зв'язків, що забезпечують цілісне зберігання відомостей. Поверх реляційної бази даних реалізований спеціальний прошарок, який перетворить реляційну систему до багатомірної. Багатомірна ідеологія використовується тому, що воно набагато краще реляційної відповідає ідеології аналізу даних. Завдяки цьому прошарку користувач оперує багатомірними поняттями, такими як "вимір" або "факт", а система автоматично виробляє всі необхідні маніпуляції, необхідні для роботи з реляційною СУБД.

Окрім консолідації даних, робота із створення закінченого аналітичного рішення містить декілька етапів.

Очищення даних. На цьому етапі проводиться редагування аномалій, заповнення пропусків, згладжування, очищення від шумів, виявлення дублікатів і протиріч.

Трансформація даних. Виробляється заміна порожніх значень, квантування, таблична заміна значень, перетворення до ковзаючого вікна, зміна формату набору даних.

Data Mining. Будуються моделі з використанням нейронних мереж, дерев рішень, самоорганізующихся карт , асоціативних правил.

Програмний комплекс KXEN. Програмне забезпечення KXEN є розробкою однойменної французько-американської компанії, яка працює на ринку з 1998 року. Абревіатура KXEN означає "Knowledge eXtraction Engines" - "движки" для витягання знань. Відразу слід сказати, що розробка KXEN має особливий підхід до аналізу даних. У KXEN немає дерев рішень, нейронних мереж і іншої популярної техніки. KXEN - це інструмент для моделювання, який дозволяє говорити про еволюцію Data Mining і реінжинірінге аналітичного процесу в організації в цілому. У основі цих тверджень лежать досягнення сучасної математики і принципово інший підхід до вивчення явищ в бізнесі. Слід зазначити, що все те, що відбувається усередині KXEN сильно відрізняється (принаймні, по своїй філософії) від того, що ми звиклися рахувати традиційним Data Mining.

Бізнес-моделювання KXEN - це аналіз діяльності компанії і її оточення шляхом побудови математичних моделей. Він використовується в тих випадках, коли необхідно зрозуміти взаємозв'язок між різними подіями і виявити ключові рушійні сили і закономірності в поведінці об'єктів, що цікавлять нас, або процесів. KXEN охоплює чотирьох основних типів аналітичних задач:

- Задачі регрессії - классифікації (в т.ч. визначення вкладів змінних).
- Задачі сегментації – кластеризації.
- Аналіз часових рядів.
- Пошук асоціативних правил (аналіз споживчої корзини).

Побудована модель в результаті стає механізмом аналізу, тобто частиною бізнес-процеса організації. Головна ідея тут - на основі побудованих моделей створити систему "крізного" аналізу процесів, що відбуваються. Це дозволяє автоматично виробляти їх оцінку і будувати прогнози в режимі реального часу (у міру того, як ті або інші операції фіксуються обліковими системами організації).

У 1990 році були отримані важливі результати в математиці і машинному навчанні. Ініціатором досліджень в цій області став Володимир

Вапник, що опублікував свою Статистичну Теорію Навчання. Він був першим, хто відчинив двері до нових доріг декомпозиції помилки, що отримується в процесі вживання методів машинного навчання. Він виявив і описав структуру цієї помилки і на основі зроблених виводів відшукав спосіб структурувати методи моделювання. Математичний апарат, який закладений в KXEN, в ході аналізу будує декілька конкуруючих моделей. Але цей процес здійснюється не випадковим чином (перебором різних методів моделювання), а шляхом вивчення різних наборів моделей з опорою на Теорію мінімізації структурних ризиків В. Вапника (Structured Risk Minimization). Творці KXEN розробили механізм порівняння моделей, з тим аби добитися найкращого співвідношення між їх точністю і надійністю, і вже цю оптимальну модель представити як результат аналізу користувачеві.

KXEN Analytic Framework за своєю суттю не є монолітним застосуванням, а виконує роль компонента, який вбудовується в існуюче програмне середовище. Цей "движок" може бути підключений до DBMS-систем (наприклад, Oracle або MS SQL-Server) через протоколи ODBS.

KXEN Analytic Framework є набором модулів для проведення описового і передбачаючого аналізу. Зважаючи на специфіку задач конкретної організації, конструюється оптимальний варіант програмного забезпечення KXEN. Завдяки відкритим програмним інтерфейсам, KXEN легко вбудовується в існуючі системи організації. Тому форма представлення результатів аналізу, з якою працюватимуть співробітники на місцях, може визначатися побажаннями замовника і особливостями його бізнес-процеса. На рис. 1.18 представлена структура KXEN Analytic Framework Version 3.0. Розглянемо ключові компоненты системи KXEN.

Компонент Агрегації Подій (KXEN Event Log - KEL) призначений для агрегації подій, подій за певні періоди часу. Вживання KEL дозволяє з'єднати транзакційні дані з демографічними даними про клієнта.

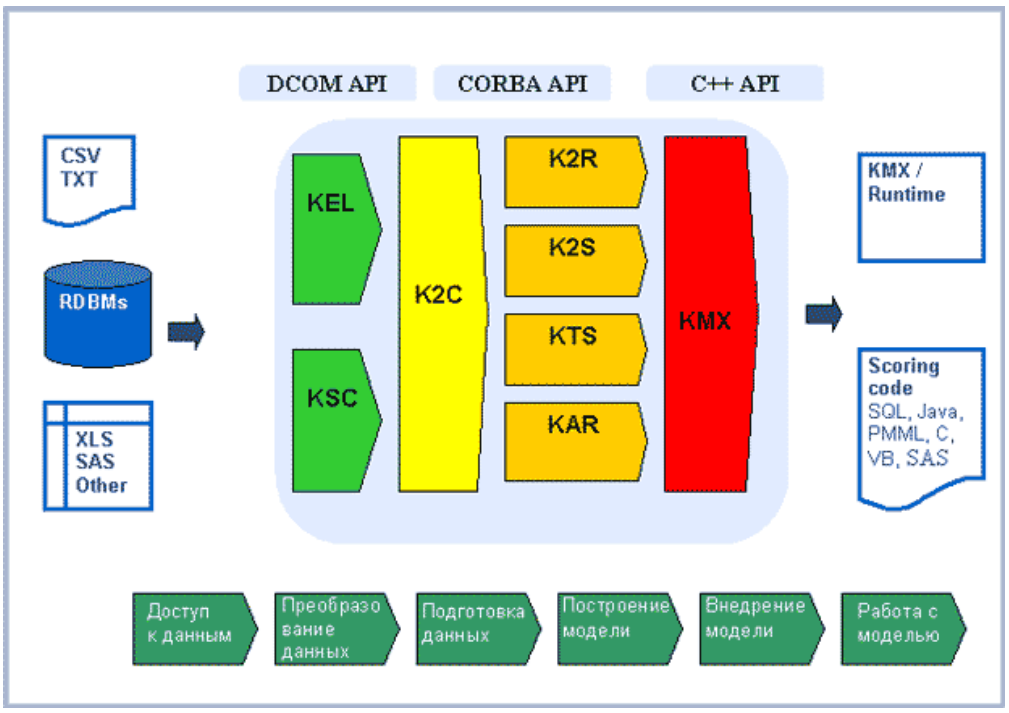


Рис. 1.18. Структура KXEN Analytic Framework Version 3.0.

Компонент використовується у випадках, коли "сирі" дані містять одночасно статичну інформацію (наприклад, вік, пів або професія індивіда) і динамічні змінні (наприклад, шаблони покупок або транзакції по кредитній карті). Дані автоматично агрегуються усередині визначених користувачем інтервалів без програмування на SQL або внесеннях змін в схему бази даних. Компонент KEL комбінує і стискує ці дані для того, щоб зробити їх доступними для інших компонентів KXEN. Перевагою використання даного компонента є можливість інтегрувати додаткові джерела інформації "на льоту" для того, щоб поліпшити якість моделі.

Компонент Кодування Послідовностей (KXEN Sequence Coder - KSC) дозволяє агрегувати події в серії транзакцій. Наприклад, потік "кліків" клієнта, що фіксується на Web-сайті, може трансформуватися в ряди даних для кожної сесії. Кожна колонка відображає конкретний перехід з однієї сторінки на іншу. Як і у випадку з KEL, нові колонки даних можуть додаватися до існуючих даних про клієнтів і доступні для обробки іншими компонентами KXEN. Перевагою використання даного компонента є можливість застосовувати

незадіяні раніше джерела інформації для того, щоб поліпшити якість прогнозуючих моделей.

Компонент Погодженого Кодування (KXEN Consistent Coder - K2C) дозволяє автоматично підготувати дані і трансформувати їх у формат, відповідний для використання аналітичними додатками KXEN. Використання K2C дозволяє трансформувати номінальні і порядкові змінні, автоматично заповнювати відсутні значення і виявляти викиди. Перевагою використання даного компонента є можливість автоматизації підготовки даних, яка дозволяє звільнити час безпосередньо досліджень і моделювання.

Компонент Робастной Регресії (KXEN Robust Regression - K2R) використовує відповідний регресійний алгоритм для того, щоб побудувати моделі, що описують існуючі залежності, і згенерувати прогнозуючі моделі. Ці моделі можуть потім застосовуватися для скоринга, регресії і класифікації. На відміну від традиційних регресійних алгоритмів, використання K2R дозволяє безпечно справлятися з великою кількістю змінних (більше 10 000). Модуль K2R будує індикатори і графіки, які дозволяють легко переконатися в якості і надійності побудованої моделі. Перевагою використання даного компонента є автоматизація процесу інтелектуального аналізу даних. Моделі дозволяють деталізувати індивідуальні вклади змінних.

Компонент Інтелектуальної Сегментації (KXEN Smart Segmenter - K2S) дозволяє виявити природні групи (кластери) в наборі даних. Модуль оптимізований для того, щоб знаходити кластери, які відносяться до конкретного поставленого завдання. Він описує властивості кожної групи і вказує на її відмінності від всієї вибірки. Як і у випадку з іншими модулями, цей модуль також будує індикатори якості і надійності моделі. Перевагою використання даного компонента є автоматичне виявлення груп, значимих для того конкретного завдання, яке необхідно вирішити.

Машина Опорних Векторов KXEN (Support Vector Machine - KSVM) дозволяє виробляти бінарну класифікацію. Використання компонента потрібно для вирішення задач, заснованих на наборах даних з невеликою кількістю

спостережень і великою кількістю змінних. Це робить модуль ідеальним для вирішення задач в областях з дуже великою кількістю розмірності, таких як медицина і біологія. Перевагою використання даного компонента є можливість вирішення завдань, які раніше вимагали написання спеціальних програм, за допомогою промислового програмного забезпечення.

Компонент Аналізу Часових Рядів (KXEN Time Series - KTS) дозволяє прогнозувати значимі шаблони і тренди у часових рядах. Використовує хронологічні дані, що накопичилися, для того, щоб спрогнозувати результати наступних періодів. Модуль KTS виявляє тренди, періодичність і сезонність для того, щоб отримати точні і достовірні прогнози. З'являється можливість підстроїтися під шаблони бізнесу, що повторюються, і передбачати скорочення постачань до того як вони стануться.

Компонент Експорту Моделей (KXEN Model Export - KMX) дозволяє створювати коди різного типа: SQL, C, VB, SAS, PMML і багато інших для вбудовування в існуючі застосування і бізнес-процеси. Побудована модель у вигляді кода може бути передана на іншу машину для подальшого аналізу даних в пакетному або інтерактивному режимі. Використання даного модуля дає можливість виробляти аналіз знов поступаючої інформації за допомогою моделі автономно, поза самою системою моделювання. Це істотно прискорює впровадження моделей у виробничий процес і не вимагає створення спеціальних умов і програм для аналізу всієї бази даних за допомогою моделі. Користувач також може перекласти модель тією мовою, яка підтримує його комп'ютер.

1.6. Новітні напрямки застосування Data Mining.

Як відмічалось раніше, область використання Data Mining нічим не обмежена - вона скрізь, де є які-небудь дані. Сучасна практика інтелектуального аналізу даних виділяє два основні напрями вживання систем

Data Mining: як масового продукту і як інструменту для проведення унікальних досліджень [19, 57, 70].

Слід зазначити, що на сьогоднішній день найбільшого поширення технологія Data Mining набула при вирішенні бізнес-задач. Можливо, причина в тому, що саме в цьому напрямі віддача від використання інструментів Data Mining може складати, за деякими джерелами, до 1000% і витрати на її впровадження можуть досить швидко окупитися. Зараз технологія Data Mining використовується практично у всіх сферах діяльності людини, де накопичені ретроспективні дані.

Серед новітних напрямків застосування технологій інтелектуального аналізу даних слід виділити Web-задачі, практика вирішення яких викликає поширений інтерес і набуває популярності.

Web Mining. Web Mining можна перевести як "видобуток даних в Web". Web Intelligence готовий "відкрити нову главу" в стрімкому розвитку електронного бізнесу. Здатність визначати інтереси і переваги кожного відвідувача, спостерігаючи за його поведінкою, є серйозною і критичною перевагою конкурентної боротьби на ринку електронної комерції. Системи Web Mining можуть відповісти на багато питань, наприклад, хто з відвідувачів є потенційним клієнтом Web-магазина, яка група клієнтів Web-магазина приносить найбільший дохід, які інтереси певного відвідувача або групи відвідувачів [3, 72].

Технологія Web Mining охоплює методи, які здатні на основі даних сайту виявити нові, раніше невідомі знання і які надалі можна буде використовувати на практиці. Іншими словами, технологія Web Mining застосовує технологію Data Mining для аналізу неструктурованої, неоднорідної, розподіленої і значної за об'ємом інформації, що міститься на Web-узлах. Згідно таксономії Web Mining, тут можна виділити два основні напрями: Web Content Mining і Web Usage Mining.

Web Content Mining має на увазі автоматичний пошук і витягання якісної інформації зі всіляких джерел Інтернету, переобтяжених

"інформаційним шумом". Тут також йде мова про різні засоби кластеризації і анотування документів. У цьому напрямі, у свою чергу, виділяють два підходи: підхід, заснований на агентах, і підхід, заснований на базах даних.

Підхід, заснований на агентах (Agent Based Approach), включає такі системи:

- інтелектуальні пошукові агенти (Intelligent Search Agents);
- фільтрація інформації - класифікація;
- персоніфіковані агенти мережі.

Приклади систем інтелектуальних агентів пошуку: Harvest (Brown і ін., 1994), FAQ-Finder (Hammond і ін., 1995), Information Manifold (Kirk і ін., 1995), OCCAM (Kwok and Weld, 1996) and ParaSite (Spertus, 1997), ILA (Information Learning Agent) (Perkowitz and Etzioni, 1995), ShopBot (Doorenbos і ін., 1996).

Підхід, заснований на базах даних (Database Approach), включає системи:

- багаторівневі бази даних;
- системи web-запросов (Web Query Systems). Наприклад, W3QL (Konopnicki і Shmueli, 1995), WebLog (Lakshmanan і ін., 1996), Lorel (Quass і ін., 1995), UNQL (Buneman і ін., 1995 and 1996), TSIMMIS (Chawathe і ін.., 1994).

Другий напрям Web Usage Mining має на увазі виявлення закономірностей в діях користувача Web-узла або їх групи. Аналізується наступна інформація: які сторінки переглядав користувач, яка послідовність перегляду сторінок. Також аналізується, які групи користувачів можна виділити серед загального їх числа на основі історії перегляду Web-узла.

Web Usage Mining включає наступні складові:

- попередня обробка;
- операційна ідентифікація;
- інструменти виявлення шаблонів;
- інструменти аналізу шаблонів.

При використанні Web Mining перед розробниками виникає два типи завдань. Перша стосується збору даних, друга - використання методів персоніфікації. В

результаті збору деякого об'єму персоніфікованих ретроспективних даних про конкретного клієнта, система нагромаджує певні знання про нього і може рекомендувати йому, наприклад, певні набори товарів або послуг. На основі інформації про всіх відвідувачів сайту Web-система може виявити певні групи відвідувачів і також рекомендувати їм товари або ж пропонувати товари в розсилках.

Завдання Web Mining можна підрозділити на такі категорії:

- попередня обробка даних для Web Mining;
- виявлення шаблонів і відкриття знань з використанням асоціативних правил, тимчасових послідовностей, класифікації і кластеризації;
- аналіз отриманого знання.

Text Mining. Text Mining охоплює нові методи для виконання семантичного аналізу текстів, інформаційного пошуку і управління. Синонімом поняття Text Mining є KDT (Knowledge Discovering in Text - пошук або виявлення знань в тексті). На відміну від технології Data Mining, яка передбачає аналіз впорядкованої в деякі структури інформації, технологія Text Mining аналізує великі і надвеликі масиви неструктурованої інформації. Програми, що реалізовують це завдання, повинні деяким чином оперувати природною людською мовою і при цьому розуміти семантику аналізованого тексту. Один з методів, на якому засновані деякі Text Mining системи, - пошук так званого підрядка в рядку [3, 45].

Call Mining. За словами аналітиків "видобуток дзвінків" може стати популярним інструментом корпоративних інформаційних систем. Технологія Call Mining об'єднує в собі розпізнавання мови, її аналіз і Data Mining. Її мета - спрощення пошуку в аудіо-архівах, що містять записи переговорів між операторами і клієнтами. За допомогою цієї технології оператори можуть виявляти недоліки в системі обслуговування клієнтів, знаходити можливості збільшення продажів, а також виявляти тенденції в зверненнях клієнтів. Серед розробників нової технології Call Mining ("видобуток" і аналіз дзвінків) - компанії CallMiner, Nexidia, ScanSoft, Witness Systems [9, 58].

У технології Call Mining розроблено два підходи - на основі перетворення мови в текст і на базі фонетичного аналізу. Прикладом реалізації першого підходу, заснованого на перетворенні мови, є система CallMiner. В процесі Call Mining спочатку використовується система перетворення мови, потім слідує її аналіз, в ході якого залежно від змісту розмов формується статистика телефонних викликів. Отримана інформація зберігається в базі даних, в якій можливий пошук, витягання і обробка. Приклад реалізації другого підходу - фонетичного аналізу - продукція компанії Nexidia. При цьому підході мова розбивається на фонеми, що є звуками або їх поєднаннями. Такі елементи утворюють розпізнавані фрагменти. При пошуку певних слів і їх поєднань система ідентифікує їх з фонемами.

Аналітики відзначають, що за останні роки інтерес до систем на основі Call Mining значно зріс. Це пояснюється тим фактом, що менеджери вищої ланки компаній, що працюють в різних сферах, в т.ч. в області фінансів, мобільного зв'язку, авіабізнесу, не хочуть витрачати багато часу на прослухування дзвінків з метою узагальнення інформації або ж виявлення яких-небудь фактів порушень. По словах Деніела Хонг, аналітика компанії Datamonitor: "Використання цих технологій підвищує оперативність і знижує вартість обробки інформації". Типова інсталяція продукції від розробника Nexidia обходиться в суму від 100 до 300 тис. дол. Вартість впровадження системи CallMiner по перетворенню мови і набору аналітичних застосувань складає близько 450 тис. дол.

На думку Шоллера, додатки Audio Mining і Video Mining знайдуть з часом набагато ширше вживання, наприклад, при індексації учбових відеофільмів і презентацій в медіабібліотеках компаній. Проте технології Audio Mining і Video Mining знаходяться зараз на рівні становлення, а практичне їх вживання - на самій початковій стадії.

На завершення розділу хотілося б трохи зупинитися на можливих варіантах впровадження систем інтелектуального аналізу даних. Різні варіанти впровадження Data Mining мають свої сильні і слабкі сторони. Так, перевагами

готового програмного забезпечення є готові алгоритми, технічна підтримка виробника, повна конфіденційність інформації, також не потрібно дописувати програмний код, існує можливість придбання різних модулів і надбудов до використовуваного пакету, спілкування з іншими користувачами пакету і інше. Проте, таке рішення має і слабкі сторони. Залежно від інструменту, це може бути достатньо висока вартість ліцензій на програмне забезпечення, неможливість додавати свої функції, складність підготовки даних, практична відсутність в інтерфейсі термінів наочної області та інше. Таке рішення вимагає наявність висококваліфікованих кадрів, які зможуть якісно підготувати дані до аналізу, знають, які алгоритми слід застосовувати для вирішення яких завдань, зуміють проінтерпретувати отримані результати в термінах вирішуваних бізнес-задач. Далеко не кожна компанія може тримати штат таких фахівців, а часто їх утримання навіть неефективно.

Представимо ситуацію, коли менеджер стикається "наодинці" з одним з продуктів, в якому реалізовані методи технології Data Mining (від найпростіших, таких, що включають 1-2 алгоритми, до повнофункціональних програмних комплексів, що пропонують десятки різних алгоритмів). Перед ним стоїть задача - виявити найбільш перспективних потенційних клієнтів, а він бачить перед собою всього лише набір математичних алгоритмів. Це і є "зворотна сторона" використання готових інструментів. Таким чином, покупці готового інструменту повинна передувати серйозна підготовка до впровадження Data Mining.

Розглянемо інший варіант впровадження, який припускає використання Data Mining консалтингу або так званої адаптації програмного забезпечення під конкретне завдання. За даними консалтингової компанії Meta Group, в світі не менше 85% ринку Data Mining займають саме послуги, тобто консультації по ефективному впровадженню цієї технології для вирішення актуальних бізнес-задач. В Інтернеті можна знайти перелік більше ста відомих компаній, що займаються консалтингом у сфері Data Mining. Наприклад, одна з всесвітньо відомих консалтингових компаній у сфері Data Mining - компанія Two Crows.

Вона спеціалізується на публікації звітності Data Mining, проводить освітні семінари, консультує користувачів і розробників Data Mining у всьому світі. Одна з відомих методологій Data Mining розроблена компанією Two Crows.

До консалтингових Data Mining-компаній відносять і деяких виробників готового програмного забезпечення: IBM Global Business Intelligence Solutions, SAS Institute, SPSS, Statsoft та інші. Деякі консалтингові компанії надають свої послуги на певних територіях. Це, наприклад, компанія Arvato Business Intelligence, яка забезпечує Data Mining консультування і моделювання у Франції, Німеччині, Іспанії і деяких інших європейських країнах. Деякі консалтингові компанії спеціалізуються на наданні послуг в певних наочних областях. Наприклад, компанія Blue Hawk LLC, здійснює Data Mining і надає консультаційні послуги в сферах Direct Marketing і CRM. Деякі компанії надають послуги з використанням певних методів Data Mining. Компанія Bayesia, надає консультування і "настройку рішення" під клієнта на основі байєсовськой класифікації. Компанія Visual Analytics забезпечує послуги з бізнес - консультування для знаходження шаблонів з використанням візуального Data Mining.

Розглянемо переваги, які має цей варіант впровадження Data Mining в порівнянні з готовими програмними продуктами і їх самостійним використанням.

Висококваліфіковані фахівці. Для ефективного застосування технології Data Mining потрібні кваліфіковані фахівці, які зуміють якісно провести весь цикл аналізу. Поки що таких грамотних фахівців на просторах СНД та України дуже небагато, і тому вони досить дорогі. Навчання ж власних, по-перше, достатньо ризиковано (його із задоволенням переманить конкурент), по-друге, потрібні чималі витрати бо стоїть це дорого. Клієнти, скориставшись послугами консалтингової компанії, дістають доступ до висококласних професіоналів компанії, економлячи при цьому значні кошти на пошуку або навчанні власних фахівців.

Адаптованість. Готові продукти призначені для вирішення хоч і широкого, але все таки стандартного і обмеженого круга задач, тому адаптація продукту до умов конкретного бізнесу лягає на плечі співробітників компанії. Тут перед замовником знову встане згадана проблема кваліфікованих фахівців. Консалтингова компанія надає послуги, повністю адаптовані під бізнес замовника і його задач.

Гнучкість інструменту. Його можливість швидко підстроїти програмне забезпечення під потреби бізнесу:

- можливість вибору найбільш зручних понять, в термінах яких повинні бути сформульовані знання або терміни наочної області. Так, аналізуючи конкурентів, що діють на ринку, їх можна поділити на "сильних" і "слабких" або ж на "агресивних", "спокійних" і "пасивних" - залежно від того, що цікавить аналітика в певний момент. Відповідно, знання будуть сформульовані у вибраних замовником термінах, і у результаті він отримує рішення саме в тих термінах, які йому цікаві і зрозумілі;
- отримання осмислених і зрозумілих замовникові знань в природній формі. Використання адаптованого під конкретний бізнес програмного забезпечення позбавить користувача від необхідності вивчення формул або залежностей в математичній формі, а надасть знання в найбільш інтуїтивному вигляді.

Розглянемо приклад практичного використання розглянутого підходу, який був застосован компанією Snow Cactus для оцінки кредитоспроможності позичальника банку. Реалізація задачі "Видавати кредит?" відбувалася в системі dm-Score, адаптованої під конкретну бізнес-задачу програмного забезпечення кредитного скорінга.

Система кредитового скорінга dm-Score (dm - від Data Mining) впроваджена в банку для аналізу кредитних історій і виявлення прихованих впливів параметрів позичальників на їх кредитоспроможність. Така система повинна вписуватися в інформаційний простір банку, тобто безпосередньо взаємодіяти з базами даних, де зберігається інформація про позичальників і

кредити, з автоматизованою банківською системою (АБС), іншим програмним забезпеченням, і працювати з ними як єдине ціле.

В процесі впровадження фахівці знайомляться з використовуваємою в банку АБС системою автоматизації рітейла, базами даних і т.д., погоджують з фахівцями вимоги до системи скорінга - як функціональні, так і не функціональні, а також вивчають, які дані накопичені банком і які задачі вони дозволяють вирішувати, здійснюють адаптацію системи відповідно до них. Однією з важливих переваг впровадження системи dm-Score є те, що в процесі впровадження враховуються всі індивідуальні вимоги і побажання до неї з боку банку. Важливо також відзначити, що в цьому випадку відбувається інтеграція системи dm-Score в інформаційний простір банку, а не навпаки, тобто впровадження не зажадає яких-небудь змін в існуючих бізнес-процесах. Таким чином, в результаті впровадження банк отримує систему скорінга, яка враховує всі специфічні особливості і потреби клієнта.

Описувана система dm-Score дозволяє вирішувати наступні задачі:

- оцінка кредитоспроможності позичальника (скорінг позичальника);
- ухвалення рішення про видачу кредиту або відмові в ньому. При цьому система може пояснити фахівцеві банку, чому було ухвалено саме таке рішення;
- визначення максимального розміру кредиту (ліміту кредиту по кредитній карті) на основі скорінга позичальника;
- винесення професійної думки про кредитний ризик по позиках;
- вироблення індивідуальних умов кредитування для кожного позичальника з урахуванням ризику для банку;
- прогнозування поведінки позичальника, тобто наявність і частоту прострочень конкретного позичальника, середній розмір використовуваного кредиту по кредитній карті і т.д.;
- оптимізація анкети позичальника (виключення не значущих питань без погіршення якості анкети);

- перевірка анкети конкретного позичальника на повноту і внутрішню несуперечність;
- рішення інших задач, специфічних для конкретного банку.

Ця система робить свої висновки на основі даних, вже накопичених банком в процесі роботи на ринку роздрібного кредитування. При цьому в процесі впровадження система настроюється саме на той набір даних, на який орієнтований конкретний банк. Іншими словами, система dm-Score готова працювати з тими даними, які є в наявності, і не вимагає фіксації на якій-небудь конкретній жорстко заданій анкеті. В процесі аналізу даних про позичальників і кредити застосовуються різні математичні методи, які виявляють в них чинники і їх комбінації, що впливають на кредитоспроможність позичальників, і силу їх впливу. Виявлені залежності складають основу для ухвалення рішень у відповідному блоці. Блок аналізу повинен періодично використовуватися для аналізу нових даних банку (приходять нові позичальники, поточні проводять виплати), для забезпечення актуальності системи і адекватності ухвалюваних нею рішень. Блок ухвалення рішень використовується безпосередньо для отримання висновку системи dm-Score про кредитоспроможність позичальника, про можливість видачі йому кредиту, про максимально допустимий розмір кредиту і т.д. З цим блоком працює співробітник банку, який або вводить в нього анкету нового позичальника, або отримує її з торгової точки, де банк здійснює програму споживчого кредитування. Завдяки тісній інтеграції системи з інформаційним простором банку, результати роботи цього блоку передаються безпосередньо АБС і системі автоматизації рітейла, які вже формують всі необхідні документи, ведуть історію кредиту і т.д. Таким чином, і система dm-Score, і всі банківські системи працюють як одне ціле, підвищуючи продуктивність праці співробітників банку.

Тест.

1. Під аналітичною технологією Ви розумієте:

а) методики, які на основі моделей дозволяють за відомими даними оцінювати значення невідомих характеристик;

б) методики, які на основі розрахунків дозволяють за відомими даними проводити обчислення економічних характеристик;

в) методики, які на основі моделей дозволяють за не відомими даними оцінювати значення невідомих характеристик.

2. Які з перерахованих технологій слід віднести до аналітичних технологій:

а) нарахування заробітної плати;

б) оцінка конкурентоспроможності компанії;

в) складання прайс-листів;

г) формування портфелю замовлень;

д) формування портфелю цінних паперів.

3. Класичними підходами до аналізу даних Ви вважаєте:

а) детерміновані технології;

б) імовірнісні технології;

в) фізичні технології;

г) лінгвістичні технології.

4. Технології інтелектуального аналізу використовуються для …

а) знаходження моделей і відносин при побудові середовища даних;

б) знаходження моделей і відносин, прихованих в базах даних;

в) знаходження моделей і відносин, прихованих в середовищі даних.

5. Термін Data Mining в Вашому уявленні співвідноситься з ….

а) автоматизованою обробкою економічної інформації;

б) інтелектуальним аналізом даних;

в) засобами пошуку закономірностей;

г) експертними системами;

д) інформаційною проходкою даних.

6. Яка концепція покладена у основу технології Data Mining ?

а) концепція взаємозв’язків, що відображають фрагменти багатоаспектних взаємин в даних;

б) концепція шаблонів, що відображають фрагменти багатоаспектних взаємин в даних;

в) концепція відносин, що відображають фрагменти багатоаспектних взаємин в даних.

7. Упорядкуйте етапи побудови моделі інтелектуального аналізу даних.

а) підготовка та огляд даних;

б) постановка задачі.

в) побудова моделей;

г) спостереження за моделлю;

д) використання моделей.

8. Якщо застосовується такий метод інтелектуального аналізу даних як класифікація, то за його допомогою Ви ...

а) виявляєте ознаки, що характеризують той або інший об'єкт;

б) виявляються функціональні залежності, що характеризують той або інший об'єкт;

в) виявляєте ознаки, що характеризують групу, до якої належить той або інший об'єкт.

9. При проведенні інтелектуального аналізу даних метод регресійного аналізу використовується …

а) коли взаємовідносини між змінними можуть бути виражені кількісно у вигляді деякої комбінації цих змінних;

б) коли взаємовідносини між змінними можуть бути виражені якісно у вигляді деякої комбінації цих змінних;

в) коли взаємовідносини між змінними можуть бути виражені кількісно у вигляді графіків та таблиць.

10. Ви застосовуєте метод прогнозування временних рядів в тому випадку, коли …

а) на основі аналізу поведінки временних рядів оцінюєте нинішні значення змінних;

б) на основі аналізу поведінку временних рядів оцінюєте майбутні значення прогнозованих змінних;

в) на основі аналізу поведінку временних рядів оцінюєте минулі значення прогнозованих змінних.

11. Якщо Ви застосовуєте метод кластеризації, то припускаєте що …

а) класи заздалегідь не задані і за допомогою моделі кластеризації засоби інтелектуальних обчислень самостійно створюють різнорідні групи даних;

б)класи заздалегідь задані і за допомогою моделі кластеризації засоби інтелектуальних обчислень самостійно створюють однорідні групи даних;

в) класи заздалегідь не задані і за допомогою моделі кластеризації засоби інтелектуальних обчислень самостійно створюють однорідні групи даних.

12. Коли, на Ваш погляд, застосовується метод асоціацій?

а) асоціація застосовується, коли декілька подій зв'язано між собою;

б) асоціація застосовується, коли декілька подій не зв'язано між собою;

в) асоціація застосовується, коли потрібен факторний аналіз подій.

13. Якщо при аналітичних дослідженнях Ви застосовуєте інтелектуальну технологію нейроних мереж, то розумієте, що …

а) нейронні мережі є сукупністю зв'язаних між собою прошарків нейронів, які отримують вхідні дані, здійснюють їх обробку і генерують на виході результат навчання;

б) нейронні мережі є сукупністю зв'язаних між собою прошарків нейронів, які отримують вхідні навчальні дані, здійснюють їх обробку і генерують на виході результат;

в) нейронні мережі є сукупністю зв'язаних між собою прошарків нейронів, які отримують вхідні дані, здійснюють їх обробку і генерують на виході результат.

14. Результатом застосування інтелектуальної технології дерева рішень є …

а) створення ієрархічної структури правил класифікації типу "ТОМУ... ЩО...", що мають вид дерева;

б) створення ієрархічної структури правил класифікації типу "ЯКЩО... ТОДІ...", що мають вид дерева;

в) створення ієрархічної структури правил класифікації типу "ЯКЩО... НІ...", що мають вид дерева.

15. Інтелектуальна технологія роздумів на основі аналогічних випадків робить прогноз майбутнього слідуючим чином:

а) система знаходить у минулому близькі аналоги наявної ситуації і вибирає ту ж відповідь, що була для її правильною;

б) система знаходить нові варіанти наявної ситуації і вибирає відповідь, що є правильною;

в) система знаходить у минулому близькі аналоги наявної ситуації і вибирає протилежну відповідь, що є правильною.

16. При впровадженні інтелектуальної технології виявлення асоціацій Ви передбачаєте, що …

а) будуть знайдені правила про окремі предмети, які з'являються разом у всіх транзакціях;

б) будуть знайдені правила про окремі предмети, які з'являються разом в одній транзакції;

в) будуть знайдені правила про всі предмети, які з'являються разом в одній транзакції.

17. В яких випадках Ви має намір застосовувати інтелектуальну технологію нечіткої логіки?

а) В тих випадках, коли існує "чому" в доповненні до "та" чи ні";

б) В тих випадках, коли існує "якщо" в доповненні до "та" чи ні";

в) В тих випадках, коли існує "може бути" в доповненні до "та" чи ні".

18. В основі інтелектуальної технології генетичних алгоритмів лежить …

а) метод, що імітує процес відбору рішень в множині Парето;

б) метод, що імітує процес природного відбору в природі;

в) метод, що імітує процес відбору рішень засобами дослідження операцій.

19. Якщо при аналітичних дослідженнях Ви застосовуєте інтелектуальну технологію еволюційного програмування, то розумієте, що …

а) в її основі лежать гіпотези про вид залежності цільової змінної від інших змінних, які формулюються системою у вигляді програм і цей процес будується як еволюція в світі програм;

б) в її основі лежать гіпотези про вид залежності цільової змінної від інших змінних, які формулюються системою у вигляді програм і цей процес будується як наперед задана залежність;

в) в її основі лежать гіпотези про вид залежності цільової змінної від інших змінних, які формулюються системою у вигляді задач математичного програмування і цей процес будується як еволюція в світі теорії операцій.

20. Програми візуалізації даних Вами можуть застосовуватися для …

а) інтелектуального аналізу величезних об'ємів даних;

б) математичного опису величезних об'ємів даних;

в) наочного узагальненя величезних об'ємів даних.

Розділ 2.

СХОВИЩА ДАНИХ ТА OLAP - ТЕХНОЛОГІЇ

2.1. Концепція сховищ даних

На початку восьмидесятих років минулого століття, в період бурхливого розвитку регіструючих інформаційних систем, виникло розуміння обмеженості можливостей їх застосування для аналізу даних і побудови на їх основі систем інтелектуального аналізу даних. Регіструючі системи створювалися для автоматизації рутинних операцій по веденню бізнесу - виписки рахунків, оформлення договорів, перевірки стану складу і т. п. Основними вимогами до таких систем були забезпечення транзакційності змін, що вносилися, і максимізація швидкості їх виконання. Саме ці вимоги визначили вибір реляційних СУБД і моделі представлення даних "суть-зв'язок" в якості основного технічного рішення при побудові регіструючих систем.

Для менеджерів і аналітиків у свою чергу були потрібні системи, які б дозволяли:

- аналізувати інформацію в часовому аспекті;
- формувати довільні запити до системи;
- обробляти великі об'єми даних;
- інтегрувати дані з різних реєструючих систем.

Очевидно, що регіструючи системи не задовольняли жодному з вищезгаданих вимог. У региструючій системі інформація актуальна тільки на момент звернення до бази даних, в наступний момент часу по тому ж запиту можна отримати абсолютно інший результат. Інтерфейс регіструючих систем розрахований на проведення жорстко певних операцій і можливості отримання результатів на нерегламентований (ad-hoc) запит сильно обмежені. Можливість обробки великих масивів даних також мала через налаштування СУБД на виконання коротких транзакцій і неминучого уповільнення роботи решти

користувачів. Відповіддю на виниклу потребу стала поява нової технології організації баз даних - технології сховищ даних (Data Warehouse) [4, 9, 45, 66].

За спостереженнями дослідницької компанії Forrester Research [71], більшість крупних компаній стикаються з наступною проблемою: вони накопичують величезну кількість інформації, яка ніколи не використовується. Практично в будь-якій організації реально функціонує безліч транзакційних систем, орієнтованих на оперативну обробку даних (кожна для конкретного класу задач) і безперервно поповнюють численні бази даних. Окрім цього, часто підприємства володіють величезними об'ємами інформації, що зберігається в так званих успадкованих системах. Всі ці дані розподілені по мережах персональних комп'ютерів, зберігаються на мейнфреймах, робочих станціях і серверах. Таким чином, інформація є, але вона розосереджена, неузгоджена, неструктурована, часто надмірна і не завжди достовірна. Тому в більшості організацій ці дані до цих пір не можуть бути використані для ухвалення критичних бізнес-рішень. На вирішення цього протиріччя і направлена концепція сховищ даних (Data Warehouse) (рис. 2.1.).

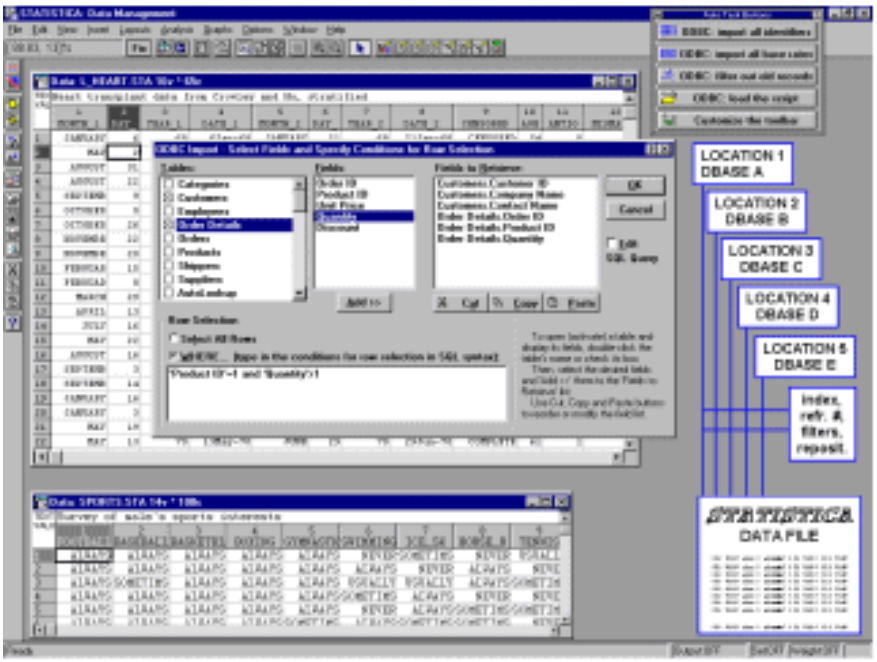


Рис. 2.1. Сховище даних SENS компанії StatSoft Enterprise Systems.

Зберігання даних є наріжним каменем систем бізнес-аналітіки. 87% компаній мають одне або більше функціонуючих сховищ даних, причому більше половини мають декілька подібних сховищ. Фактично, кожні чотири компанії мають двадцять або більше сховищ, які деяким способом логічно інтегровані, 18% цих сховищ має об'єм в терабайт або більше, 30% респондентів чекає, що об'єм їх сховища даних зросте удвічі або навіть у декілька разів за подальші два або три роки. Різноманітність джерел даних, що наповнюють сховища, просто дивує. Природно, кожне сховище використовує транзакційні системи як джерело даних. Проте відомо декілька незвичайних джерел даних, такі як Web-сайти електронної комерції (51%), прив'язки до баз даних, розташованих у замовників або бізнес-партнерів (49%), сервіси розширеного доступу до даних, які забезпечують характеристичну інформацію по замовниках (47%), витягання інформації з Web-сайтів (41%), сервіси пошуку/виявлення (34%) і канали новинних даних від крупних новинних агентств (33%). Нарешті, 47% компаній мають одну або більш вітрин даних, і 49% використовують накопичувачі операційних даних.

Білл Інмон, автор концепції, в своїй класичній статті «Що таке сховища даних» (1996 р. с. 4) визначає сховища даних як *«предметно орієнтовані, інтегровані, незмінні, такі, що підтримують хронологію набори даних, організовані для цілей підтримки управління»*. Він розглядає сховища як «єдине і одноосібне джерело істини», «центр всесвіту» систем підтримки прийняття рішень (СППР). «З сховищ даних, - пише він, - інформація перетікає в різні відділи, фільтруючись відповідно до заданих настройок СППР. Ці окремі бази даних для ухвалення рішень називаються вітринами даних» [16, 37, 51].

У основі концепції сховищ даних лежить ідея об'єднання корпоративних даних, розсіяних по системах оперативної обробки даних, історичних архівах і інших зовнішніх джерелах. Ці джерела можуть містити дані, які не використовуються безпосередньо в системах обробки інформації, але що є життєво необхідними для СППР: законодавча база (включаючи податкові прогнози), плани розвитку галузей, статистичні дані, електронні довідники. Як

показує практика, рішення, прийняте на основі лише внутрішніх даних, найчастіше виявляється некоректним.

Мета концепції сховищ даних - прояснити відмінності в характеристиках даних в операційних і аналітичних системах, визначити вимоги даним, що поміщаються в сховище, визначити загальні принципи і етапи його побудови, основні джерела даних, дати рекомендації по рішенню потенційних проблем, що виникають при їх вивантаженні, очищенні, узгодженні, транспортуванні і завантаженні до цільової базу даних сховища.

Предметом концепції сховищ даних є не аналіз даних, а власне дані, тобто концепція їх підготовки для подальшого аналізу. В той же час концепція сховища даних визначає не просто єдиний логічний погляд на корпоративні дані, а реалізацію єдиного інтегрованого джерела даних.

Розглянемо порівняльні характеристики даних, що застосовуються в інформаційних системах, орієнтованих на операційну і аналітичну обробку даних (таб. 2.1.).

Таблиця 2.1.

Порівняння характеристик даних.

Характеристика	Операційні	Аналітичні
Частота оновлення	Висока частота, маленькими порціями	Мала частота, великими порціями
Джерела даних	В основному внутрішні	В основному зовнішні
Об'єми зберігаємих даних	Сотні мегабайт, гігабайти	Гігабайти та терабайти
Вік даних	Поточні (за період від декількох місяців до одного року)	Поточні та історичні (за період в декілька років, десятки років)
Призначення	Фіксація, оперативний пошук і перетворення даних	Зберігання деталізованих і агрегованих історичних даних, аналітична обробка, прогнозування і моделювання

Згідно цієї таблиці сховища даних мають переваги порівняно з використанням оперативних систем або баз даних:

- на відміну від оперативних систем, сховище даних містить інформацію за весь необхідний часовий інтервал - аж до декількох десятиліть - в єдиному інформаційному просторі, що робить такі сховища ідеальною основою для виявлення трендів, сезонних залежностей і інших важливих аналітичних показників;

- як правило, інформаційні системи підприємства зберігають і представляють аналогічні дані по-різному. Наприклад, одні і ті ж показники можуть зберігатися в різних одиницях вимірювання. Одна і та ж продукція або одні і ті ж клієнти можуть іменуватися по-різному. У системах сховищ невідповідності в даних усуваються на етапі збору інформації і занурення її в єдину базу даних. При цьому організовуються єдині довідники, всі показники в яких приводяться до однакових одиниць вимірювання;

- дуже часто оперативні системи внаслідок помилок операторів містять деяку кількість невірних даних. На етапі поміщення в сховище даних інформація заздалегідь обробляється. Дані за спеціальною технологією перевіряються на відповідність заданим обмеженням і при необхідності коректуються (очищаються). Технологія забезпечує побудову аналітичних звітів на основі надійних даних і своєчасне сповіщення адміністратора сховища про помилки у вхідній інформації;

- універсалізація доступу до даних. Сховище даних надає унікальну можливість отримувати будь-які звіти про діяльність підприємства на основі одного джерела інформації. Це дозволяє інтегрувати дані, що вводяться і вже накопичені в різних оперативних системах, легко і просто порівнювати їх. При цьому в процесі створення звітів користувач не зв'язаний відмінностями в доступі до даних оперативних систем;

- прискорення отримання аналітичних звітів. Отримання звітів за допомогою засобів, що надаються оперативними системами, - спосіб неоптимальний. Ці системи витрачають значний час на агрегацію інформації (розрахунок сумарних, середніх, мінімальних, максимальних значень). Крім того, в поточній базі оперативної системи знаходяться тільки найнеобхідніші і

свіжіші дані, тоді як інформація за минулі періоди поміщається в архів. Якщо дані доводиться отримувати з архіву, тривалість побудови звіту зростає ще в два три разу. Слід також враховувати, що сервер оперативної системи часто не забезпечує необхідну продуктивність при одночасній побудові складних звітів і введенні інформації. Це може катастрофічно позначатися на роботі підприємства, оскільки оператори не зможуть оформляти накладні, фіксувати відвантаження або отримання продукції в той час, коли виконується побудова чергового звіту. Сховище даних дозволяє вирішити ці проблеми. По-перше, робота сервера сховища не заважає роботі операторів. По-друге, в сховищі крім детальної інформації містяться і наперед розраховані агреговані значення. По-третє, в сховищі архівна інформація завжди доступна для включення в звіти. Все це дозволяє значно скоротити час створення звітів і уникнути проблем в оперативній роботі;

- побудова довільних запитів. Інформацію в сховищі даних недостатньо тільки централізувати і структурувати. Аналітикові потрібні засоби візуалізації цієї інформації, інструмент, за допомогою якого легко отримувати дані, необхідні для ухвалення своєчасних рішень. Одна з головних вимог будь-якого аналітика - простота формування звітів і їх наочність. У разі оперативних систем побудова звітів часто позбавлена гнучкості; щоб створити новий звіт, доводиться задіювати фахівців інформаційного відділу, які об'єднують дані декількох систем. У разі ж використання сховища даних вирішення проблеми надає технологія OLAP (On-Line Analytical Processing). Ця технологія забезпечує доступ до даних в термінах, звичних для аналітика. Технологія OLAP базується на концепції багатовимірного представлення даних. Дійсно, кожне числове значення, що міститься в сховищі даних, має до декількох десятків атрибутів (наприклад, кількість продажів певним менеджером в певному регіоні на певну дату і т.п.). Таким чином, можна вважати, що робота йде з багатовимірними структурами даних (багатовимірними кубами), в яких числові значення розташовані на перетині декількох вимірювань. Саме цей підхід використовується в OLAP-системах. Вони надають гнучкі засоби

навігації по багатовимірних структурах - так звані OLAP-маніпуляції. З їх допомогою аналітик може отримувати різні зрізи даних, "крутити" дані.

Як видно з перерахованих переваг використання технології сховищ даних, велика їх частина може істотно спростити, підвищити швидкість і якісно поліпшити процес Data Mining. Таким чином, комплексне впровадження цих технологій дає розробникам і користувачам незаперечні переваги перед використанням розрізнених баз даних різних інформаційних систем при створенні систем підтримки прийняття рішень [3, 66].

В той же час, на сьогоднішній день є, принаймні, три істотні проблеми, пов'язані зі сховищами даних. Вони полягають в управлінні брудними даними, оптимальному виборі джерела даних, а також в продуктивності і масштабованості операцій, заснованих на скануванні.

З наслідками брудних даних ми стикаємося в своєму повсякденному житті. Ми отримуємо масу поштових відправлень з орфографічними помилками в іменах, безліч послань з різними варіантами одного і того ж імені, масу листів, адресованих людям, які давно переїхали, банківські повідомлення про численні витратні операції (тоді як гроші з рахунку знімалися лише одного разу) і т.д. До брудних даних відносяться відсутні, неточні або даремні дані з погляду практичного застосування (наприклад, представлені в невірному форматі, не відповідному стандарту). Брудні дані можуть з'явитися з різних причин, таких як помилка при введенні даних, використання інших форматів або одиниць вимірювання, невідповідність стандартам, відсутність своєчасного оновлення, невдале оновлення всіх копій даних, невдале видалення записів-дублікатів і т.д. Очевидно, що результати запитів, здобичі даних або бізнес-аналіза над сховищем, що містить велике число брудних даних, не можуть вважатися надійними і корисними. Представлені сьогодні на ринку засобу очищення даних (наприклад, продукти компаній Vality/Ascential Software, Trillium Software і First Logic) допомагають виявляти і автоматично коректувати деякі найбільш важливі типи даних, особливо, імена і адреси людей (з використанням національного каталогу імен і адрес). Проте цим засобам

належить пройти ще довгий шлях, оскільки сьогодні вони не вміють працювати зі всіма типами брудних даних, і далеко не всі компанії використовують навіть наявні засоби. Більш того, більшість підприємств не упроваджують надійні методики і процеси, що гарантують високу якість даних в сховищі. Недостатня увага, що приділяється якості даних, обумовлена відсутністю розуміння типів і об'єму брудних даних, проникаючих в сховища; впливу брудних даних, ухвалення рішень і виконувані дії; а також тим фактом, що продукти очищення даних, представлені на ринку, не дуже добре рекламуються або дуже дорого стоять. Для того, щоб почати приділяти необхідну увагу якості даних в своїх сховищах, підприємствам, перш за все, потрібно розібратися в різноманітті можливих брудних даних, джерелах їх появи і методах їх виявлення і очищення.

Поза сумнівом, що більшість організацій сьогодні не роблять достатніх зусиль для забезпечення високої якості даних в своїх сховищах. Для забезпечення високої якості даних потрібно мати процес, методології і ресурси для відстежування і аналізу якості даних, методологію для запобігання або виявлення і очищення брудних даних і методології для оцінки вартості брудних даних і витрат на забезпечення високої якості даних. У Ewha Women's University розроблений прототип інструментального засобу DAQUM (Data Quality Measurement), призначений для відстежування більшості типів брудних даних і приписування різним типам брудних даних кількісної міри якості даних залежно від особливостей програм.

Іншою проблемою сховищ даних є проблема вибору джерел даних. Сьогодні проектувальники сховищ даних проектують схему бази даних цільового сховища даних з використанням засобів моделювання баз даних. Схема бази даних складається з таблиць, стовпців (полів) таблиць, типів даних і обмежень стовпців, а також зв'язків між таблицями. Проектувальники також визначають відображення (перетворення) схем джерел інформації на схему цільового сховища даних. Але як проектувальники можуть переконатися в тому, що сховище даних містить всі дані, потрібні програмним комплексам і не

містить ніяких даних, які додаткам не потрібні? Сьогодні це грунтується на основі припущень досвідчених проектувальників. Їм доводиться виявляти потреби даних (таблиці і стовпці), опитуючи розробників програмних додатків, бізнес-аналітиків і адміністраторів баз даних. Після початкового створення сховища часто виявляється, що в ньому відсутні дані, потрібні для отримання відповідей на деякі запити, і присутні дані, які ніколи не потрібні додаткам. Хоча вартість зберігання може бути відносно невеликою, всі поля, потрібні і непотрібні, зберігаються в одних і тих же записах і прочитуються і записуються спільно, що уповільнює швидкість вибірки, збільшує час обробки, а також приводить до неефективного використання середовища зберігання.

Існує багато пропозицій по моделюванню сховища даних у вигляді репозитарія результатів всіх виконуваних запитів. Автори цих пропозицій намагаються знайти алгоритми, які вибиратимуть (для завантаження в сховищі даних) підмножину початкових даних, що мінімізує загальний час відповідей на запити. Деякі з авторів намагаються також мінімізувати вартість оновлення сховища даних. Іншими словами, вони ґрунтуються на припущенні, що всі запити до сховища даних можна наперед дізнатися або передбачити і що можна наперед дізнатися або передбачити всі можливі зміни сховища даних, а отже, і початкових джерел даних.

Ідеальний спосіб вибірки даних для завантаження в сховище даних полягає в тому, що перш за все визначаються всі запити, які генеруватимуться всіма програмними додатками, що виконуються над сховищем даних, і визначаються таблиці і поля, що фігурують в цих запитах. Визначення всіх запитів до створення сховища даних є важким завданням. Проте це може стати можливим після початкового створення сховища даних за рахунок реєстрації протягом розумного проміжку часу всіх запитів, що поступають від додатків. Аналіз зареєстрованих запитів може бути використаний для тонкої настройки сховища даних і видалення даних, до яких додатки не здійснюють доступ.

Потенційно корисним і практичним є засіб, який аналізує потреби додатків в даних, автоматично зіставляє ці потреби з схемами джерел даних і

видає рекомендації по складу оптимального піднабору джерел даних, які потрібно завантажити в сховищі даних, щоб в ньому знаходилися всі потрібні дані і не знаходилися які-небудь непотрібні. Таким засобом є MaxCentra. Функціонування MaxCentra спирається на наявність попередньої побудованої бази знань ключових слів, яка представляє потреби програмних додатків в даних. Ключові слова в основному є неявними вказівками таблиць і полів, до яких здійснюватиметься доступ при виконанні запитів, що генеруються додатком. Такий список ключових слів може бути забезпечений бізнес-аналітиками або розробниками програмних додатків, або ж він може бути отриманий автоматично шляхом аналізу запитів від додатків, що виконуються над неоптимізованим сховищем даних. MaxCentra відштовхується саме від цього і за підтримки і сприянні проектувальників дозволяє отримати оптимальну схему бази даних для сховища даних.

Для сховищ даних існує також проблема продуктивності та масштабованості. У системах реляційних баз даних для вибірки невеликої кількості потрібних записів без повного сканування таблиці або бази даних використовуються різні методи доступу, такі як індекси на основі хешування або B+-дерев. Такі методи доступу вельми ефективні при вибірці по одному ключовому полю (або невеликому числу полів), коли результати є малою частиною таблиці. Прикладами подібних запитів є: «Знайти всіх 25-річних людей» або «Знайти всіх 25-річних інженерів-програмістів». Для швидкої відповіді на такі запити можна створити індекси: на стовпці «вік» таблиці «співробітники» або на стовпцях «вік» і «посада» таблиці «співробітники» відповідно. Проте методи доступу в загальному випадку не допомагають при відповіді на запити, результатами яких є значна частина таблиці. Прикладами є запити: «Знайти всіх співробітників жіночого полу» або «Знайти молодих співробітників». Крім того, методи доступу не приносять користі, якщо значення стовпця часто змінюється, оскільки такі зміни вимагають перебудови методів доступу.

Крім подібних «простих» запитів існують два класи операцій, для яких методи доступу в системах реляційних баз даних стають безсилими. До першого класу відносяться операції «агрегації», що передбачають групування всіх записів таблиці і застосування до згрупованих записів агрегатних функцій (середнього значення, загального числа, суми, мінімального або максимального значення). Цей тип операцій важливий в таких програмних додатках як аналіз даних Web-журналів, сегментації даних про замовників і т.д. На ринку є продукти MaxScan і Ab Initio, призначені для вирішення проблем продуктивності і масштабованості при виконанні даного типу операцій. У MaxScan застосовується метод зберігання таблиць по полях, методи хешування для групування і агрегації записів, а також методи паралельної обробки. Продуктивність і масштабованість при виконанні агрегатних операцій в 10-20 разів перевищують показники систем реляційних баз даних. Пакет Ab Initio є засобом ETL, в якому в механізмі трансформації даних використовуються методи підвищення продуктивності.

До іншого класу відноситься операції «переміщення файлів», що читають і/або записують файли цілком. Цей тип операцій важливий на етапі «перетворення даних», що вимагає великих тимчасових витрат при створенні сховища даних або на етапі «підготовки даних» при автоматичному витяганні знань (здобичі даних) з наявних джерел. Етап перетворення даних включає трансформацію формату і представлення даних в заданих полях (зміна одиниці вимірювання, зміна формату дати і часу, зміна абревіатури і т.д.), злиття двох або більше полів в одне, розщеплювання поля на два або більше полів, сортування таблиці, побудову узагальненої таблиці з таблиць, що містять деталізовані дані, створення нової таблиці шляхом з'єднання двох або більше таблиць, розщеплювання таблиці на дві або більше і т.д. До етапу підготовки даних відноситься перетворення даних заданого поля в цифровий код (у нейронних мережах), перетворення безперервних цифрових даних в заданому полі в категоричні дані (наприклад, вік, що перевищує 60 років, вважається «пенсійним»), додавання до запису нового поля, взяття з таблиці зразків даних,

реплікація в таблиці деяких записів (для досягнення бажаного розподілу записів) і т.д.

Сьогодні операції переміщення файлів знаходяться в майже повній залежності від послідовних операцій систем реляційних баз даних над файлами, тобто читання одного або більше файлів, створення тимчасового файлу і запису результуючого файлу або файлів. Частота виконання подібних операцій і об'єм використовуваних даних може зробити виправданим застосування сервера перетворення/підготовки даних. У ідеалі такий сервер може складатися з декількох паралельно працюючих процесорів. Незалежно від конфігурації процесора на ньому повинні виконуватися програмні засоби перетворення/підготовки даних, спроектовані для паралельної обробки. Коли це виправдано, потрібно застосовувати конвеєрну паралельну обробку, при якій отримання часткових результатів однієї операції ініціюють виконання іншої операції без потреби очікування завершення першої операції. Конвеєрна паралельна обробка усуває потребу в записі в тимчасовий файл повних результатів однієї операції і в їх читанні наступною операцією, що дозволяє заощадити декілька процесів обробки файлів.

Технологія сховищ даних зародилася в США, а потім отримала визнання по всьому світу. Перші сховища даних з'явилися ще на початку 80-х років минулого століття (тоді вони називалися бази атомарних даних), але тільки до кінця 80-х років була повною мірою усвідомлена необхідність інтеграції корпоративної інформації і належного управління нею, а також з'явилися технічні можливості для створення відповідних систем, спочатку названих "сховищами інформації" (Information Warehouse - IW) і лише пізніше отримавших свою сучасну назву "сховища даних" (Data Warehouse - DW) . На початку 90-х, з появою технологій витягання, перетворення і завантаження даних (Extraction, Transmission and Loading - ETL) і оперативної аналітичної обробки (On-Line Analitic Processing - OLAP) почалося активне розповсюдження сховищ даних в комерційному секторі. Цьому процесу сприяло також опублікування першої книги Білла Інмона (W.H. Inmon. Building

the Data Warehouse, QED/Wiley, 1991, 312 p.), який отримав загальне визнання як "батько концепції сховища даних". Незабаром технологія сховищ даних перетворилася на розвинену архітектуру, відому як фабрика корпоративної інформації (Corporate Information Factory - CIF).

Іншим важливим напрямом, який сприяв появі сховищ даних, була поява систем підтримки ухвалення рішень (DSS — Decision Support Systems) та інформаційних систем для вирішень (EIS — Executive Information Systems). Не претендуючи на володіння власним інтелектом, на відміну від всього того, що обговорювалося в 60–70-і роки і що залишилося в області вічних ідей, системи DSS і EIS виявилися практично корисними: вони стали прототипами сучасних систем розтину даних, онлайнової аналітики, нової дисципліни корпоративного управління на основі знань. Системи DSS були спочатку налаштовані на менеджмент середньої ланки, а EIS намагалися дати більш загальне і багатовимірне бачення над полем даних — для керівників корпоративного рівня. Саме ці дві технологічні ідеї, що глибоко перетиналися між собою, з'явилися, найімовірніше, прямими предками сучасної концепції сховищ даних.

Розглянемо типові проблеми, що вирішуються за допомогою сховищ даних. До них відносяться, зокрема, аналіз клієнтської бази, аналіз продажів і аналіз доходів, а також управління пасивами і активами та інші.

Аналіз клієнтської бази дозволяє сформувати цільові сегменти клієнтів і використовувати цю інформацію при продажі банківських продуктів і послуг. Цільові сегменти формуються на основі демографічних і фірмографічних відомостей, фінансових показників (наприклад, обороту або прибутку), галузевих ознак і інших параметрів клієнтів. Одним з найбільш важливих питань є виділення сегментів прибуткових клієнтів, націлене на їх подальше утримання. Зокрема, за рахунок детальнішої сегментації підрозділи маркетингу починають краще розуміти потреби клієнтів і можуть використовувати ці дані при проведенні маркетингових кампаній. Аналіз клієнтської бази і сегментація дають можливість наблизитися до реалізації концепції індивідуального

маркетингу і ефективніше застосовувати систему управління взаєминами з клієнтами.

Аналіз продажів допомагає виявляти тенденції, планувати продажі по продуктах, клієнтах, підрозділам і, виходячи з результатів збуту, будувати механізми стимулювання клієнтських і продуктових підрозділів. Завдяки використанню сховища даних можна отримати інтегроване уявлення про результати продажів і узяти цю інформацію на озброєння при формуванні планів.

Аналіз доходів актуальний для будь-якого банку, причому понад усе затребуваний аналіз в розрізі клієнтів. Дуже важливо також мати уявлення про розподіл доходів по продуктах і послугах, каналах надання послуг і підрозділах банку. Аналіз доходів в розрізі клієнтів і продуктів дозволяє формувати “унікальні” пропозиції для кожного “унікального” клієнта з метою максимізації прибутку в довгостроковій перспективі. Він сприяє формуванню цінової політики банку, виділенню сегментів, продуктів і послуг, які стратегічно важливі для нього. Наприклад, банк Chase Manhatten Bank має сховище об'ємом більше 560 Гбайт, компанія Mastercard Online - 1,2 Тбайт. Коли всі дані містяться в єдиному сховищі, вивчення зв'язків між окремими елементами даних може бути плідніше, а результатом аналізу стають нові знання.

Управління активами і пасивами. За допомогою сховища даних можна проводити ефективний аналіз активів і пасивів і управляти не тільки ними, але і миттєвою ліквідністю банку на основі інструментального і портфельного підходів. Ці завдання вирішуються при мінімальних витратах на підготовку спеціальних даних і з урахуванням лише обмеженого об'єму інформації, що збирається з джерел у філіалах. Програмний комплекс забезпечує завантаження з інформаційних джерел семи типів і дозволяє формувати декілька десятків звітів.

Існують різні типи сховищ даних, які мають свою специфіку.

Фінансові сховища даних. В більшості випадків фінансові сховища даних це сховища, які організації будують в першу чергу. Створення фінансового сховища - дуже привабливе рішення, оскільки:

- фінансові дані завжди знаходяться в центрі «мозоку» організації;
- у більшості організацій фінансові дані представляють найменші об'єми даних з тих, що є;
- фінанси охоплюють всі аспекти функціонування компанії;
- фінансові дані за своєю природою мають структуру, на яку безпосередньо впливає повсякденна практика обробки фінансової інформації.

З цих причин фінанси стають переважною областю побудови корпоративного сховища даних. Проте, фінансові сховища даних мають серйозні, властиві тільки цьому типу сховищ, недоліки. Перший з них полягає в тому, що в організаціях чекають, що відомості з фінансових сховищ з точністю до однієї копійки співпадатимуть з даними існуючого фінансового середовища. Раз у раз можна почути, що «це фінансове сховище явно несправне, тому що в звіті, який я отримав вчора, було вказано, що доходи складають 145,998.32 доларів, коли ж я виконав той же звіт у фінансовому сховищі даних, то отримав величину, рівну 139, 762.01 доларам. Цьому сховищу просто не можна вірити». Очікування того, що інформація у фінансовому сховищі даних повинна точнісінько співпасти з цифрами з поточного фінансового звіту, є глибоко помилковим. Люди (тобто фінансові працівники), які так думають, просто не розуміють, що, коли дані переходять з операційного середовища у фінансове сховище даних, відбувається трансформація. А коли дані перетікають з світу додатків в реальний світ компанії, їх розглядають в іншому вимірюванні. А ось, що точно відбувається при такому переході даних з одного світу в інший:

- міняються звітні періоди. У операційному середовищі звітний період завершується в кінці місяця, в середовищі сховища даних закінчується на корпоративному календарі;
- міняються схеми угрупування і кодування рахунків. У операційному середовищі дані розраховуються відповідно до одного плану бухгалтерських

рахунків, а у фінансовому середовищі всієї компанії може бути абсолютно інший набір схеми угрупування і кодування;

- міняються класифікації даних. Так, в операційному середовищі Північна Америка складається всього з 48 континентальних штатів, в глобальному сховищі даних Північна Америка включає також Канаду, Мексику, Аляску і острови Карибського басейну.
- міняються валюти. Операційні грошові кошти відповідають тій валюті, в якій вони звертаються: гривни, євро, фунти, долари і так далі. У глобальному середовищі гроші перетворяться до однієї загальної валюти: доларам або євро.

Сховища даних в області страхування. Сховища даних в області страхування за деякими невеликими виключеннями схожі на інші сховища. Перше виключення (і це особливо справедливо відносно страхування життя) полягає в тому, що тривалість існування наявних сховищ дуже велика. Такі сховища містять дані, які є старими, дуже старими. Причина, по якій страхові компанії вимушені цікавитися такими даними - актуарна обробка даних. Практично для кожної справи приводиться довід, що діяльність, якою організація займалася в 1950 році, практично не пов'язана з сьогоднішнім заняттям. І часто цей довід звучить правдоподібно.

Друга відмінність цих сховищ визначається датами, які зберігаються в цьому бізнесі. Середовище страхування - відрізняється наявністю величезного числа дат, пов'язаних з бізнесом, чим який-небудь інший видом діяльності. Так, у сфері роздрібної торгівлі є декілька важливих дат: дата продажу, дата появи на складі, можливо, дата виробництва. У банківській справі істотна дата транзакції. У телекомунікації - це дата телефонного дзвінка. У страхуванні ж присутні дати всіляких типів.

Нарешті, третя відмінність полягає в тому, що ці сховища даних використовують свій робочий цикл ділової активності. Більшість організацій має вельми обмежений і короткий економічний цикл. Так, в банках це обналічуваня чека. У торгівлі - покупка виробу. У телефонній компанії - дзвінок. Проте, в страхуванні їм може бути заявка на страхове відшкодування,

яка може бути задоволена через п'ять років. Або закриття поліса може супроводжуватися двомісячним відстроченням. Резюмуючи, можна сказати, що швидкість, з якою функціонує страхування, відрізняється від швидкості, характерної для інших галузей. Ця різниця в швидкості відбивається в сховищі даних. У інших сховищах транзакції просто збираються і обробляються. В області страхування транзакція може відкладатися на невизначений термін, а її різні частини можуть відбиватися в сховищі даних. Результатом цього є абсолютно особливий підхід при проектуванні і впровадженні таких сховищ даних.

Сховища даних для управління людськими ресурсами. Сховища даних для управління людськими ресурсами мають вельми істотні відмінності від інших сховищ. Перша відмінність - число предметних областей. Таке сховище даних неминуче має одну важливу предметну область - це працівник. Практично все інше підпорядковане цій області або займає другорядне положення. Більшість же інших сховищ даних мають декілька базових предметних областей. Проте, основна відмінність сховищ даних для управління людськими ресурсами полягає в тому, що такі сховища взагалі-то використовують дуже мало транзакцій. Так, є дата, коли суб'єкт стає працівником, дата, коли людина звільняється, а також річні надбавки і підвищення. Але, окрім транзакцій фонду заробітної плати і інших рідкісних, згенерованих працівником транзакцій, в такому сховищі практично більше нічого і немає. Порівняйте сферу управління людськими ресурсами з комунікацією або банківським середовищем, і різниця в числі транзакцій стане очевидною. Ця різниця в темпах транзакцій між тією, що розглядається і іншими сферами діяльності є причиною виникнення певної складності, яка полягає в тому, що в області управління людськими ресурсами спостерігається тенденція до об'єднання операційної обробки людських ресурсів і обробки людських ресурсів для систем ухвалення рішення в одне середовище. У інших же галузях спокуса зробити таку архітектурну помилку вельми невелика.

Глобальні сховища даних. Глобальні сховища даних призначені для глобального представлення корпорації. Розрізняють три типи таких сховищ:

- географічно превалююча обробка даних. Наприклад, необхідно інтегрувати бізнес в Гонконзі з бізнесом в Парижі, який у свою чергу слід інтегрувати з Ріо-де-Жанейро, а той - з Нью-Йорком.
- функціонально превалююча обробка даних. Виробнича діяльність повинна бути інтегрована з постачанням, яке необхідно інтегрувати з продажами, а ті - з дослідженнями і так далі.
- галузева превалююча обробка даних. Наприклад, потрібно інтегрувати друкарську справу з консалтингом, який підлягає інтеграції з бізнесом у сфері медичного устаткування, а той із спеціалізацією в області програмного забезпечення.

Особливість глобального сховища даних полягає в тому, що на глобальному рівні часто дуже мало загальних вимірювань. Єдине загальне вимірювання - це гроші. І інтеграція бізнесу може бути досягнута тільки з його допомогою. Інші ж вимірювання можуть мати або не мати сенсу на глобальному рівні. Так, клієнт, продукт, постачальник, транзакція - всі ці класичні предметні області можуть бути як присутніми, так і бути відсутнім в глобальній інтегрованій сфері - глобальному сховищі даних. Крім цього, глобальне сховище даних схильне до того, від чого інші сховища захищені - від руйнівної дії змін. Якщо в інших сховищах зміни базових даних трапляються нечасто, то для цього типу сховищ вони відбуваються постійно і в самій основі. Так, у будь-який момент може бути відкрите нове родовище нафти, наприклад, у Венесуелі. У наступну хвилину в Перу спалахне революція. А потім, завдяки розвитку технології, стануть доступними поклади нафти в Луїзіані. Услід за цим послідують санкції ОПЕК. У Мексиці буде змінено законодавство. І так далі. Якщо розглядати ситуацію в глобальному аспекті, то видно, що зміни носять постійний характер. Тому структура і технологія, використовувана для розміщення і обслуговування глобального сховища даних, повинні дозволяти підтримувати ці безперервні зміни.

Сховища даних з можливостями Data Mining/Data Mining і Exploration. Сховища даних, що підтримують технологію Data Mining і Exploration є гібридом класичних сховищ. Такі сховища використовуються для виконання могутньої статистичної обробки даних. Ці сховища є дуже детальними, глибоко історичними, оптимізованими для статистичного аналізу. Крім того, для таких сховищ характерна орієнтація на який-небудь проект. Це означає, що, на відміну від всіх інших типів сховищ даних, їх перестають використовувати відразу після закінчення аналізу, ради якого вони створювалися. Ще одна важлива відмінність сховищ даних з можливостями Data Mining/Data Mining і Exploration полягає в тому, що ці сховища дуже часто включають зовнішні дані. Такі дані дуже корисні з погляду забезпечення бізнес-перспективи, яку не так легко побачити без їх участі.

Сховища даних в області телекомунікацій. Відмітна особливість цих сховищ полягає в тому, що вони в значній мірі визначаються даними, що згенеровані в деталях на рівні дзвінка. Зрозуміло, в галузі телекомунікації присутня безліч інших типів даних. Але жодна інша область сховищ даних не зумовлюється в такому ступені розміром однієї предметної області - деталями на рівні дзвінка. Існують багато способів зберігання деталей на рівні дзвінка:

- зберігання деталей на рівні дзвінка тільки за декілька місяців;
- зберігання безлічі деталей на рівні дзвінка, розміщених на різних носіях;
- резюмування або агрегація деталей на рівні дзвінка;
- зберігання тільки відібраних деталей на рівні дзвінка, і так далі.

На жаль, не дивлячись на різноманітність методів обробки, для даного сховища даних обробка може бути виконана тільки над деталями на рівні дзвінка. А робота на підсумковому або агрегованому рівні просто неможлива.

2.2. Технології побудови сховищ даних

Ідея, покладена в основу технологій інформаційних сховищ, полягає в тому, що проводити оперативний аналіз безпосередньо на базі інформаційних систем неефективно. Натомість, всі необхідні для аналізу дані витягуються з декількох традиційних баз даних (в основному, реляційних), перетворюються і потім поміщаються в одне джерело даних – сховище даних [4, 9, 11, 49].

В процесі занурення дані:

- очищаються - усунення непотрібної інформації;
- агрегуються - обчислення сум, середніх;
- трансформуються - перетворення типів даних, реорганізація структур зберігання;
- об'єднуються із зовнішніх і внутрішніх джерел - приведення до єдиних форматів;
- синхронізуються - відповідність одному моменту часу.

Сьогодні, технології побудови сховищ даних є основою для створення повноцінних інтелектуальних система налізу даних, орієнтованих на рішення слабко структурованих задач прийняття рішень, оскільки вони містить дані, що володіють наступними властивостями:

Цілісністю і внутрішнім взаємозв'язком. Хоча дані занурюються з різних джерел, але вони об'єднані єдиними законами іменування, способами вимірювання атрибутів і т.д. Це має велике значення для корпоративних організацій, в яких одночасно можуть експлуатуватися різні по своїй архітектурі обчислювальні системи, що представляють однакові дані по-різному. Наприклад, можуть використовуватися декілька різних форматів представлення дат, або один і той же показник може називатися різним чином, наприклад, "вірогідність доведення інформації" і "вірогідність отримання інформації". В процесі занурення подібні невідповідності усуваються автоматично.

Предметною орієнтованістю. Локальні бази даних містять мегабайти інформації, абсолютно не потрібної для аналізу (адреси, поштові індекси, ідентифікатори записів і т.п.). Подібна інформація не заноситься в сховище, що обмежує спектр розглядаємих при ухваленні рішення даних до мінімуму.

Відсутністю часової прив'язки. Оперативні системи охоплюють невеликий інтервал часу, що досягається за рахунок періодичної архівації даних. Сховища даних, навпаки, містять історичні дані, накопичені за великий інтервал часу (роки, десятиліття).

Доступністю виключно для читання. Модифікація даних не проводиться, оскільки вона може привести до порушення цілісності сховища даних. Оскільки не потрібно мінімізувати час занурення, то структура сховища може бути оптимізована для обробки певних запитів, що досягається за рахунок денормалізації реляційної схеми, попередньої агрегації і побудови найбільш доречних індексів.

Інтегрованість означає, що дані задовольняють вимогам всього підприємства, а не одній функції бізнесу. Цим сховище даних гарантує, що однакові звіти, що згенерували для різних аналітиків, міститимуть однакові результати.

Незмінність означає, що, потрапивши один раз в сховищі, дані там зберігаються і не змінюються. Дані в сховищі можуть лише додаватися.

Всі дані, які містяться в сховищі, можна розділити на наступні категорії:

- метадані (дані про даних);
- агреговані дані;
- детальні дані.

Важливою особливістю систем інтелектуального аналізу даних на основі сховищ даних є метадані. Це різного роду системні словники, що дозволяють контролювати склад і структуру інформації в сховищі, управляти процесами завантаження і розрахунків і т.п. Без прошарку управлінських даних сховище з часом загрожує перетворитися на велике електронне звалище.

Ключовою відмінністю інформації сховища є не тільки спосіб його наповнення (з різних зовнішніх і внутрішніх джерел), але і модель зберігання даних, особливо це стосується агрегованої інформації. У сховищі інформація розміщується в денормалізованому вигляді у формі класичної «сніжинки» або «зірки». Такий підхід дозволяє істотно понизити час відгуку бази даних при виконанні запитів. Не вдаючись до технологічних особливостей проектування, відзначимо, що відбувається це за рахунок деякої надмірності зберігання даних. Така форма зберігання інформації в корпоративних сховищах є загальносвітовою практикою. Разом з тим частина детальних даних цілком може зберігатися в нормалізованих таблицях. Вимога до спеціальних структур зберігання обумовлена деяким спеціальним способом використання даних, про що доцільно сказати докладніше. Використовувати інформацію, накопичену в сховищі, можна як за допомогою традиційних звітів, так і з використанням динамічних запитів до бази даних. Існують також абсолютно специфічні способи використання інформації, призначені спеціально для аналітичних завдань. До них відносяться так звані OLAP-технології (On-line Analytical Processing) і технології інтелектуального аналізу даних. З практичної точки зору рішучий крок, який робить OLAP-технологія, полягає в тому, щоб, відмовившись від зайвої спільності, зробити процес аналізу максимально швидким. В рамках цієї технології передбачається, що склад і структура показників для аналізу відомий наперед і міняється дуже рідко (для деяких видів систем - практично не міняється). Користувач може виконувати над даними в такому багатовимірному уявленні набір OLAP-операцій підйому (консолідації по деяких напрямах), спуску (деталізації по деякому напряму), повороту (зміни напряму сортування). Детальні дані в системах сховищ даних найчастіше є джерелом для інтелектуального аналізу.

Таким чином, дані, занурені в інформаційне сховище даних, організовуючись в інтегровану цілісну структуру, володіють природними внутрішніми зв'язками, набувають нових властивостей, що дає їм можливість набути статус інформації.

Розглянемо характеристики інтеграції даних в сховищі даних. Як вже відомо, метою інтеграції даних є отримання єдиної та цільної картини бізнес-даних. Для її досягнення застосуємо модель, яка включає додатки, продукти, технології та методи:

- додатки - це рішення, створені постачальниками відповідно до вимог клієнтів, які використовують один або більше продуктів інтеграції даних;
- продукти - це готові комерційні рішення, що підтримують одну або більше технології інтеграції даних;
- технології реалізують один або більше методів інтеграції даних;
- методи - це підходи до інтеграції даних, незалежні від технологій.

Існує три основні методи інтеграції даних: консолідація, федералізація і розповсюдження (рис. 2.2).

Консолідація даних. При використанні цього методу дані збираються з декількох первинних систем і інтегруються в одне постійне місце зберігання. Таке місце зберігання може бути використане для підготовки звітності і проведення аналізу, як у випадку застосування сховища даних, або як джерело даних для інших додатків, як у випадку впровадження операційного складу даних. При використанні цього методу зазвичай існує деяка затримка між моментом оновлення інформації в первинних системах і часом, коли ці зміни з'являються в кінцевому місці зберігання. Залежно від потреб бізнесу таке відставання може складати декілька секунд, годин або багато днів. Термін «*режим, наближений до реального часу»* часто використовується для опису кінцевих даних, оновлення яких відстає від джерела на декілька секунд, хвилин або годин. Дані, що не відстають від джерела, вважаються даними «*в режимі реального часу»*, але це важко досягнути при використанні методу консолідації даних.

Перевагою консолідації даних є те, що цей підхід дозволяє здійснювати трансформацію значних об'ємів даних (реструктуризацію, узгодження, очищення і/або агрегацію) в процесі їх передачі від первинних систем до кінцевих місць зберігання. Деякі складнощі, пов'язані з даним підходом, - це

значні обчислювальні ресурси, які потрібні для підтримки процесу консолідації даних, а також істотні ресурси пам'яті, необхідні для підтримки кінцевого місця зберігання. Але з урахуванням постійного вдосконалення апаратних засобів це не проблема.

Консолідація даних - це основний підхід, який використовується програмними додатками сховищ даних для побудови і підтримки оперативних складів даних і корпоративних сховищ. Консолідація даних також може знайти застосування для створення залежної вітрини даних, але в цьому випадку в процесі консолідації використовується тільки одне джерело даних (наприклад, корпоративне сховище). У середовищі сховищ даних однієї з найпоширеніших технологій підтримки консолідації є технологія ETL (витягання, перетворення і завантаження - extract, transform, and load). Ще одна поширена технологія консолідації даних - управління змістом корпорації (Enterprise Content Management - ECM). Більшість рішень ECM направлені на консолідацію і управління неструктурованими даними, такими як документи, звіти і web-сторінки.

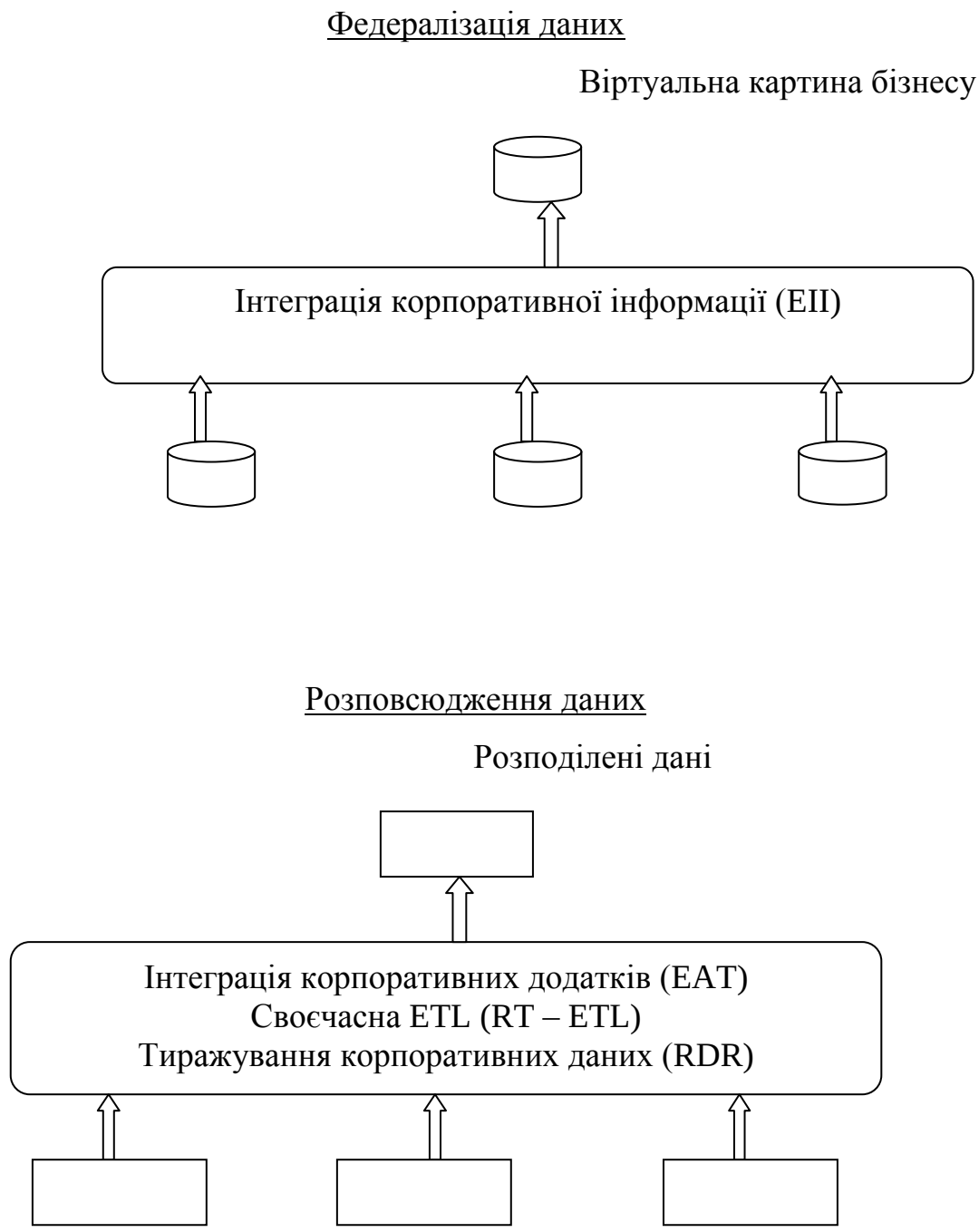


Рис. 2.2. Методи інтеграції даних.

Федералізація даних. Процес забезпечує єдину віртуальну картину одного або декількох первинних файлів даних. Якщо бізнес-додаток генерує запит до цієї віртуальної картини, то процесор федералізації даних витягує дані з відповідних первинних складів даних, інтегрує їх так, щоб вони відповідали віртуальній картині і вимогам запиту, і відправляє результати бізнес-додатку від якого прийшов запит. За визначенням, процес федералізації даних завжди

полягає у *витяганні* даних з первинних систем на підставі зовнішніх вимог. Всі необхідні перетворення даних здійснюються при їх витяганні з первинних файлів. Інтеграція корпоративної інформації (Enterprise Information Integration - EII) - це приклад технології, яка підтримує федеральний підхід до інтеграції даних. Один з ключових елементів федеральної системи - це метадані, які використовуються процесором федералізації даних для доступу до первинних даних. В деяких випадках ці метадані можуть складатися виключно з визначень віртуальної картини, які ставляться у відповідність («меппіруются») первинним файлам. У більш передових рішеннях метадані також можуть містити детальну інформацію про кількість даних, що знаходяться в первинних системах, а також про шляхи доступу до них. Така розширена інформація може допомогти федеральному рішенню оптимізувати доступ до первинних систем.

Вважається, що основна перевага федерального підходу - той факт, що він забезпечує доступ до поточних даних і позбавляє від необхідності консолідувати первинні дані в новому сховищі даних. Але слід пам'ятати, що федералізація даних не дуже добре підходить для витягання і узгодження великих масивів даних або для тих додатків, де існують серйозні проблеми з якістю даних в первинних системах. Ще один істотний чинник - потенційний вплив на продуктивність і додаткові витрати на доступ до численних джерел даних під час виконання програми.

Федеральна архітектура дуже корисна для крупних транснаціональних корпорацій і є вельми зручним підходом для підтримки балансу між необхідністю автономії місцевих підрозділів компанії і їх гнучкості, з одного боку, і стандартизації і централізованого контролю, які здійснює центральний офіс, - з іншою. При цьому федеральним сховищем може бути як єдине фізичне федеральне сховище, так і федерація дрібніших спеціалізованих сховищ даних.

Розповсюдження даних. Додатки розповсюдження даних здійснюють копіювання даних з одного місця в інше. Ці додатки зазвичай працюють в оперативному режимі і проводять переміщення даних до місць призначення, тобто залежать від певних подій. Оновлення в первинній системі можуть

передаватися в кінцеву систему синхронно або асинхронно. Синхронна передача вимагає, щоб оновлення в обох системах відбувалися під час однієї і тієї ж фізичної транзакції. Незалежно від використовуваного типу синхронізації, метод розповсюдження гарантує доставку даних в систему призначення. Така гарантія - це ключова відмітна ознака розповсюдження даних. Більшість технологій синхронного розповсюдження даних підтримують двосторонній обмін даними між первинними і кінцевими системами. Прикладами технологій, що підтримують розповсюдження даних, є інтеграція корпоративних додатків (Enterprise Application Ntegration - EAI) і тиражування корпоративних даних (Enterprise Data Replication - EDR).

Великою перевагою методу розповсюдження даних є те, що він може бути використаний для переміщення даних в режимі реального часу або близькому до нього. Інші достоїнства включають гарантовану доставку даних і двостороннє розповсюдження даних. Метод розповсюдження даних може також використовуватися для урівноваження робочого навантаження, створення резервних копій і відновлення даних, зокрема у разі надзвичайних ситуацій. Практичне застосування цього методу відрізняється чималою різноманітністю як в плані продуктивності, так і відносно можливостей реструктуризації і очищення даних. Деякі корпоративні продукти розповсюдження даних можуть підтримувати переміщення і реструктуризацію крупних масивів даних, тоді як продукти EAI часто мають обмежені можливості пересування великої кількості даних і їх реструктуризації. Одна з причин подібної відмінності - той факт, що в центрі архітектури тиражування корпоративних даних лежать дані, а в центрі технології EAI - повідомлення або транзакції.

Гібридний підхід. Методи, які використовуються додатками інтеграції даних, залежать як від потреб бізнесу, так і від технологічних вимог. Достатньо часто додаток інтеграції даних використовує так званий гібридний підхід, який включає декілька методів інтеграції. Хороший приклад такого підходу - інтеграція даних про клієнтів (Customer Data Ntegration - CDI), метою якої є

забезпечення узгодженої картини інформації про клієнтів. Найпростіший підхід до CDI - це створення консолідованого сховища даних про клієнтів, який містить дані, отримані з первинних систем. Відставання інформації в консолідованому сховищі залежатиме від режиму консолідації даних (оперативний або пакетний) і від частоти оновлення цієї інформації. Інший підхід до CDI - це федералізація даних, коли визначаються віртуальні бізнес-представлення даних про клієнтів в первинних системах. Ці уявлення використовуються бізнес-додатками для доступу до поточної інформації про клієнтів в первинних системах. При федеральному підході також може використовуватися довідковий файл метаданих для зв'язку інформації про клієнтів на основі загальних ключових елементів.

Гібридний підхід, що використовує як консолідацію, так і федералізацію даних, також може мати місце. Загальні дані про клієнтів (ім'я, адреса і т.д.) можуть бути консолідовані в одному сховищі, а дані, які відносяться до певного первинного додатку (наприклад, замовлення), можуть бути федералізовані. Такий гібридний підхід може бути розширений за рахунок розповсюдження даних. Якщо клієнт оновлює своє ім'я і адресу під час транзакції в Інтернет-магазині, то ці зміни можуть бути відправлені до консолідованого сховища даних, а звідти поширені в інші первинні системи, такі як база даних про клієнтів роздрібного магазина.

На сьогоднішній день існує два основні підходи до архітектури сховищ даних. Це так звана корпоративна інформаційна фабрика (Corporate Information Factory - CIF) Білла Інмона і сховище даних з архітектурою шини (Data Warehouse Bus - BUS) Ральфа Кимболла. Розглянемо кожний з них докладніше.

Corporate Information Factory. На рис. 2.3 представлений підхід, використовуваний в сховищах даних з архітектурою CIF. Колись цей підхід був відомий під назвою корпоративного сховища даних (Enterprise Data Warehouse - EDW). Робота такого сховища починається з скоординованого витягання даних з джерел. Після цього завантажується реляційна база даних з третьою нормальною формою, що містить атомарні дані. Отримане нормалізоване

сховище використовується для того, щоб наповнити інформацією додаткові репозиторії презентаційних даних, тобто даних, підготовлених для аналізу. Ці репозиторії, зокрема, включають спеціалізовані сховища для вивчення і «здобичі» даних (Data Mining), а також вітрини даних [16, 71].

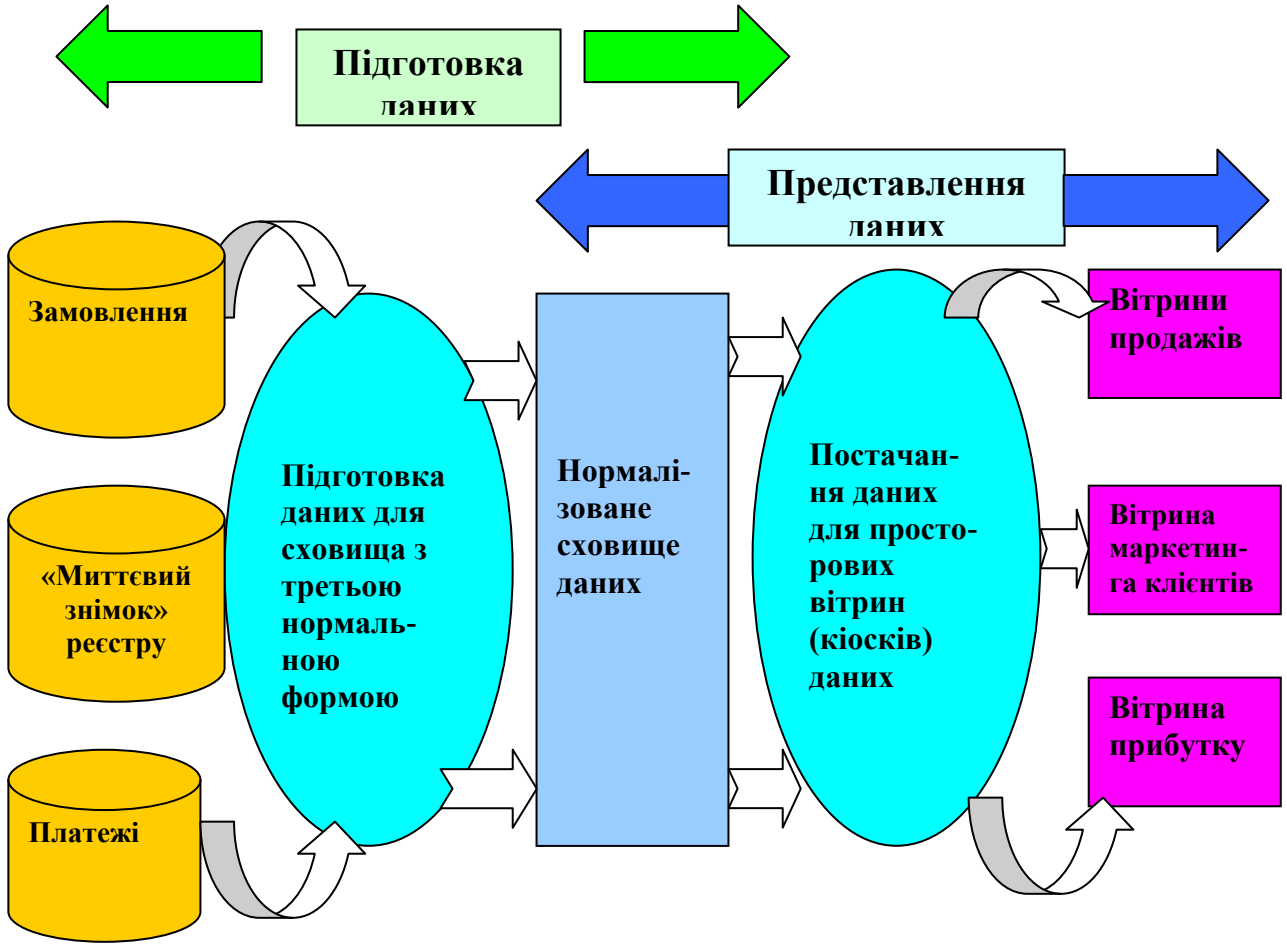


Рис. 2.3. Нормалізоване сховище даних з просторовими вітринами підсумкових даних (CIF).

При такому сценарії кінцеві вітрини даних створюються для обслуговування бізнес-відділів або для реалізації бізнес-функцій і використовують просторові моделі для структуризації сумарних даних. Атомарні дані залишаються доступними через нормалізоване сховище даних. Очевидно, що структура атомарних і сумарних даних при такому підході

істотно розрізняється. В якості відмітних характеристик підходу Білл Інмона до архітектури сховищ даних можна назвати наступні:

- використання реляційної моделі організації атомарних даних і просторовою - для організації сумарних даних;
- використання ітеративного або «спірального» підходу при створенні великих сховищ даних, тобто «будівництво» сховища не відразу, а по частинах. Це дозволяє при необхідності вносити зміни в невеликі блоки даних або програмних кодів і позбавляє від необхідності перепрограмувати значні об'єми даних в сховищі. Те ж саме можна сказати і про потенційні помилки: вони також будуть локалізовані в межах порівняльного невеликого масиву без ризику зіпсувати все сховище;
- використання третьої нормальної форми для організації атомарних даних, що забезпечує високий ступінь детальної інтегрованих даних і, відповідно, надає корпораціям широкі можливості для маніпулювання ними і зміни формату і способу представлення даних в міру необхідності;
- сховище даних - це проект корпоративного масштабу, що охоплює всі відділи і обслуговує потреби всіх користувачів корпорації.
- сховище даних - це не механічна колекція вітрин даних, а фізично цілісний об'єкт.

Data Warehouse Bus. Рисунок. 2.4. представляє альтернативний підхід до архітектури сховищ даних, відомий як сховище з архітектурою шини або підхід Ральфа Кимболла [62, 71].

У цій моделі первинні дані перетворяться в інформацію, придатну для використання, на етапі підготовки даних. При цьому обов'язково приймаються до уваги вимоги до швидкості обробки інформації і якості даних. Як і в моделі Білла Інмона, підготовка даних починається з скоординованого витягання даних з джерел. Ряд операцій здійснюється централізовано, наприклад, підтримка і зберігання загальних довідкових даних, інші дії можуть бути розподіленими. Область уявлення просторово структурована, при цьому вона може бути централізованою або розподіленою.

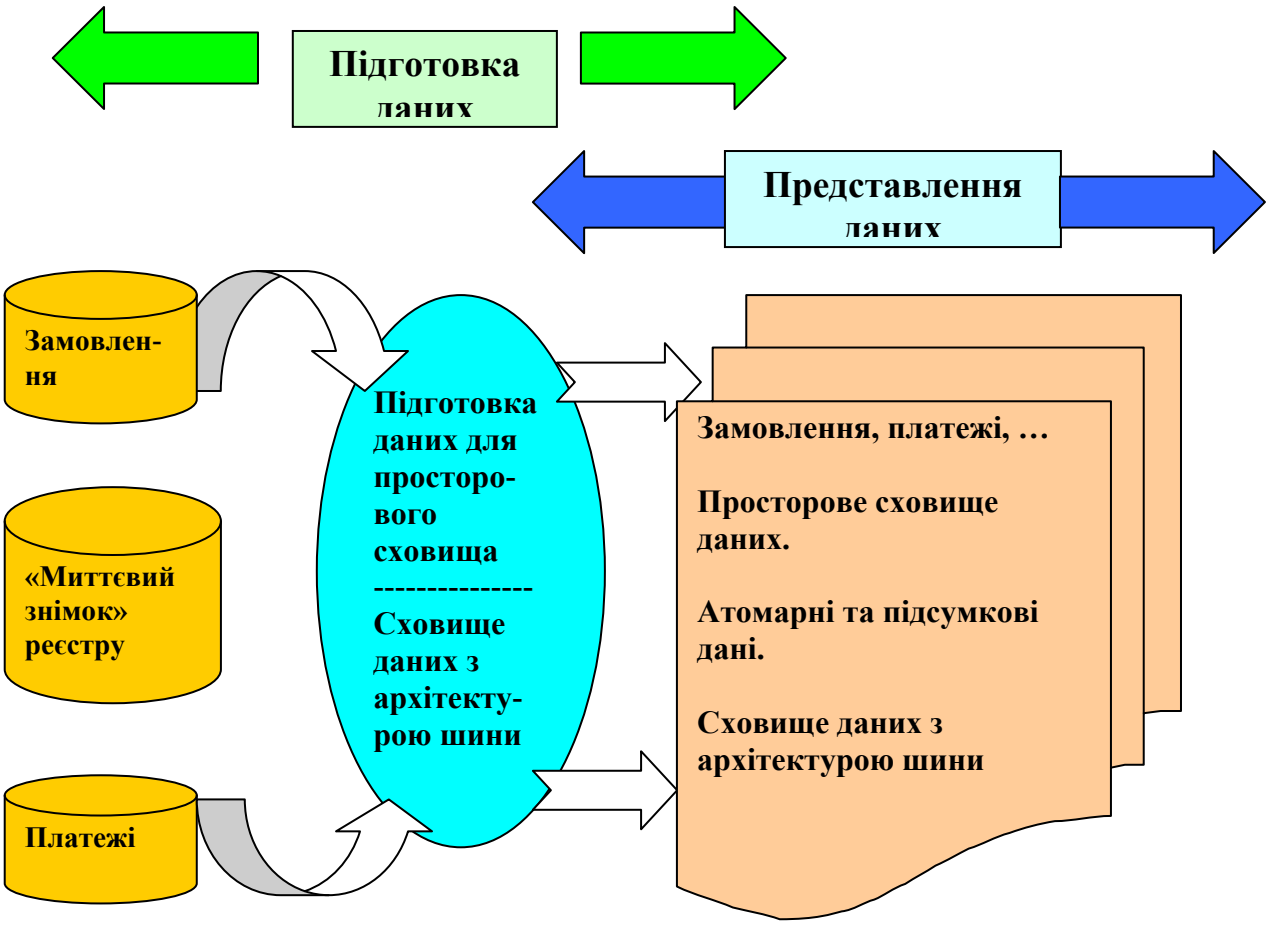


Рис. 2.4. Просторове сховище даних.

Просторова модель сховища даних містить ту ж атомарну інформацію, що і нормалізована модель, але інформація структурована по-іншому, щоб полегшити її використання і виконання запитів. Ця модель включає як атомарні дані, так і загальну інформацію (агрегати в зв'язаних таблицях або багатовимірних кубах) відповідно до вимог продуктивності або просторового розподілу даних. Запити в процесі виконання звертаються до все більш низького рівня деталізації без додаткового перепрограмування з боку користувачів або розробників додатку.

На відміну від підходу Білла Інмона, просторові моделі будуються для обслуговування бізнес-процесів (які, у свою чергу, пов'язані з бізнес-показниками), а не бізнес-відділа. Наприклад, дані про замовлення, які повинні бути доступні для загально корпоративного використання, вносяться до просторового сховища даних тільки один раз, на відміну від CIF-підходу, в

якому їх довелося б тричі копіювати у вітрини даних відділів маркетингу, продажів і фінансів. Після того, як в сховищі з'являється інформація про основні бізнес-процеси, консолідовані просторові моделі можуть видавати їх перехресні характеристики. Матриця корпоративного сховища даних з архітектурою шини виявляє і підсилює зв'язки між показниками бізнес-процесів (фактами) і описовими атрибутами (вимірюваннями).

Підсумовуючи все вищесказане, можна відзначити типові риси підходу Ральфа Кимболла:

- використання просторової моделі організації даних з архітектурою «зірка» (star scheme);
- використання дворівневої архітектури, яка включає стадію підготовки даних, недоступну для кінцевих користувачів, і сховище даних з архітектурою шини як таке. До складу останнього входять декілька вітрин атомарних даних, декілька вітрин агрегованих даних і персональна вітрина даних, але воно не містить одного фізично цілісного або централізованого сховища даних;
- сховище даних з архітектурою шини володіє наступними характеристиками: воно просторове, воно включає як дані про транзакції, так і сумарні дані, воно включає вітрини даних, присвячені тільки одній предметній області або що мають тільки одну таблицю фактів (fact table), воно може містити безліч вітрин даних в межах однієї бази даних;
- сховище даних не є єдиним фізичним репозиторієм (на відміну від підходу Білла Інмона). Це «віртуальне» сховище. Це колекція вітрин даних, кожна з яких має архітектуру типу «зірка».

До теперішнього часу технології сховищ даних досягли рівня зрілості - і як дисципліна, і як ринок технологій. Попит на них високий як ніколи. Про це, наприклад, свідчить той факт, що в 2007 р. дані засоби вперше потрапили в десятку головних пріоритетів директорів по інформаційним технологіям (згідно опиту компанії Gartner Group Inc.). Багато корпорацій вже володіють інфраструктурою сховищ даних і зараз займаються її удосконаленням і вирішенням проблем, з якими не вдалося справитися на стадії впровадження.

Частина корпорацій також прагнуть досягти наступних рівнів в розвитку цих засобів і акцентують увагу на 10 основних тенденціях, які сьогодні впливають на підхід корпорацій до впровадження сховищ даних. Ці тенденції включають як ті поліпшення, які компанії вносять до своєї стратегії і інфраструктури, так і нові технології і ініціативи, сприяючі подальшому розвитку сховищ даних [71].

Тенденція № 1: серйозне відношення до якості даних. Мало хто займатиметься вирішенням проблеми якості даних тільки ради якості як такого. Так що ж примушує корпорації приймати конкретні заходи по поліпшенню якості даних, а не просто говорити про це? По-перше, низька якість даних коштує гроші в тому сенсі, що веде до зниження продуктивності, ухвалення неправильних бізнес-рішень і неможливості отримати бажаний результат, не дивлячись на істотні інвестиції в корпоративні додатки. По-друге, низька якість даних може утруднити виконання вимог законодавства. За прогнозом компанії META Group, щорічне зростання ринку програмного забезпечення і послуг у сфері якості даних складатиме 20-30% аж до 2010 р. Цей прогноз співпадає з вже існуючими тенденціями: компанії дійсно мають намір робити конкретні дії для вирішення проблем якості даних. Враховуючи ту різноманітність засобів, яка зараз існує на ринку, для будь-якої компанії дуже важливо правильно вибрати такі методологію і інструменти, які дозволять практично підійти до вирішення цих проблем.

Тенденція № 2: стандартизація і консолідація інфраструктури. Корпорації стали усвідомлювати проблему роз'єднаності своїх рішень у сфері сховищ даних зовсім недавно, буквально пару років назад. По-перше, виявилося, що усунення дублюючих один одного інструментів сховищ даних або вітрин даних може понизити витрати на ліцензії і підтримку. Крім того, поліпшення доступу до інформації - це також істотна вигода від усунення незалежних структур даних, хоча вона і не піддається прямим кількісним оцінкам. Але здійснити реальну стандартизацію і консолідацію інфраструктури сховищ даних набагато складніше, ніж просто декларувати такий намір. Ця

діяльність включає політичні і організаційні аспекти, які такі ж проблематичні, як і технологічна сторона процесу.

Тенденція № 3: використання зарубіжних трудових ресурсів в області сховищ даних. На відміну від американських компаній, тенденція перенесення виконання проектів в області сховищ даних за межі країни з метою економії засобів мало актуальна для України, де заробітна плата і так менше, ніж в розвинених країнах. Тому тут ми детально не зупинятимемося на цій тенденції, відзначимо лише два основні моменти.

1. Така тенденція дійсно існує в США протягом декількох останніх років і активно обговорюється бізнес-аналітиками.

2. Перші результати цього процесу достатньо суперечливі. Разом з економією засобів компанії стикаються і з труднощами, пов'язаними з якістю роботи, комунікаційними проблемами і т.д. У результаті компанії далеко не завжди отримують ту економію засобів, на яку вони розраховували.

Тенденція № 4: стратегічний підхід до інформації. Поволі, але вірно корпорації починають розглядати інформацію як стратегічну складову бізнесу. Поки мало хто утілив ідею «інформація як капітал» в практику, але в багатьох організаціях вже є співробітники, які усвідомлюють стратегічну цінність інформації. Менеджери вищої ланки також сталі прихильно відносяться до цієї ідеї. Доказом даної тенденції є той факт, що сховища даних стають важливою частиною інших проектів з великими перспективами для бізнесу. Компанії можуть не впроваджувати систем інтелектуального аналізу даних в масштабах всієї корпорації, але вони використовують прийоми таких систем і сховища даних в ключових корпоративних проектах, які направлені на оптимізацію бізнес-процесів і зниження витрат.

Отже, з чого ж починається відношення до інформації як до капіталу? Перший крок - це розробка інформаційної стратегії і архітектури, а також впровадження основоположних стандартів управління даними. Це вимагає участі як IT-відділу, так і менеджерів, оскільки тут однаково важливі і технологічні, і організаційні і політичні аспекти. IT-фахівці повинні працювати

разом з менеджерами, щоб зрозуміти вимоги останніх і з'ясувати, які саме дані потрібні для вирішення основних питань бізнесу.

Тенденція № 5: виконання вимог законодавства як двигун розвитку сховищ даних. Для виконання законодавчих вимог необхідний доступ до даних. Як результат усвідомлення цього, в США зросли інвестиції в рішення, що надають доступ до інформації (або що забезпечують безпеку даних) і гарантують її точність. Необхідність виконання вимог законодавства стала основним стимулом для проектів, пов'язаних з сховищами даних, позначивши реальні негативні наслідки відсутності доступу до якісних даних. Багато корпорацій можуть випробовувати спокусу проігнорувати сховища даних і звернутися до своїх систем планування ресурсів підприємства (Enterprise Resource Planning - ERP) і фінансових систем, але прозорість і можливості контролю бізнесу, які надають рішення систем аналізу даних і сховищ даних, залишаються поза конкуренцією.

Тенденція № 6: подальший розвиток проблеми інтеграції корпоративних даних. Для передових корпорацій вже не стоїть питання, які технології вибрати для інтеграції даних, - вони зробили свій вибір. Багато хто вибрав технології витягання, перетворення і завантаження даних (Extraction, Transmission, Loading - ETL) як можливий спосіб для інтеграції корпоративних даних. Тепер перед цими корпораціями встає інше питання: як використовувати ті інструменти ETL, які вони вибрали. Таким чином, проблема інтеграції даних піднімається на наступний рівень. Для повноцінної інтеграції даних організаціям необхідно здійснювати управління метаданими і довідковими даними, а також забезпечити якість даних. Фактично, їм потрібний ще один інтегруючий засіб, який може зв'язати воєдино вибрані ними інструменти і збільшити вартість кінцевої продукції компанії. В той же час, багато корпорацій все ще знаходяться в процесі пошуку відповідних інструментів для інтеграції даних, вибираючи між ETL-технологіями, інтеграцією корпоративних додатків (Enterprise Application Integration - EAI) і web-сервісами. Але треба мати на увазі, що межі між цими технологіями стають все

більш розпливчатими: у ETL-технологіях з'являються можливості інструментів EAI для роботи в режимі реального часу, а в засобах EAI розширюються можливості для перетворення даних. Навіть для тих корпорацій, які все ще не визначилися з вибором інтеграційних технологій, буде корисно заглянути вперед і подумати про ті кроки, які доведеться зробити для впровадження вибраної технології і її успішного використання.

Тенденція № 7: навчання кінцевих користувачів. Рішення сховищ даних виявляться незатребуваними, якщо кінцевим користувачам не пояснити, які дані тепер їм доступні, і не переконати їх, що ці дані точні і повноцінні. Багато організацій вважають, що буде досить навчити кінцевих користувачів загальним прийомам роботи з цими інструментами. Але, на думку відомої аналітичної корпорації Gartner Group Inc., «набагато важливіше навчити користувачів інтелектуальному аналізу даних». Користувач, навчений роботі з інструментами, але що не знає, як їх використовувати в контексті конкретного середовища сховища даних, не зможе отримати бажані аналітичні результати. Відповідно, такий користувач або звернеться в IT-відділ, або взагалі відмовиться від використання даних засобів. Недостатнє визнання користувачами і усвідомлення компаніями цінності попередніх проектів сховищ даних примушують багато організацій визнати значущість комплексних навчальних програм для кінцевих користувачів.

Тенденція № 8: управління довідковими даними. У кожній корпорації є набір даних, які забезпечують важливу інформацію, необхідну для ідентифікації і конкретного визначення ключових об'єктів, таких як споживачі, продукція, постачальники і т.д. Ці дані називаються довідковими даними (master data), і зараз вони стають головною турботою для все більшого числа організацій. Управління довідковими даними, на перший погляд, виглядає простим завданням. Дійсно, що може бути складного для організації ідентифікувати своїх споживачів або визначити кожен вид своєї продукції? Як і у багатьох випадках, що відносяться до сфери інтелектуального аналізу даних і сховищ даних, те, що здається зовсім простим, в реальності може виявитися

вельми складним. Розповсюдження корпоративних додатків укупі з переважанням в багатьох організаціях ізольованих структур даних призводить до того, що довідкові дані виявляються розсіяними по всій корпорації. Різні області бізнесу можуть по-різному визначати і ідентифікувати такі об'єкти, як «споживач» або «продукт», і, можливо, зберігати їх довідкові дані в роз'єднаних базах даних. Таким чином, тенденція до інтеграції і оптимальної організації корпоративних систем зробила управління довідковими даними пріоритетним завданням. Але треба мати на увазі, що будь-яке технічне рішення, яке буде вибрано для цього завдання, повинне не тільки включати конкретні управлінські інструменти, але і приймати до уваги організаційні і політичні аспекти, якось: хто є «власником» цих довідкових даних і хто повинен визначати їх.

Тенденція № 9: поява нових учасників ринку сховищ даних. Зростання ринку сховищ даних не пройшло непоміченим серед постачальників в інших областях технології. Серед нових учасників цього ринку є як невеликі компанії, що починають, так і вже достатньо відомі фірми, але найбільш могутні з них - це постачальники систем планування ресурсів підприємства (ERP системи) і систем управління відносинами з клієнтами (Customer Relationship Management - CRM), наприклад, такі американські компанії, як SAP, Oracle і Siebel. Ці компанії бачать великі можливості на ринку сховищ даних, по-перше, оскільки неухильно зростає оборот цього ринку, а по-друге, тому що багато корпорацій виявили, що їх ERP і CRM системи функціонують не зовсім так, як очікувалося, в сенсі можливостей доступу до даних і їх аналізу. Постачальники ERP і CRM систем приходять на цей ринок, пропонуючи своїм клієнтам засоби інтелектуального аналізу даних і сховищ даних, щоб допомогти їм в отриманні необхідних даних, і статистика показує, що ці постачальники досягають успіху на новому терені.

Тенденція № 10: використання сховищ даних як робочого інструменту. Компанії усвідомлюють стратегічну роль сховищ даних, але вони також хочуть використовувати інформацію, отриману за допомогою своїх даних, для того,

щоб ухвалювати і тактичні рішення. Наприклад, роздрібні компанії зацікавлені в тому, щоб зрозуміти, як вони можуть використовувати інформацію, отриману з ланцюга постачань, для ухвалення своєчасних рішень. Таким чином, сховища даних можуть зіграти роль як в оперативному функціонуванні корпорації, так і у визначенні стратегічного напряму її розвитку.

2.3. Вітрини та кіоски даних

У найбільш загальному виді сховища даних можуть бути розбиті на два типи: корпоративні сховища даних (Enterprise Data Warehouses) і кіоски або вітрини даних (Data Marts) [9, 37, 49, 51].

Корпоративні сховища даних містять інформацію, що відноситься до всієї корпорації і зібрану з безлічі оперативних джерел для консолідованого аналізу. Зазвичай такі сховища охоплюють цілий ряд аспектів діяльності корпорації і використовуються для ухвалення як тактичних, так і стратегічних рішень. Корпоративне сховище містить детальну і узагальнену інформацію, його об'єм може досягати від 50 Гбайт до одного або декількох терабайт. Вартість створення і підтримки корпоративних сховищ може бути дуже високою. Зазвичай їх створенням займаються централізовані відділи інформаційних технологій, причому створюються вони зверху вниз, тобто спочатку проектується загальна схема, і тільки тоді починається заповнення даними. Такий процес може займати декілька років.

Кіоски або вітрини даних містять підмножину корпоративних даних і будуються для відділів або підрозділів усередині організації (рис. 2.5). Кіоски даних часто будуються силами самого відділу і охоплюють конкретний аспект, що цікавить співробітників даного відділу. Кіоск даних може отримувати дані з корпоративного сховища (залежний кіоск) або, що поширеніше, дані можуть поступати безпосередньо з оперативних джерел (незалежний кіоск). Кіоски і

сховища даних будуються за схожими принципами і використовують практично одні і ті ж технології.

Багато компаній, що усвідомлюють необхідність розробки корпоративного сховища даних, все ж таки не в силах справитися зі всіма завданнями виділення, стандартизації і об'єднання терабайт даних. Натомість вони вважають за краще будувати кіоски (або вітрини) даних (Data Marts) - спеціалізовані сховища даних, присвячені тільки одному напряму діяльності організації. Кіоск (вітрина) даних - це, найчастіше, найбільш керований різновид сховища даних. Його безперечний недолік полягає в тому, що без сховища даних, яке охоплювало б інформацію всього підприємства, неможливо порівнювати і аналізувати дані по всіх відділах і процесах. У багатьох компаніях вже зрозуміли, що кіоски (вітрини) даних можуть послужити хорошу службу і навіть стати єдино можливим рішенням для виконання термінових аналітичних завдань, але створення спеціалізованих кіосків без попередньої розробки корпоративної інфраструктури сховища даних може згодом привести до великих утруднень.

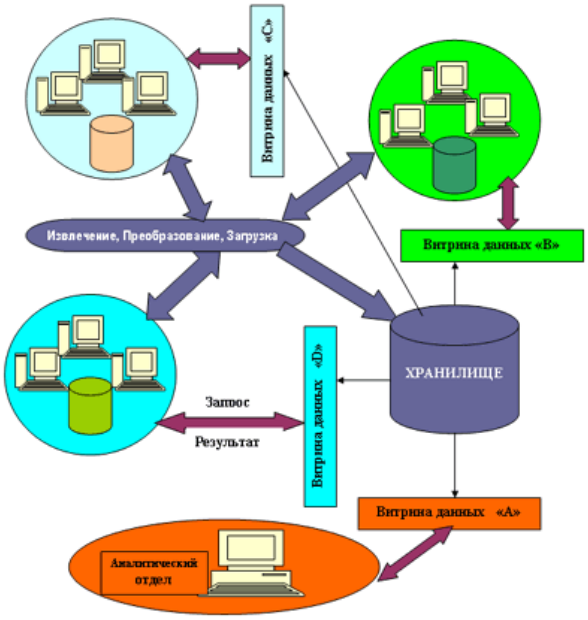


Рис. 2.5. Кіоски (вітрини) даних організації.

За класичним визначенням, вітрина (кіоск) даних (Data Mart) є підмножиною сховища даних, що відображає специфіку підрозділу (бізнес-об'єкт) і що забезпечує підвищену продуктивність. Таким чином, вітрина є ланкою, на якій базується конкретна аналітична система для вирішення свого кола завдань. Проте можлива ситуація, коли деяка область діяльності підприємства практично не корелює з іншими, і можливо побудувати відповідну вітрину даних автономно, без прив'язки до корпоративного сховища. Тоді вітрина поповнюватиметься даними безпосередньо з оперативних систем обробки транзакцій. Такі вітрини даних отримали назву незалежних, на відміну від класичних залежних від сховища даних і поповнюваних з нього вітрин.

У ряді випадків представляється доцільним розвернути вітрину (кіоск) даних замість повністю сформованого сховища. Вітрини даних накладають менші зобов'язання, вони дешевше і простіше в побудові і базуються на дешевших серверах, а не на мультипроцесорних комплексах. При такому підході немає необхідності задіювати цілу інформаційну систему корпорації і підтримувати складні процедури синхронного оновлення вітрини даних при оновленні сховища. В той же час необхідний розуміти, що при такому підході вітрини даних можуть розмножитися в цілі комплекси незалежних інформаційних баз даних, і природно буде поставлено завдання управління індивідуальними стратегіями пошуку, обслуговування і відновлення. З іншого боку, будувати єдине корпоративне сховище на основі множини незалежних вітрин даних значно вигідніше, ніж спираючись на розсіяні по системах обробки транзакцій дані.

Так що ж доцільно застосовувати: єдине сховище, самостійні вітрини (кіоски) даних, сховище із залежними вітринами або інші варіанти? Універсальної відповіді на питання про необхідність застосування того або іншого варіанту не існує. В кожному випадку оптимальний варіант визначається вимогами бізнесу, інтенсивністю запитів, мережевою архітектурою, необхідною швидкістю реакції і іншими умовами.

Розглянемо аргументи «За» і «Проти» використання вітрин (кіосків) даних відносно використання сховищ даних (таб. 2.2).

Таблиця 2.2.

Порівняльні аргументи застосування вітрин (кіосків) та сховищ даних.

	Аргументи «За»
Вартість	Створення навіть декількох вітрин (кіосків) даних обходиться значно дешевше, ніж організація єдиного сховища даних
Терміни	На опис предметної області, взаємних зв'язків між даними, організацію сховища даних і розробку механізмів його поповнення може піти декілька років, тоді як опис якого-небудь одного напряму діяльності підприємства без урахування різних зв'язків і з невеликим числом джерел надходження інформації займе менше часу.
Розміри	Оскільки вітрини (кіоски) даних зазвичай містять лише дані по певному кругу питань і, отже, займають менше місця і вимагають менше технічних ресурсів, то для них менш гостро коштує питання апаратної платформи і вартості устаткування.
Безпека	З вітринами (кіосками) даних зазвичай працює менше число користувачів, чим зі сховищем даних. З'являється можливість контролю прав не тільки на рівні окремих таблиць і записів, а і на рівні доступу до всього додатку, що надійніше.
	Аргументи «Проти»
Дублювання даних	Різні вітрини даних можуть містити одну і ту ж інформацію, якщо цього вимагають їх предметні області. Природно, дублювання інформації ставить перед користувачами і адміністраторами проблему синхронізації даних (тобто їх порівняння і уніфікації).
Розширення	Хоча виробники і стверджують, що вітрини (кіоски) даних поступово можна наростити до рівня сховищ даних (як модульні системи), в це погано віриться. Інакше вони б

	коштували стільки ж, скільки і самі сховища. Представляється, що процес об'єднання незалежних (логічно і фізично) вітрин даних вельми трудомісткий.
Обмеженість	Вітрини (кіоски) даних задумані як склади даних, що містять інформацію по якій-небудь одній темі. Для великих компаній з широким колом вирішуваних задач і різноманітними інтересами, вигідніше мати повноцінне сховище даних, оскільки воно зможе вміщати всі необхідні для їх життєдіяльності відомості.

Розповідаючи про триваючі протягом півроку дослідження (з питання про переважний вибір між кіосками даних і сховищами даних), аналітик Кевін Стрендж, керівник досліджень по програмних структурах компанії Gartner Group, відзначив, що кіоски не обов'язково мають малий розмір, деколи вони вимагають рішення складних адміністративних задач, і причому далеко не завжди обходяться дешевше за сховища даних [37, 51]. Кіоски (вітрини) даних створюються спеціально під конкретний програмний додаток і вирішують специфічні задачі, скажемо проводять аналіз, необхідний для роботи служби технічної підтримки користувачів, тоді як крупні сховища даних не залежать від додатку. Але кіоски даних можуть зберігати сотні гігабайт інформації і не обмежуються загальним стандартом в 50 Гбайт.

Кіоски добре підходять компаніям, які вимушені підтримувати системи підтримки ухвалення рішення і не володіють достатнім досвідом для розробки повномасштабного сховища даних. По словах Стренджа, організації, що вже реалізували кіоски даних, поступово починають об'єднувати їх, і без сумніву до 2010 року більшість організацій, які зараз створюють кіоски, прагнутимуть групувати їх в крупніші структури.

Компанія Gartner Group також прийшла до висновку, що орієнтація сховищ даних на Web обмежувалася додатками для Intranet і не була

розрахована на використання Internet із-за уразливості захисту і недостатнього рівня продуктивності.

Прийоми моделювання кіосків (вітрин) даних відрізняються від прийомів моделювання сховищ даних через різні вимоги до структур даних. Якщо основною задачею сховища даних є зберігання консолідованої історичної інформації, то вітрина даних будується з урахуванням вимог по доступу до даних і представлення інформації. Як правило, для моделювання вітрин (кіосків) даних використовуються типи моделі під назвою: схема "зірка" і схема "сніжинка". Зупинимося докладніше на кожному з цих типів моделей.

Схема "зірка" - популярний тип моделі даних для вітрин даних. Дана модель характеризується наявністю таблиці фактів, оточеної пов'язаними з нею таблицями розмірностей. Запити до такої структури включають прості об'єднання таблиці фактів з кожною з таблиць розмірностей. Характеризується високою продуктивністю запитів. Проектується для виконання аналітичних запитів. Характеризується невеликою надмірністю даних і високою в порівнянні з нормалізованими структурами продуктивністю. Деякі промислові СУБД і інструменти класу OLAP/Reporting уміють використовувати переваги схеми "зірка" для скорочення часу виконання запитів. На рисунку 2.6 зображений приклад схеми "зірка" для аналізу залишків на рахунках і кількості транзакцій в розрізі часу, клієнтів, рахунків, продуктів, відділень, статусів рахунків. Дана модель дозволяє відповісти на широкий спектр аналітичних питань.

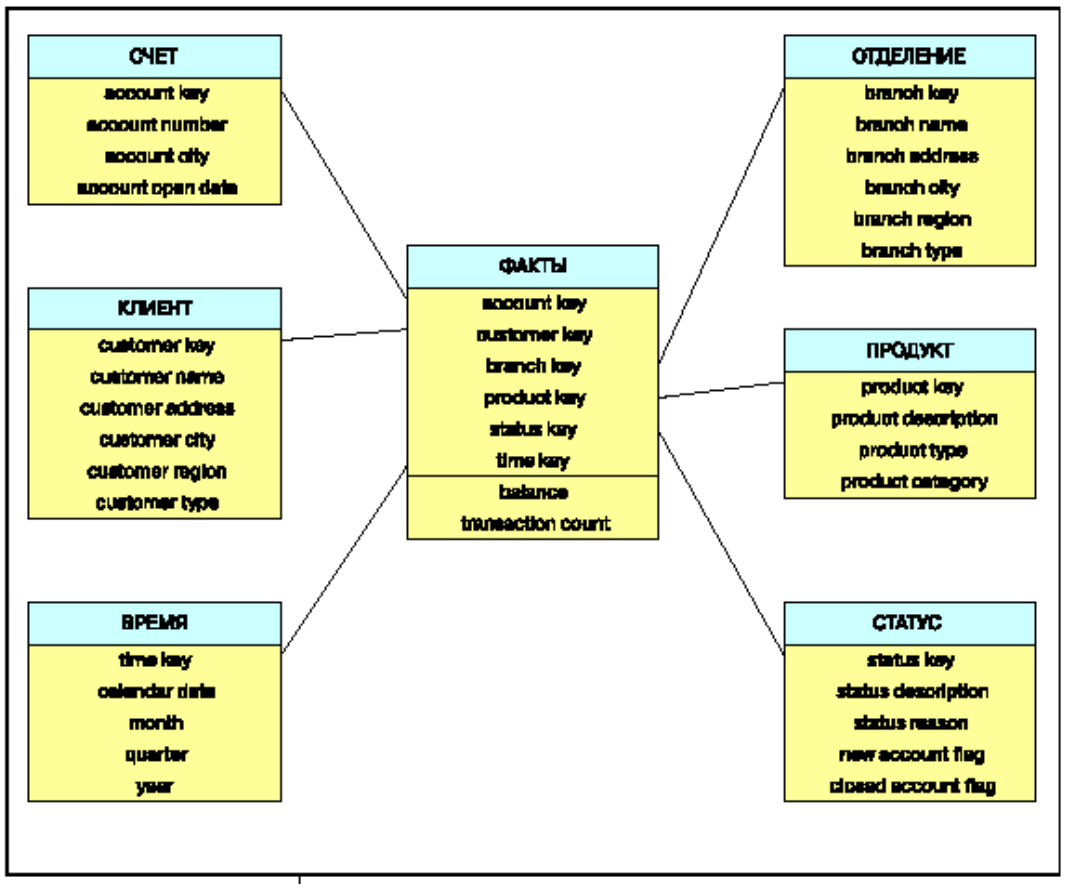


Рис. 2.6. Приклад схеми "зірка" для аналізу залишків на рахунках.

Розглянемо компоненти схеми "зірка".

Розмірності. У технології багатовимірного моделювання розмірність - це аспект, в розрізі якого можна отримувати, фільтрувати, групувати і відображати інформацію про факти. Типові розмірності, що зустрічаються практично в будь-якій моделі:

- Клієнт
- Продукт
- Час
- Географія
- Співробітник

Розмірності, як правило, мають багаторівневу ієрархічну структуру. Наприклад, розмірність ЧАС може мати наступну структуру: РІК – КВАРТАЛ - МІСЯЦЬ - ДЕНЬ.

Факти. Факти - це зазвичай числові величини, що зберігаються в таблиці фактів і є предметом аналізу. Приклади фактів: об'єм операцій, кількість проданих одиниць товару і так далі. Факти мають ряд властивостей, на яких ми коротко зупинимося.

1. Адитивні факти. Адитивність визначає можливість підсумування факту уздовж певної розмірності. Такі факти можна підсумовувати і групувати уздовж всієї розмірності на будь-яких рівнях ієрархії.

2. Напівадитивні факти. Напівадитивний факт — це факт, який можна підсумовувати уздовж певної розмірності, і не можна — уздовж інших. Прикладом може служити залишок на рахунку (або залишок товару на складі). Дану величину не можна підсумовувати уздовж розмірності ЧАС. Проте сума залишків по рахунках уздовж розмірності є предметом для аналізу.

Фахівці рекомендують моделювати напівадитивні факти так, щоб зробити їх більш адитивними. Наприклад, представити відсоток складовими його величинами.

3. Неадитивні факти. Неадитивні факти взагалі не можна підсумовувати. Приклад неадитивного факту — відношення (наприклад, виражене у відсотках).

Таблиці покриття. Таблиці покриття використовуються з метою моделювания поєднання розмірностей, для яких відсутні факти. Наприклад, потрібно знайти кількість категорій продуктів, які сьогодні жодного разу не продавалися. Таблиця фактів продажів не може відповісти на дане питання, оскільки вона реєструє лише факти продажів. Для того, щоб модель дозволяла відповідати на подібні питання, потрібна додаткова таблиця фактів (яка, по суті справи, не містить фактів).

Схема "сніжинка" використовується для нормалізації схеми "зірка" (рис. 2.7). Вона декілька скорочує надмірність в таблицях розмірності. Одним з достоїнств є швидше виконання запитів про структуру розмірності (запити вигляду «Вибрати всі рядки з таблиці розмірності на певному рівні»), які дуже часто виконуються при аналізі даних, і можуть затримувати хід аналізу. Проте основною відмінністю схеми "сніжинка" є не економія дискового простору, а

можливість мати таблиці фактів з різним рівнем деталізації. Наприклад, фактичні дані на рівні дня, а планові — на рівні місяця.

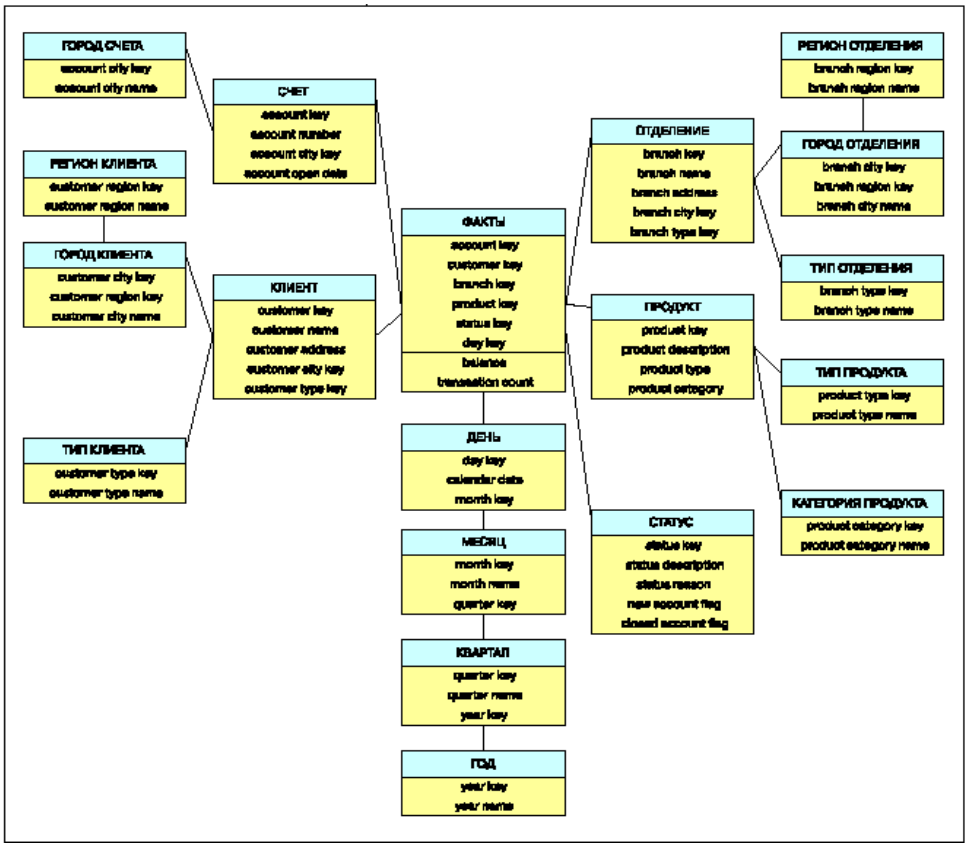


Рис. 2.7. Приклад схеми "сніжинка" для аналізу залишків на рахунках.

Методика побудови вітрин (кіосків) даних з простої теоретичної дисципліни поступово перетворюється на складну науку, повну варіацій і напрямів. Якщо раніше було відомо лише про EDW (Enterprise Data Warehouse), то тепер з’явилися поступово розвиваємі вітрини даних (Incremental Architected Data Mart, ADM), розподілені вітрини (кіоски) даних (Distributed Data Mart, DDM), об'єднані вітрини даних (Federated Data Mart, FDM). Розглянемо деякі з цих нових напрямків.

Системи об'єднаних вітрин даних. У багатьох організаціях склалася практика реалізації багаточисельних сховищ даних. Хоча, за визначенням, існує лише одне сховище даних, а всі останні об'єкти є його підмножиною або вітринами (кіосками) даних, що поступово розвиваються, але не всі організації дотримуються цього правила. Таким чином, в багатьох компаніях існує два,

три, десяток і навіть більше систем сховищ даних. Поширення сховищ даних привело до розвитку архітектури сховища даних підприємства, а саме: до появи об'єднаних систем сховищ даних або вітрин (кіосків) даних.

Система об'єднаних вітрин даних характеризується спільним використанням загальних інформаційних ресурсів, усуваючи, таким чином, надмірність і гарантуючи достовірність інформації по всій організації (рис. 2. 8).

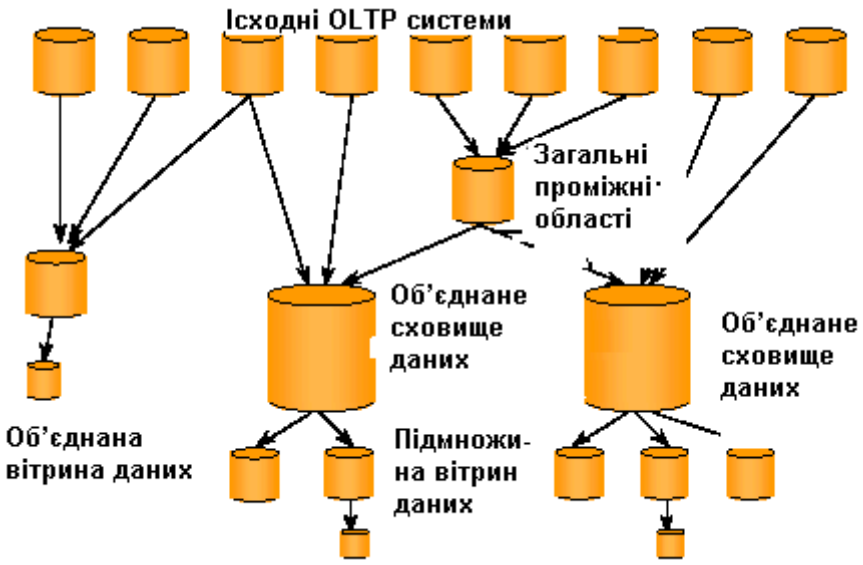


Рис. 2.8. Система об'єднаних вітрин даних.

Позитивними рисами об'єднаних вітрин даних є: загальна семантика бізнес-правила; один набір процесів витягання і бізнес-правил; децентралізовані ресурси і управління; паралельна розробка.

Недоліками такого архітектурного рішення є: необхідність в координуванні робіт; складнощі в подоланні "політичних" моментів і вирішенні питань авторських прав; потрібна узгодженість серед різних відділів по питаннях архітектури, бізнес-правил і семантики; складне технічне середовище; наявність багаточисельних репозиторіїв метаданих.

Непроектуємі вітрини даних. Поява непроектуємих вітрин даних (Non-Architected Data Marts) пояснюється, перш за все, складнощами, пов'язаними з реалізацією систем EDW і FDW. Брудні і швидко отримувані набори даних не піддаються очищенню і, отже, не можуть використовуватися для подальшої

інтеграції з будь-якими іншими джерелами даних систем сховищ даних. Дуже швидко вони перетворюються на застарілі системи, які лише додають проблем, а не вирішують їх. Для цих систем характерні багаточисельні процеси витягання, безліч бізнес-правил, невірогідність інформації (рис. 2.9).

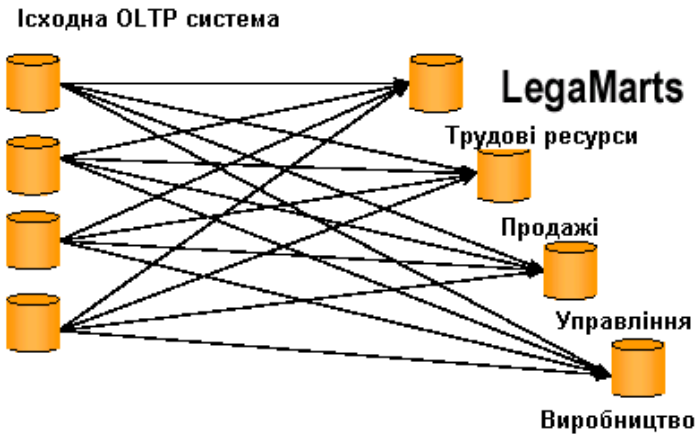


Рис. 2.9. Система непроектуємих вітрин даних (LegaMart).

Позитивними рисами непроектуємих вітрин даних є: висока продуктивність; низька вартість. Недоліками таких систем є: недостовірна інформація; багаточисельні процеси витягання; багаточисельні бізнес-правила; підвищена складність при інтеграції.

Система вітрин (кіосків) даних, що поступово розвиваються. Ця архітектура є альтернативою сховища даних підприємства. Для наповнення таких вітрин зазвичай використовується інструментальний засіб класу підприємства, що реалізовує стратегію «витягаєш один раз, наповнюєш багато» (рис. 2.10).

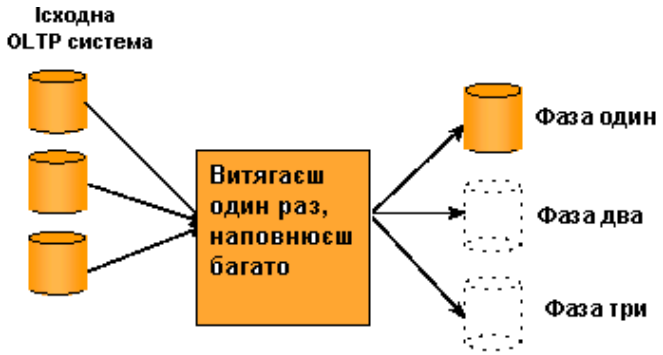


Рис. 2.10. Система поступово розвиваємих вітрин даних.

Достоїнствами таких вітрин даних є: єдиний набір процесів витягання; здійснимий масштаб. Недоліки: найбільш ефективні при використанні інструментального засобу класу підприємства; необхідність в архітектурі вітрин даних підприємства (Enterprise Data Mart Architecture, EDMA).

Методика побудови вітрин (кіосків) даних виявилася напрямом ринку проектів інтелектуального аналізу даних, що нестримно розвивається, швидко змінюється. Якщо раніше не було механізмів їх ефективного проектування, і був лише один способів їх створення, в даний час можна знайти незчисленне число таких інструментів і ряд технологій життєздатної архітектури таких систем. За умови вибору відповідної архітектури і належного підходу до проекту можна побудувати систему сховища та вітрин даних, яка забезпечить не лише високе повернення інвестицій, але і значно підвищить ефективність функціонування всього підприємства [37, 70].

Декілька фірм пропонує системи побудови вітрин даних: Informatica (PowerMart Suite), Sagent Technology (Data Mart Solution) і Oracle (DataMart Suite). Для ілюстрації процесу розробки вітрини даних можна розглянути коротко склад і функціональність пакету DataMart Suite [43].

Пакет включає п'ять основних компонентів: Data Mart Designer, Data Mart Builder, Oracle7 Enterprize Server, Web Server і Discoverer 3.0. Data Mart Designer дозволяє описувати структуру вітрини і запам'ятовувати її в репозітарії. На виході Data Mart Designer породжує опис на мові DDL SQL, який потім подається на вхід Oracle7 Enterprize Server. В результаті створюється структура бази даних, що реалізовує вітрину даних. В ході побудови вітрини користувач може застосовувати існуючі описи структур або будувати вітрину «з нуля». Крім того, Data Mart Designer дозволяє будувати додатки для Oracle Web Server на базі PL/SQL. Data Mart Builder витягує дані із зовнішніх джерел і заповнює вітрину. Він володіє наочним спеціалізованим інтерфейсом, що відображає потоки даних при заповненні сховища. Data Mart Builder здатний витягувати дані з реляційних СУБД і CSV-файлів. Web Server

надає відкриту платформу для розробки Web-додатків. Він включає Web Request Broker (WRB), що реалізований на основі технології картриджів і дозволяє розробляти Web-додатки, що вбудовуються в Web Server. Як засоби розробки можуть використовуватися Java, PL/SQL, LiveHTML, C і C++. Discoverer 3.0 - це засіб кінцевого користувача, що дозволяє генерувати звіти, а також виконувати деякі OLAP-операції з вітриною даних. Звіти, побудовані за допомогою Discoverer 3.0, можна експортувати у форматі HTML, роблячи їх доступними для web-браузеров. Discoverer 3.0 також дозволяє створювати і підтримувати таблиці агрегованих даних. Крім цього, DataMart Suite включає готовий додаток Sales Analyzer.

2.4. OLAP - технологія

Вміння швидко і головне правильно приймати рішення має в сучасному бізнесі принципове значення для досягнення успіху. Проте кількість інформації, що впливає на предмет рішення, інколи може бути просто-таки величезною. Як поступати в такій ситуації? Покластися на випадок або все ж таки взятися за повномасштабний аналіз? Досвідченний керівник незмінно вибере другий спосіб, тим більше що сьогодні існує ряд технологій, здатних спростити процес прийняття і моделювання рішень при великій кількості «вхідної» інформації [4, 9, 11, 66].

Власне, аби спростити роботу з багатоцільовими даними і не загрузнути в їх океані, а також уміло перетворити набір кількісних показників на якісні, і застосовується метод OLAP – On-Line Analytical Processing (оперативна аналітична обробка). Останній, на відміну від інших способів автоматизації бізнес-діяльності, дає можливість отримати користувачеві «на виході» не готове чітко структуроване рішення, що видається після включення раніше налагодженого майстра обробки форм, а своєрідний матеріал для наочної і, якщо можна так виразитися, творчої оцінки існуючої ситуації. Тому сфера

вживання OLAP-аналізу зазвичай обмежується менеджерським складом підприємств різних розмірів, якому доводиться часто займатися тактичними і стратегічними завданнями на зразок аналізу ключових показників діяльності і сценаріїв розвитку, маркетинговим і фінансово-економічним аналізом груп товарів або послуг, а також довгостроковим прогнозуванням роботи підприємства або його підрозділів. Для цього користувач OLAP-систем отримує в руки потужний і головне дуже гнучкий інструмент створення різних звітів по вибраних їм же розрізах і напрямах. При цьому методики OLAP більш досконаліша за звичні електронні таблиці, адже окрім простих функцій створення таблиць, графіків і діаграм, OLAP-системи дають можливість отримати узагальнені дані по самостійно вибраних критеріях, вмить поглибиться в деталі вибраних напрямів, відфільтрувати, сортувати або відкинути непотрібні цифри або показники. Наприклад, якщо менеджерові продажів компанії потрібно отримати сезонні зведення динаміки продажів вибраній категорії товарів, система запропонує йому всілякі дані про продажі за місяць, квартал, рік, а також знайде і проаналізує їх залежність від зазначених чинників, скажімо, часу проведення маркетингових акцій. Крім того, базуючись на одній лише статистиці продажів, OLAP-система може виявити ефективність роботи різних підрозділів компанії, у тому числі і в розрізі географічної ієрархії їх взаємодії. При цьому параметри, що характеризують успішність підрозділів, вибираються менеджером самостійно і у ряді випадків можуть стати інструментом мотивації успішного персоналу.

Щоб зрозуміти, як працюють OLAP-системи, досить звернутися до її механізмів. Найбільш показове поняття OLAP-технології - гіперкуб (метакуб), що є умоглядною фігурою в багатовимірному просторі, утвореному площинами даних, які важливі для діяльності підприємства. При цьому сама OLAP-система виступає саме в ролі гіперкуба, здатного накопичувати в собі всю інформацію, що цікавить керівника. Як ребра куба виступають різні категорії товарів або послуг просуваємих компанією. Наприклад, ціна виробленого або конкурентного товару, компанії-учасники виробничого циклу, підрядчики при

організації послуг, об'єми продажів, географія самої компанії. Важливо відзначити, що градація різних осей квадрантів куба може мати різну структуру, а крім того, самі осі можуть бути взаємозалежними. Так, вісь часу може бути розбита по роках, кварталах, тижнях, а вісь доходів або шкідливих викидів при виробництві - прологарифмована. Інформація, необхідна для аналізу в даний момент, вирізається з гіперкуба перетином площин даних, що використовуються при аналізі, немов його шар або частина. При цьому розрізи можуть проходити як через весь куб, так і обмежуватися певними рамками і межами осей.

Технічно системи оперативного аналізу даних зазвичай функціонують у зв'язці з сховищами та вітринами даних, а клієнтські OLAP-системи встановлюються на будь-яких призначених для користувача комп'ютерах корпоративної інформаційної системи. Рідше OLAP-модулі взаємодіють з іншими системами автоматизації, адже бази даних останніх досить часто мають вельми своєрідний вигляд і набір спеціальних показників. Втім, для сучасного українського підприємства характерна нетипова ситуація, коли є декілька систем автоматизації (для вирішення різних завдань) і, як наслідок, дані зберігаються розрізнено, в результаті відсутній єдиний погляд на управлінську інформацію. Тому в процесі складання звіту беруть участь два фахівці – програміст, що забезпечує запити до баз даних, і економіст, що намагається за допомогою електронних таблиць звести ці дані в звіт, необхідний керівництву. Як показує практика, подібна модель взаємодії користувача звіту (керівника) і самих даних незмінно приводить до ефекту «зіпсованого телефону», не говорячи вже про істотні витрати часу. І з даної точки зору використання OLAP-систем також представляється вельми раціональним, адже використання декількох інформаційних систем незмінно приводить до «надлишку» даних, які можуть бути впорядковані OLAP-системою.

У чому ж відмінність OLAP-системи від сховища даних? З точки зору користувача, відповідь на це питання досить проста: у мірі предметної структурованості інформації. Працюючи з OLAP-додатком, користувач

застосовує звичні економічні категорії і показники – види матеріалів і готової продукції, регіони продажів, об'єм реалізації, собівартість, прибуток і тому подібне. А для того, щоб сформувати будь-який, навіть досить складний запит, користувачеві не доведеться вивчати SQL. При цьому відповідь на запит буде отримана протягом всього декількох секунд. Крім того, працюючи з OLAP-системою, економіст може користуватися такими звичними для себе інструментами, як електронні таблиці або спеціальні засоби побудови звітів. Таким чином, якщо сховище даних – в основному об'єкт уваги спеціаліста по інформаційним технологіям, то OLAP без перебільшення можна назвати програмним засобом з арсеналу економіста. Адже саме економіст має справу з самими різними аналітичними задачами: маркетинговим аналізом, аналізом продажів, аналізом бюджетних показників, аналізом фінансової звітності і так далі [17, 49, 71].

Цікаво, що розшифровка абревіатури OLAP – «система оперативного аналізу даних» – не повністю відповідає реальному призначенню і методу роботи цих систем. Отримана назва є наслідком того, що OLAP-модулі оперують багатовимірними моделями, якими, до речі, мислить і людина. Тому OLAP, так само як і людський мозок, вимагає вичерпної інформації по самих різних напрямах і навряд чи розкриє весь свій потенціал при недоліку даних. В будь-якому разі «океан даних» якраз те середовище, де використання інструментів OLAP найбільш виправдано. Слід розуміти, що OLAP - це не окремо взятий програмний продукт, не мова програмування і навіть не конкретна технологія. Якщо постаратися охопити OLAP у всіх його проявах, то можна стверджувати, що це є сукупність концепцій, принципів і вимог, які лежать в основі програмних продуктів, що полегшують аналітикам доступ до даних.

Розглянемо деякі сценарії практичного застосування OLAP-продуктів.

Аналіз фінансових показників діяльності підприємства. Бухгалтерські системи 1С, БЕСТ, Парус, Інфін, RS-Balance та інші день за днем нагромаджують результати обліку господарської діяльності підприємств. Вони

забезпечують розрахунок фінансових показників і випуск звітності для наглядових органів. Проте, фіскальна звітність не призначена для управління організацією. Керівника цікавить динаміка залишків і рух фінансів, структура доходів і їх розподіл по клієнтах, товарах, днях тижня, місяцях, кварталах, за рік і так далі. Аби забезпечити керівників управлінськими звітами, OLAP-система налаштовується на базу даних будь-якої облікової системи. У додатку описуються запити на здобуття і розрахунок нових фінансових показників, таблиці і графіки звітів для різних користувачів: фінансових директорів, співробітників планово-економічних служб, бухгалтерів. Важливий поточний звіт можна роздрукувати для обговорення з товаришами по службі або зберегти у вигляді мікрокуба і відіслати директорові або інвесторам. В результаті дані фіскального обліку реально використовуються для управління організацією. Вони миттєво витягуються з бази даних бухгалтерської системи, користувач самостійно управляє формою звіту, отримує фінансові показники в різних розрізах.

Корпоративна звітність. У розподіленій організації філії регулярно передають дані в центральний офіс. Тут дані потрапляють в єдине сховище. Над ними виконуються додаткові розрахунки, для яких в філіях немає даних. Наприклад, загально корпоративні витрати розносяться на філії, зменшуючи тим самим їх прибуток в звіті про прибутки і збитки. Аби філії могли ознайомитися з остаточними звітами після виконання всіх розрахунків і перевірок, запускається генератор кубів, що є програмним модулем OLAP-системи. В результаті для кожної філії генерується окремий мікрокуб, який відправляється по e-mail. Одержувач - співробітник планово-економічного відділу філії - відкриває куб в OLAPBrowser, аналізує, роздруковує і підшиває звіти. Для керівника філією створюється новий, узагальнений мікрокуб. У ньому додатково побудовані діаграми, що наочно показують структуру доходів і витрат, динаміку продажів за останній квартал в розрізі товарів і тому подібне. Таку технологію, наприклад. використовує Укрсиббанк. За допомогою систем генерації і перегляду мікрокубів забезпечується інформаційна підтримка

співробітників 20 філій банку, розподілених по всій Україні. Мікрокуби створюються в головному банку за даними бухгалтерського і управлінського обліку, які інтегровані в єдиному фінансовому сховищі.

Аналіз бюджетних даних. Для ведення фінансового планування і обліку фактичного виконання бюджетів підприємства застосовують прикладні модулі у складі комплексних ERP- систем Галактика, БЕСТ та ін., спеціалізовані програмні комплекси, наприклад, Контур Корпорація, Бюджет холдингу, Інтальов, Бюджетне управління і тому подібне. У всіх випадках для аналізу бюджетних планів, контролю виконання бюджету і аналізу відхилень фактичних показників від планових застосовується OLAP- система (рис 2.11).

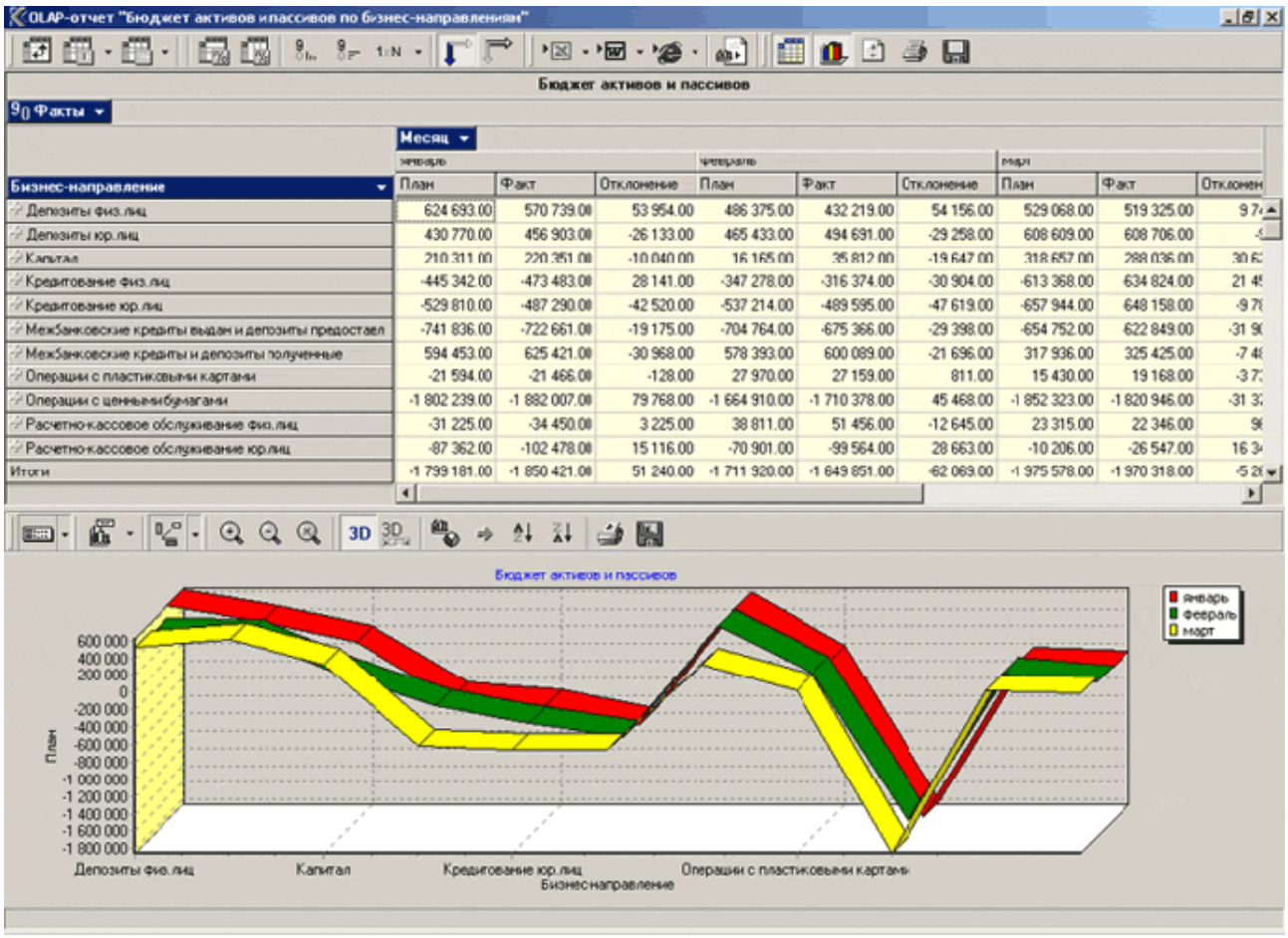


Рис. 2.11. Аналіз бюджетних даних.

Співробітники бюджетно-аналітичних і планово-економічних підрозділів випускають «план-факт» звіти про виконання бюджетів за місяць, квартал, рік,

аналізують бюджетні плани в розрізі центрів фінансової відповідальності і бізнес-напрямів, деталізують значення бюджетних статей. Наприклад, найбільший банк Казахстану «Банк ТуранАлем» випускає всю бюджетну звітність у вигляді OLAP-звітів: фінансовий план і бюджет доходів і витрат в розрізі центрів фінансової відповідальності, бізнес-напрямів, банківських продуктів і так далі.

Аналіз складських даних. Інформація про перебування і рух товарів на складі (товарні запаси, терміни зберігання товарів, постачальники і одержувачі продукції, накладні переміщення товарів) міститься в базі даних OLTP-модуля складського обліку. Аналіз цієї інформації дає відповіді на питання: «Скільки продукції було куплено замовником Івановим в третій декаді вересня?», «Який оптимальний об'єм активних і резервних запасів по даній товарній позиції?», «Чи існують сезонні коливання за даним типом товарів і яка їх амплітуда?» і тому подібне (рис. 2.12).

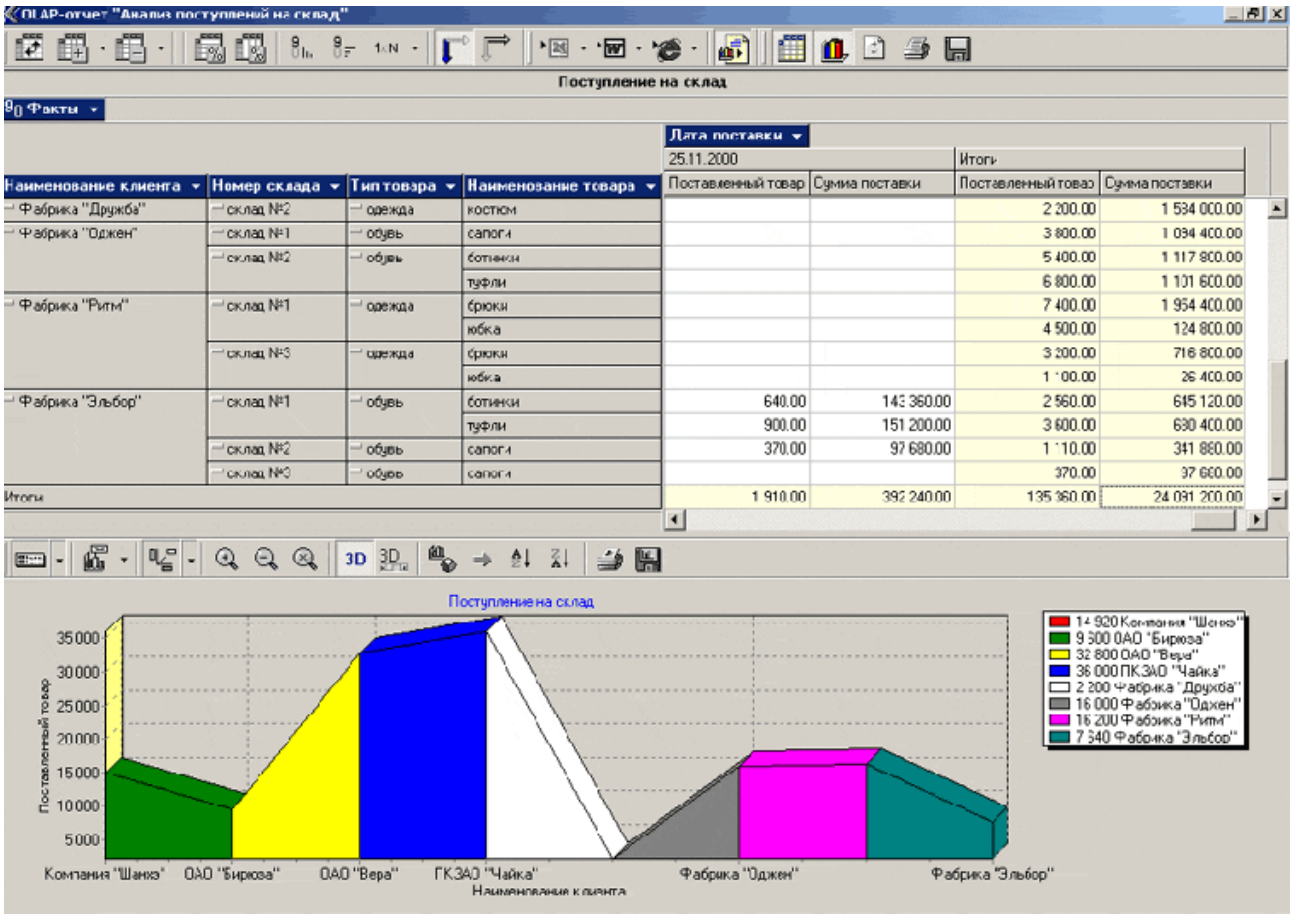


Рис.2.12. Аналіз перебування товару на складі.

Декілька операторів працюють з одним програмним додатком, який розміщується на доступному сервері і безпосередньо звертається до бази даних системи складського обліку. За допомогою OLAP-звітів вони оперативно контролюють поточну ситуацію: залишки товарних запасів в розрізі видів і партій товарів, термінів зберігання, стан відвантаження по одержувачах. Для віддалених підрозділів стан складу відбивається в мікрокубі, який розміщується на захищеній сторінці корпоративного веб-сайту.

Спільно з обліком складських даних вирішується і задача обліку транспортної логістики. Для цього до облікової системи складу підключаються довідники обслуговуючого склади транспорту, ведеться облік його використання по кожному транспортному засобу. Наприклад, у ВАТ «Трьохгорна мануфактура» аналітична система дозволяє контролювати відвантаження товарів з складів по артикулах.

Аналіз відвідуваності Web-сайту. Web-сайт є серйозним маркетинговим інструментом для багатьох компаній. Аналіз поведінки відвідувачів сайту дозволяє оцінити віддачу від маркетингових заходів і рекламних акцій, ефективність застосування on-line сервісів, інтерес до продуктів і послуг компанії і т.д. Для аналізу використовуються дані log-файлів веб-сервера, вивантажені в локальні або реляційні таблиці, або база даних сайту. Оскільки розміри таких баз, як правило, дуже великі, застосовується технологія мікрокубів. Генератор кубів щоночі за розкладом створює мікрокуб з актуальними показниками відвідуваності ресурсу. За його допомогою співробітники відділу маркетингу випускають звіти, які зберігається в локальній мережі компанії. Запити до мікрокуба виконуються практично з «нульовим» часом очікування, мережевий трафік значно нижчий за рахунок стиснення інформації в мікрокубі. Таке рішення, наприклад, застосовується в компанії «1С: Рарус» для аналізу використання співробітниками ресурсів Інтернет і аналізу дзвінків по мобільному зв'язку.

Публікація маркетингових досліджень. Маркетингові агентства збирають інформацію, обробляють її і продають результати зацікавленим

організаціям. Наприклад, якесь агентство досліджує ринок продуктів глибокого заморожування і продає свої щомісячні звіти виробникам. Якщо в агентстві немає спеціалізованого ПО, результати досліджень можуть зводитися в Excel-файл. Цей файл засобами MS Excel можна зберігати в dbf-таблиці, з якої, у свою чергу, легко створити мікрокуб. Він продається виробникам продуктів по абонементу. Маркетологи підприємств отримують мікрокуб поштою і аналізують ринок, використовуючи програму OlapBrowser. Наприклад, англійська консалтингова компанія Decision Tree Consulting виконує аналіз конкурентного середовища для брендів по замовленнях 200 найбільших компаній, таких як Sony, Toshiba, Panasonic, Nokia, Pioneer, Sanyo, Siemens, Phillips, Hewlett Packard. Результати досліджень поставляються замовникам у вигляді мікрокубів. Індійська консалтингова компанія MARC, яка обслуговує фармацевтичні компанії Індії, також поширює результати досліджень замовникам у вигляді мікрокубів, надаючи їм видалений доступ через Інтернет.

Створення інформаційного сервісу. Електронні біржі і інформаційні агентства публікують на своїх сайтах проспекти біржових індексів, котирування цінних паперів різних емітентів, рейтинги учасників фондового ринку за різними показниками і іншу інформацію у вигляді мікрокубів. Комерсанти знайомляться з актуальними даними з будь-якої точки земної кулі через Інтернет і з допомогою OlapBrowser проводять аналіз архівних і поточних біржових зведень і аналітичних довідок. Підтримка інформації в актуальному стані забезпечується за рахунок генерації мікрокубів за розкладом. Так транснаціональна інвестиційна корпорація Fidelity Investment поставляє своїм клієнтам інформацію про цінні папери у вигляді мікрокубів через Інтернет.

Світовий ринок OLAP-додатків стабільно зростав – на 16% в 2008 році, і по прогнозах – приблизно на стільки ж в 2009 – 2010 роках. У абсолютних цифрах об'єм світового ринку OLAP, за даними «The OLAP Report» [66, с. 7], в 2007 році складав 5,7 млрд. дол., і як очікується, досягне показника в 7 млрд. дол. до 2009 року. Потрібно відзначити комерційну привабливість цього сектора - адже він є в даний час одним з самих швидкозростаючих сегментів IT-

ринку. Проте так було не завжди – показник зростання дуже сильно вагався і демонстрував сильну залежність від стану галузі в цілому – від більш ніж 40% зростання (у 1995- 1998 роках) до близько 5% в 2001-2002, в період кризи доткомів в США. Є всі підстави вважати, що і надалі ринок OLAP буде так само сильний схильний до впливу загально галузевих коливань кон'юнктури IT-ринку (рис. 2.13).

Ринок OLAP сильно консолідований – на долю трьох найбільших виробників доводилося в 2008 році близько 63% ринку, а на долю найбільших десяти – ледве більше 97% ринку (рис. 2.14). Важливим також є той факт, що нинішній лідер ринку – Microsoft – єдина компанія, яка, за інформацією «The OLAP Report» [66, с.11], з року в рік демонструє зростання своєї долі на ринку (з 11,4% до 31,6% в 2008 р.).

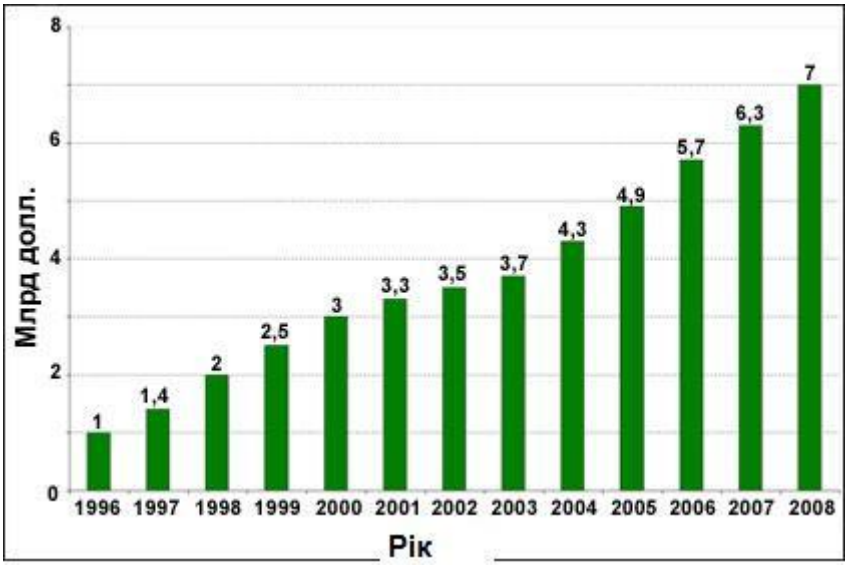


Рис. 2.13. Розвиток ринка OLAP–додатків.

Важливою тенденцією розвитку ринку OLAP-додатків є створення комплексних засобів аналізу даних, що включають як OLAP-засоби, так і інші засоби Data Mining. Поєднання OLAP з такими інструментами, як системи фільтрації і обробки даних, засоби автоматичної класифікації і кластеризації, нейронні мережі т.д., дозволяє створювати потужні аналітичні додатки з широким спектром можливого вживання. Вочевидь, саме такий універсальні додатки надалі домінуватимуть на ринку.

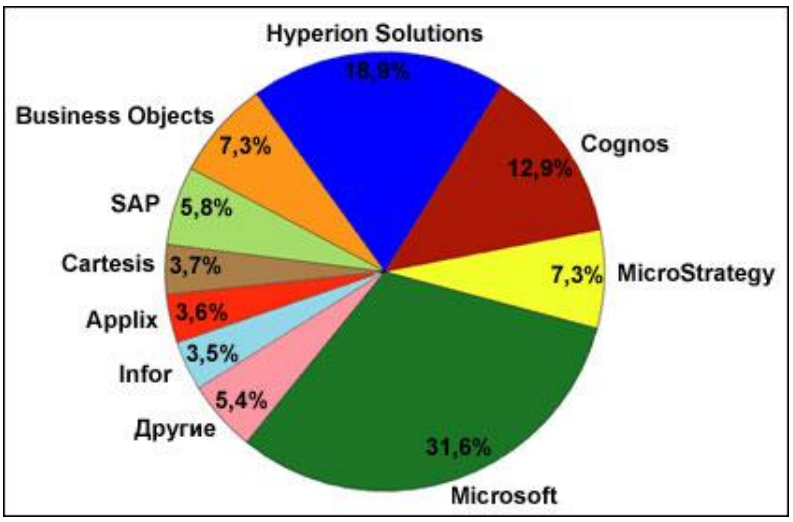


Рис. 2.14. Найбільші виробники OLAP–додатків.

Крім того, цікаво відзначити такий важливий напрям, як інтеграція OLAP-додатків з системами управління базами даних і ERP-системами. Цей напрям інтеграції активно розвиває Microsoft - Microsoft Analysis Services є додатком до Microsoft SQL Server, а рішення для управління підприємством – такі як Microsoft Dynamics AX (раніше - Microsoft Business Solutions Axapta) містять вбудовані OLAP-модулі. Схожої практики дотримуються і інші виробники СУБД і ERP- систем – Oracle і SAP. Вочевидь, що кількість і об'єм продажів інтегрованих OLAP-додатків з часом лише збільшуватиметься [11].

Одним з важливих чинників розвитку інтелектуального аналізу даних в сучасному інформаційному світі є рух Open Source. Все більше і більше компаній або використовують продукти з відкритими вихідними кодами, або самі беруть участь і підтримують розвиток таких програмних додатків. Майже для кожного типа програмних продуктів існують їх аналоги в сегменті Open Source, і OLAP-додатки не є виключенням. Вже зараз існує цілий набір додатків, що реалізовують всі функції бізнес-аналітики (такі як Pentaho Open BI Suite – повноцінна платформа для корпоративної системи аналізу даних, Palo-Server, і ін.). Головною їх перевагою можна вважати можливість змінити додаток так, щоб він найбільше підходив під конкретні умови роботи, замість того, щоб змінювати структуру ухвалення рішень, що склалася. Це може залучити крупні компанії, що використовують OLAP-додатки.

Також наслідком появи Open Source OLAP-додатків стане більше їх поширення – невеликі компанії, які до цього не застосовували дорогі OLAP-продукти із-за їх ціни, дістануть таку можливість. Внаслідок цього можна чекати появи ринку послуг з установки і підтримки Open Source OLAP-додатків, де основними споживачами будуть компанії малого і середнього бізнесу.

Ринок OLAP-додатків тісно пов'язаний з ринком консалтингових послуг. Навчання, впровадження, підтримка є природним доповненням до програмного забезпечення для аналізу даних. Вже зараз на українському ринку успішно діють компанії, які пропонують комплексні рішення по консалтингу, навчанню і впровадженню цілого спектру продуктів. Проте до того моменту, коли ринок OLAP неможливо буде відокремити від ринку консалтингових послуг, ще далеко. У довгостроковій перспективі компанії-виробники продовжуватимуть орієнтуватися на виробництво програмного забезпечення, залишаючи надання послуг партнерам.

Враховуючи вище перелічені чинники, що впливають на розвиток ринку OLAP-додатків, можна передбачити, що в середньостроковій перспективі ринок зростатиме досить швидкими темпами, але конкуренція при цьому скорочуватиметься із-за процесу злиття компаній, що продовжується. Все більше очікується комплексних рішень для аналізу даних і модулів для систем управління базами даних і ERP-систем, що реалізовують функціональність OLAP. Ціна на OLAP-додатки знижуватиметься – як із-за тиску з боку лідерів ринку, так і із-за поширення Open Source OLAP продуктів. В той же час потреба в послугах з впровадження і супроводу OLAP-систем збільшиться, що приведе до збільшення кількості спеціалізованих консалтингових компаній.

Ідея обробки багатовимірних даних не є новою. Фактично вона сходить до 1962 року, коли Кен Айверсон опублікував свою книгу "Мова програмування" ("A Programming Language" - APL). Перша практична реалізація APL була здійснена в кінці 60-х років компанією IBM. APL - це математична мова з багатовимірними змінними і витонченими, хоч і досить

абстрактними операторами. Вона призначалася більше для опису багатовимірних перетворень, ніж для використання як практичної мови програмування. Так, наприклад, в ній не приділялося уваги таким приземленим питанням, як робота з файлами або вивід на друк. У дуже стислій нотації мови використовувалися грецькі символи. Насправді, тексти програм виходили вельми компактними. Вона стала відомою, як «мова тільки для написання», тому що було набагато легше переписати наново програму, ніж виправити раніше збережений текст. APL поглинала машинні ресурси і вимагала великих витрат. Програми дуже поволі виконувалися і обходилися дуже дорого. Проте не дивлячись на невдалий початок, APL не була викинута. Вона використовувалася в багатьох ділових додатках 70-х, 80-х років, які функціонально подібні до сьогоднішніх OLAP систем. Так, IBM розробила операційну систему для APL, названу VSPC, і деякі люди вважали її ідеальним середовищем для персонального використання задовго до появи електронних таблиць. У 80-х роках APL став доступний на персональних машинах, але не знайшов ринкового застосування. Альтернативою було програмування багатовимірних додатків з використанням масивів в інших мовах. Це було дуже важким завданням навіть для професійних програмістів, так що кінцевим користувачам залишалося чекати наступного покоління багатовимірних програмних продуктів.

У 1970 році вперше з'явився прикладний багатовимірний програмний продукт, що використалися в навчальних цілях - Express. Він в повністю переписаному вигляді широко використовується в сучасних OLAP-додатках, проте оригінальні концепції 70-х років залишилися далеко позаду. Сьогодні, Express залишається однією з найбільш популярних OLAP-технологій, і компанії Oracle вдається підтримувати його на рівні сучасних вимог разом з багатьма новими продуктами з архітектурою «клієнт-сервер».

Більше багатовимірних продуктів з'явилися в 80-х роках. На початку десятиліття з'явився Stratagem, в новому обличчі - Acumate, який просувався на

ринку до середини 90-х, але сьогодні, на відміну від Express, використовується дуже обмежено.

Comshare System W був багатовимірним продуктом іншого стилю. Представлений в 1981 році, він першим використовував ідею гіперкуба і був більшою мірою орієнтований на кінцевого користувача в розробці фінансових додатків. Він привніс багато концепцій, які, правда, ще добре не пропрацювали, типу не процедурних правил, повноекранного перегляду багатовимірних даних, редагування даних, інтеграція з реляційними даними (у пакетному режимі). Проте Comshare System W був достатньо важкий для апаратного забезпечення того часу і менш програмованим в порівнянні з іншими продуктами і, відповідно, був менш популярний в середовищі професіоналів. Він також поки використовується, але продається все менше, оскільки не мав тих поліпшень, які очікувалися. В кінці 80-х Comshare випустив в середовищі DOS, а пізніше для Windows, продукт під назвою Commander Prism. Він використовував ті ж концепції, що були закладені в System W. Essbase, продукт компанії Hyperion Solution, хоч і не є прямим нащадком System W, був очевидно під впливом його рішень з своєю орієнтацією на фінансові додатки і організацією гіперкуба з повними попередніми обчисленнями.

Іншим новаторським продуктом на початку 80-х був Metaphor. Він призначався для професійних маркетологів. Цей пакет також запропонував багато нових концепцій, які стали популярними тільки в 90-х роках: такі як, клієнт - серверне обчислення, багатовимірна обробка реляційних даних, розрахований на багато користувачів режим і об'єктно орієнтована розробка додатків. На жаль стандартні персональні комп'ютери не забезпечували тих характеристик, які вимагав Metaphor, і постачальники були вимушені розробляти власні персональні машини і мережі. Надалі Metaphor став працювати вдало і на серійних персональних машинах, проте він ніколи не використав стандартний графічний інтерфейс користувача (GUI).

В кінці 80-х серед інструментів кінцевого користувача для аналізу даних стали домінувати електронні таблиці. Перша багатовимірна електронна

таблиця з'явилася у вигляді Compete. Він просувався на ринок як дуже дорогий продукт для фахівців, але постачальники не забезпечили можливість захоплення ринку цим продуктом, і компанія Computer Associates придбала права на нього разом з іншими продуктами класу «spreadsheet» (електронні таблиці), включаючи Supercalc і 20/20. Основним ефектом від придбання Compete компанією CA було різке зниження ціни і зняття захисту від копіювання, що, природно, сприяло його розповсюдженню. Проте він ще не був вдалим. Протягом декількох років Compete ще зрідка можна було зустріти у вигляді навантаження в деяких комплектах постачання. Пізніше Compete був покладений в основу Supercalc 5, але багатовимірний аспект його не просувався.

Компанія Lotus була наступною, хто спробував увійти на ринок багатовимірних електронних таблиць з продуктом Improv. Це гарантувало, що продажі 1-2-3 не знизяться, але коли той з часом був випущений під Windows, Excel вже став настільки серйозним конкурентом, що продажі Improv не внесли помітних змін в розподілі ринку. Lotus, подібно CA з Compete, скинула ціну на Improv, проте і цього було недостатньо для просування на ринку, і нові розробки в цій області не отримали продовження. Виявилось, що користувачі персональних комп'ютерів віддають перевагу електронним таблицям в оригінальній версії 1-2-3 і не цікавляться новими багатовимірними можливостями, якщо вони не повністю сумісні з їх старими таблицями. Так само концепція невеликих багатовимірних настільних електронних таблиць, пропонованих як продуктивний інструмент для персональних додатків, насправді не виявилися зручними і не прижилися в реальній практиці.

Компанія Microsoft пішла по цьому шляху, додавши PivotTables до Excel. Хоча небагато користувачів Excel отримали вигоду від використання цієї можливості, це, ймовірно, єдиний факт широкого використання можливостей багатовимірного аналізу просто тому, що в світі дуже багато користувачів Excel. Excel 2000 містить витонченішу версію PivotTables, призначену для використання і як настільний інструмент OLAP і як клієнтська частина для

взаємодії з Microsoft OLAP Services. Проте можливості OLAP в Excel 2000 не є базовими, ведучими, вони скоріше вбудовані як додаткова, другорядна можливість.

В кінці 80-х років фірма Sinper увійшла до світу багатовимірних електронних таблиць спочатку з власною електронною таблицею для DOS, а потім приєднаною до версії 1-2-3 для DOS. Перетворений на продукт TM/1, він увійшов до ери Windows як сервера баз даних у форматі Excel і 1-2-3. Трохи пізніше Arbor зробив аналогічну річ, хоча його новий Essbase міг працювати тільки в режимі клієнт-сервер, тоді як продукт фірми Sinper міг так само працювати на локальному комп'ютері. Цей підхід приніс мультирозмірність в електронні таблиці, які є такими популярними серед користувачів. Таким чином, традиційні постачальники власних інструментів представлення даних кінцевому користувачеві пішли по цьому шляху, і такі продукти, як DSS Server, Express, Holos, Gentia, Mineshare, Powerplay, Metacube і Whitelight тепер з гордістю пропонують високо інтегрований доступ до електронних таблиць в своїх серверах додатків. За іронією долі за свої перші шість місяців існування Microsoft OLAP Services був одним з декількох OLAP серверів, що не мають клієнтської частини у вигляді електронної таблиці. Пропозиції компанії Microsoft з'явилися тільки в червні 1999 року в Excel 2000. Проте OLAP@Work, вбудований в Excel, заповнив цей пропуск, і поки що має набагато кращі експлуатаційні характеристики, чим власний інтерфейс Excel компанії Microsoft.

Деякі користувачі вимагають для своїх багатовимірних додатків можливості обробки дуже великих багатовимірних баз даних. І реляційні OLAP-інструменти розвиваються в цьому напрямі, відгукуючись на ці потреби. Вони надають звичайні засоби проглядання багатовимірних даних, а іноді включають інтерфейс кінцевого користувача у вигляді електронної таблиці, навіть якщо всі дані зберігаються в реляційних СУБД. Такі засоби є дуже дорогими для користувача, вони менш продуктивні, чим спеціалізовані інструменти багатовимірного аналізу, але вони забезпечують цю, таку

популярну форму аналізу даних, навіть якщо останні зберігаються не у вигляді багатовимірних структур.

Інші поставщики розвивають те, що сьогодні називається настільним OLAP: невеликі куби, що генеруються з великих баз даних і потім завантажуються в персональний комп'ютер для обробки. Вони дійсно досягають великого успіху. А коли постачальник продає обидві можливості: і інструмент формування реляційних запитів і інструмент багатовимірного аналізу і формування звітів, то досягає більшого успіху у кінцевих користувачів, ніж в інших випадках.

2.5. Основні архітектури OLAP - систем

Системи інтелектуального аналізу даних зазвичай володіють засобами надання користувачеві агрегатних даних для різних вибірок з початкового набору в зручному для сприйняття і аналізу вигляді. Як правило, такі агрегатні функції утворюють багатовимірний (і, отже, не реляційний) набір даних (нерідко званий гіперкубом або метакубом), осі якого містять параметри, а ячейки - залежні від них агрегатні дані - причому зберігатися такі дані можуть і в реляційних таблицях, але в даному випадку ми говоримо про логічну організацію даних, а не про фізичну реалізацію їх зберігання. Уздовж кожної осі дані можуть бути організовані у вигляді ієрархії, що представляє різні рівні їх деталізації. Завдяки такій моделі даних користувачі можуть формулювати складні запити, генерувати звіти, отримувати підмножини даних [4, 9, 45, 71].

Технологія комплексного багатовимірного аналізу даних отримала назву OLAP (On-Line Analytical Processing). OLAP - це ключовий компонент організації сховищ даних. Концепція OLAP була описана в 1993 році Едгаром Коддом, відомим дослідником баз даних і автором реляційної моделі даних. У 1995 році на основі вимог, викладених Коддом, був сформульований так званий тест FASMI (Fast Analysis of Shared Multidimensional Information - швидкий

аналіз розподіленої багатовимірної інформації), що включає наступні вимоги до додатків для багатовимірного аналізу:

- Fast (Швидкий). Надання користувачеві результатів аналізу за прийнятний час (зазвичай не більше 5 с), нехай навіть ціною менш детального аналізу;
- Analysis (Аналіз). Можливість здійснення будь-якого логічного і статистичного аналізу, характерного для даного додатку, і його збереження в доступному для кінцевого користувача вигляді;
- Shared (Розподілений доступ). Розрахований на багато користувачів доступ до даних з підтримкою відповідних механізмів блокувань і засобів авторизованого доступу;
- Multidimensional (Багатовимірність). Багатовимірне концептуальне представлення даних, включаючи повну підтримку для ієрархій і множинних ієрархій (це - ключова вимога OLAP);
- Information (Інформація). Можливість звертатися до будь-якої потрібної інформації незалежно від її об'єму і місця зберігання.

Слід зазначити, що OLAP-функціональність може бути реалізована різними способами, починаючи з простих засобів аналізу даних в офісних додатках і закінчуючи розподіленими аналітичними системами, заснованими на серверних продуктах. Але перш ніж говорити про різні реалізації цієї функціональності, давайте розглянемо, що ж є куби OLAP з логічної точки зору. Як приклад реляційної бази даних, який ми використовуватимемо для ілюстрації принципів OLAP, скористаємося базою даних Northwind, що входить в комплекти постачання Microsoft SQL Server і є типовою базою даних, що зберігає дані про торгові операції компанії, що займається оптовими постачаннями продовольства. До таких даних відносяться відомості про постачальників, клієнтів, компанії, що здійснюють доставку, список товарів, що поставляються і їх категорій, дані про замовлення і замовлені товари, список співробітників компанії [11].

Для розгляду концепції OLAP скористаємося представленням Invoices і таблицями Products і Categories з бази даних Northwind, створивши запит, в результаті якого отримаємо докладні відомості про всі замовлені товари і виписані рахунки:

```
SELECT dbo.Invoices.Country,
dbo.Invoices.City,
dbo.Invoices.CustomerName,
dbo.Invoices.Salesperson,
dbo.Invoices.OrderDate,
dbo.Categories.CategoryName,
dbo.Invoices.ProductName,
dbo.Invoices.ShipperName,
dbo.Invoices.ExtendedPrice
FROM dbo.Products INNER JOIN
dbo.Categories ON dbo.Products.CategoryID = dbo.Categories.CategoryID INNER JOIN
dbo.Invoices ON dbo.Products.ProductID = dbo.Invoices.ProductID
```

Цей запит звертається до представлення Invoices, що містить відомості про всі виписані рахунки, а також до таблиць Categories і Products, що містить дані про категорії продуктів, які замовлялися, і про самі продукти відповідно. В результаті цього запиту ми отримаємо набір даних про замовлення, що включає категорію і найменування замовленого товару, дату розміщення замовлення, ім'я співробітника, що виписав рахунок, місто, країну і назву компанії-замовника, а також найменування компанії, що відповідає за доставку. Для зручності збережемо цей запит у вигляді уявлення, назвавши його Invoices1. Результат звернення до цього уявлення приведений на малюнку 2.15.

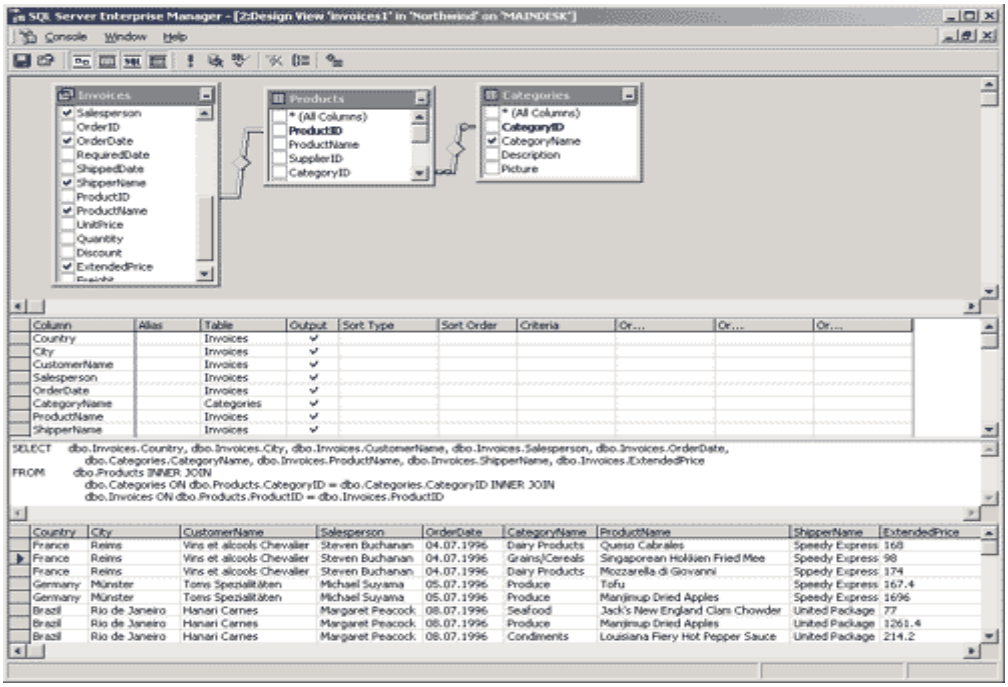


Рис. 2.15. Результат звернення до представлення Invoices 1.

Які агрегатні дані ми можемо отримати на основі цього уявлення? Звичайно це відповіді на питання типу:

- Яка сумарна вартість замовлень, зроблених клієнтами з Франції?
- Яка сумарна вартість замовлень, зроблених клієнтами з Франції і доставлених компанією Speedy Express?
- Яка сумарна вартість замовлень, зроблених клієнтами з Франції в 1997 році і доставлених компанією Speedy Express?

Переведемо ці питання в запити на мові SQL.

Питання	SQL- запит
Яка сумарна вартість замовлень, зроблених клієнтами з Франції?	SELECT SUM (ExtendedPrice) FROM invoices1 WHERE Country='France'

Яка сумарна вартість замовлень, зроблених клієнтами з Франції і доставлених компанією Speedy Express?	SELECT SUM (ExtendedPrice) FROM invoices1 WHERE Country=’France’ AND ShipperName=’Speedy Express’
Яка сумарна вартість замовлень, зроблених клієнтами з Франції в 1996 році і доставлених компанією Speedy Express?	SELECT SUM (ExtendedPrice) FROM Ord_pmt WHERE CompanyName=’Speedy Express’ AND OrderDate BETWEEN ‘December 31, 1995’ AND ‘April 1, 1996’ AND ShipperName=’Speedy Express’

Результатом будь-якого з перерахованих вище запитів є число. Якщо в першому із запитів замінити параметр ‘France’ на ‘Austria’ або на назву іншої країни, можна знову виконати цей запит і отримати інше число. Виконавши цю процедуру зі всіма країнами, ми отримаємо наступний набір даних:

Country	**SUM (ExtendedPrice)**
Argentina	7327.3
Austria	110788.4
Belgium	28491.65
Brazil	97407.74
Canada	46190.1
Denmark	28392.32
Finland	15296.35
France	69185.48
Germany	209373.6
…	…

Отриманий набір агрегатних значень може бути інтерпретований як одномірний набір даних.

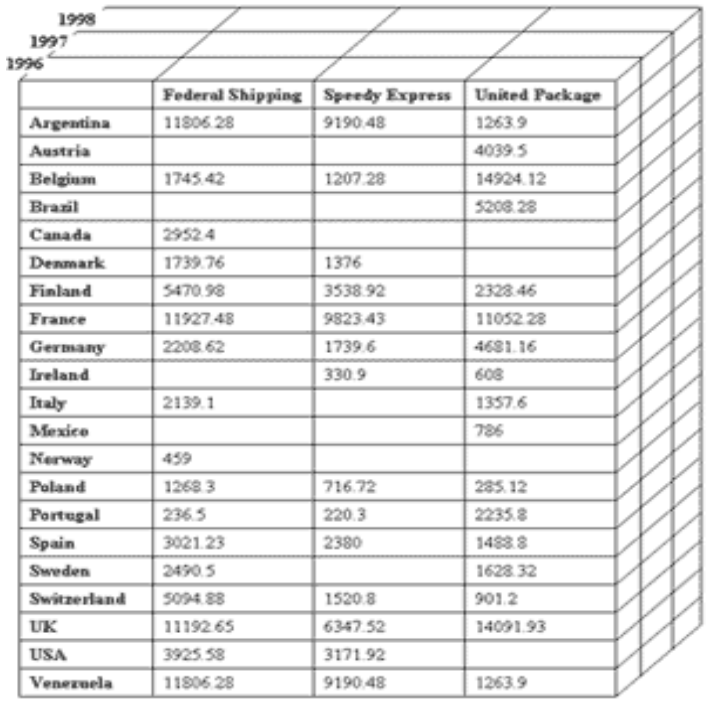
Тепер звернемося до другого з приведених вище запитів, який містить дві умови в пропозиції WHERE. Якщо виконувати цей запит, підставляючи в

нього всі можливі значення параметрів Country і ShipperName, ми отримаємо двомірний набір даних наступного вигляду:

Country	**ShipperName**		
	Federal Shipping	Speedy Express	United Package
Argentina	1 210.30	1 816.20	5 092.60
Austria	40 870.77	41 004.13	46 128.93
Belgium	11 393.30	4 717.56	17 713.99
Brazil	16 514.56	35 398.14	55 013.08
Canada	19 598.78	5 440.42	25 157.08
Denmark	18 295.30	6 573.97	7 791.74
Finland	4 889.84	5 966.21	7 954.00
France	28 737.23	21 140.18	31 480.90
Germany	53 474.88	94 847.12	81 962.58
…	…	…	…

Такий набір даних називається зведеною таблицею (Pivot Table) або кросс-таблицею (Cross Table). Створювати подібні таблиці дозволяють багато електронних таблиць і настільні СУБД - від Paradox для DOS до Microsoft Excel.

Третій з розглянутих вище запитів має вже три параметри в умові WHERE. Варіюючи їх, ми отримаємо трьохмірний набір даних (рис. 2.16). Ячейки куба містять агрегатні дані, відповідні значенням параметрів запиту, що знаходяться на осях куба в пропозиції WHERE. Можна отримати набір двомірних таблиць за допомогою перетину куба площинами, паралельними його граням (для їх позначення використовують терміни Cross Sections і Slices).



	Federal Shipping	Speedy Express	United Package
Argentina	11806.28	9190.48	1263.9
Austria			4039.5
Belgium	1745.42	1207.28	14924.12
Brazil			5208.28
Canada	2952.4		
Denmark	1739.76	1376	
Finland	5470.98	3538.92	2328.46
France	11927.48	9823.43	11052.28
Germany	2208.62	1739.6	4681.16
Ireland		330.9	608
Italy	2139.1		1357.6
Mexico			786
Norway	459		
Poland	1268.3	716.72	285.12
Portugal	236.5	220.3	2235.8
Spain	3021.23	2380	1488.8
Sweden	2490.5		1628.32
Switzerland	5094.88	1520.8	901.2
UK	11192.65	6347.52	14091.93
USA	3925.58	3171.92	
Venezuela	11806.28	9190.48	1263.9

Рис. 2.16. Трьохмірний набір агрегатних даних.

Якщо в пропозиції WHERE міститься чотири або більше параметрів, результуючий набір значень (також званий OLAP-кубом) може бути 4-мірним, 5-мірним і т.д.

Розглянувши, що є багатовимірними OLAP-кубами, перейдемо до деяких ключових термінів і понять, використовуваних при багатовимірному аналізі даних.

Разом з сумами в ячейках OLAP-куба можуть міститися результати виконання інших агрегатних функцій мови SQL, таких як MIN, MAX, AVG, COUNT, а в деяких випадках - і інших (дисперсії, середньоквадратичного відхилення і т.д.). Для опису значень даних в ячейках використовується термін Summary (у загальному випадку в одному кубі їх може бути декілька), для позначення початкових даних, на основі яких вони обчислюються, - термін Measure, а для позначення параметрів запитів - термін Dimension (що перекладається як «вимірювання», коли йдеться про OLAP-куби, і як «розмірність», коли йдеться про сховища даних). Значення, що відкладаються на осях, називаються членами вимірювань (Members).

Кажучи про вимірювання, слід згадати про те, що значення, що наносяться на осі, можуть мати різні рівні деталізації. Наприклад, нас може цікавити сумарна вартість замовлень, зроблених клієнтами в різних країнах, або сумарна вартість замовлень, зроблених іногородніми клієнтами або навіть окремими клієнтами. Природно, результуючий набір агрегатних даних в другому і третьому випадках буде детальнішим, ніж в першому. Відмітимо, що можливість отримання агрегатних даних з різним ступенем деталізації відповідає одній з вимог, що пред'являються до сховищ даних, - вимозі доступності різних зрізів даних для порівняння і аналізу.

Оскільки в розглянутому прикладі в загальному випадку в кожній країні може бути декілька міст, а в місті - декілька клієнтів, можна говорити про ієрархії значень у вимірюваннях. В цьому випадку на першому рівні ієрархії розташовуються країни, на другому - міста, а на третьому - клієнти (рис. 2.17).

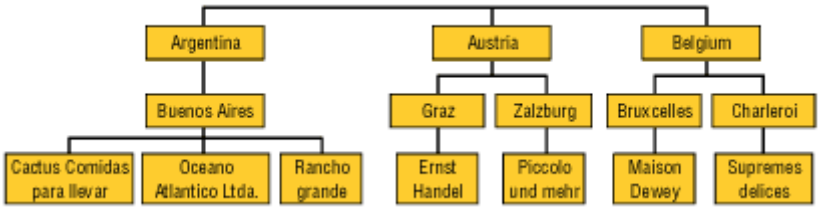


Рис. 2.17. Ієрархія у вимірюванні, пов'язаному з географічним положенням клієнтів.

Відзначимо, що ієрархії можуть бути збалансованими (balanced), як, наприклад, ієрархія, представлена на малюнку 2.17, а також ієрархії, засновані на даних типу «дата-час», і незбалансованими (unbalanced). Типовий приклад незбалансованої ієрархії - ієрархія типу «начальник-підлеглий (її можна побудувати, наприклад, використовуючи значення поля SalesPerson початкового набору даних з розглянутого вище прикладу)», представлений на рис. 2.18. Іноді для таких ієрархій використовується термін Parent-child hierarchy.

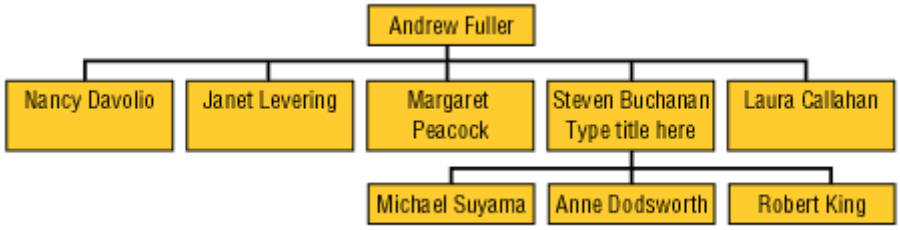


Рис. 2.18. Незбалансована ієрархія.

Існують також ієрархії, що займають проміжне положення між збалансованими і незбалансованими (вони позначаються терміном ragged – «нерівний»). Зазвичай вони містять такі члени, логічні «батьки» яких знаходяться не на безпосередньо вищестоящому рівні (наприклад, в географічній ієрархії є рівні Country, City і State, але при цьому в наборі даних є країни, що не мають штатів або регіонів між рівнями Country і City (рис. 2.19).

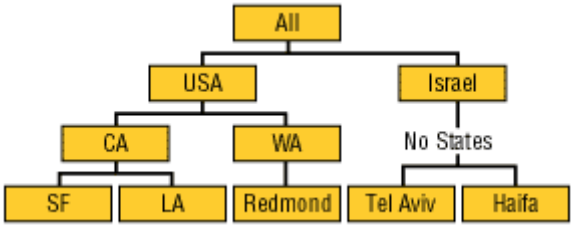


Рис. 2.19. «Нерівна» ієрархія.

Відзначимо, що незбалансовані і "нерівні" ієрархії підтримуються далеко не всіма OLAP-засобами. Наприклад, в Microsoft Analysis Services 2000 підтримуються обидва типи ієрархії, а в Microsoft OLAP Services 7.0 - тільки збалансовані. Різним в різних OLAP-засобах може бути і число рівнів ієрархії, і максимально допустиме число членів одного рівня, і максимально можливе число самих вимірювань.

Багатовимірний аналіз даних може бути проведений за допомогою різних засобів, які умовно можна розділити на клієнтські і серверні OLAP-засоби. Клієнтські OLAP-засоби є додатки, що здійснюють обчислення агрегатних даних (сум, середніх величин, максимальних або мінімальних

значень) і їх відображення, при цьому самі агрегатні дані містяться в кеші усередині адресного простору такого OLAP-засобу. Якщо початкові дані містяться в настільній СУБД, обчислення агрегатних даних проводиться самим OLAP-засобом. Якщо ж джерело початкових даних - серверна СУБД, багато хто з клієнтських OLAP-засобів посилає на сервер SQL-запити і в результаті отримують агрегатні дані, обчислені на сервері [3, 51, 62].

Відзначимо, що клієнтські OLAP-засоби застосовуються, як правило, при малому числі вимірювань (зазвичай рекомендується не більше шести) і невеликій різноманітності значень цих параметрів, - адже отримані агрегатні дані повинні уміщатися в адресному просторі подібного засобу, а їх кількість росте експоненціально при збільшенні числа вимірювань. Тому навіть найпримітивніші клієнтські OLAP-засоби, як правило, дозволяють зробити попередній підрахунок об'єму необхідної оперативної пам'яті для створення в ній багатовимірного куба. Багато клієнтських OLAP-засобів дозволяють зберегти вміст кеша з агрегатними даними у вигляді файлу, що, у свою чергу, дозволяє не проводити їх повторне обчислення. Відзначимо, що нерідко така можливість використовується для відчуження агрегатних даних з метою передачі їх іншим організаціям або для публікації. Типовим прикладом таких відчужуваних агрегатних даних є статистика захворюваності в різних регіонах і в різних вікових групах, яка є відкритою інформацією, що публікується міністерствами охорони здоров'я різних країн і Всесвітньою організацією охорони здоров'я.

Ідея збереження кеша з агрегатними даними у файлі отримала свій подальший розвиток в серверних OLAP-засобах, в яких збереження і зміна агрегатних даних, а також підтримка сховища, що містить їх, здійснюються окремим додатком або процесом, званим OLAP-сервером. Клієнтські додатки можуть запрошувати подібне багатовимірне сховище і у відповідь отримувати ті або інші дані. Деякі клієнтські додатки можуть також створювати такі сховища або оновлювати їх відповідно до початкових даних, що змінилися.

Переваги застосування серверних OLAP-засобів в порівнянні з клієнтськими OLAP-засобами схожі з перевагами застосування серверних СУБД в порівнянні з настольнимі: у разі застосування серверних засобів обчислення і зберігання агрегатних даних відбуваються на сервері, а клієнтський додаток отримує лише результати запитів до них, що дозволяє в загальному випадку понизити мережевий трафік, час виконання запитів і вимоги до ресурсів, споживаним клієнтським додатком. Відзначимо, що засоби аналізу і обробки даних масштабу підприємства, як правило, базуються саме на серверних OLAP-засобах, наприклад, таких як Oracle Express Server, Microsoft SQL Server 2000 Analysis Services, Hyperion Essbase, продуктах компаній Crystal Decisions, BusinessObjects, Cognos, SAS Institute. Оскільки всі провідні виробники серверних СУБД створюють ті або інші серверні OLAP-засоби, вибір їх достатньо широкий і майже у всіх випадках можна придбати OLAP-сервер того ж виробника, що і у самого сервера баз даних.

OLAP-система складається з множини компонент. На найвищому рівні уявлення система включає джерело даних, OLAP-сервер і клієнта. Джерело даних є засіб, з якого беруться дані для аналізу. Дані з джерела переносяться або копіюються на OLAP-сервер, де вони систематизуються і готуються для швидшого згодом формування відповідей на запити. Клієнт - це призначений для користувача інтерфейс до OLAP-сервера.

Джерела даних. Джерелом в OLAP-системах є сервер, що поставляє дані для аналізу. Залежно від області використання OLAP-продукту джерелом може служити сховище даних, успадкована база даних, що містить загальні дані, набір таблиць, об'єднуючих фінансові дані або будь-яка комбінація перерахованого. Здатність OLAP-продукту працювати з даними з різних джерел дуже важлива. Вимога єдиного формату або єдиної бази, в яких би зберігалися всі початкові дані, не підходить адміністраторам баз даних. Крім того, такий підхід зменшує гнучкість і потужність OLAP-продукту. Адміністратори і користувачі вважають, що OLAP-продукти, що забезпечують витягання даних

не тільки з різних, але і з безлічі джерел, виявляються гнучкішими і кориснішими, ніж ті, що мають жорсткіші вимоги.

Сервер. Прикладною частиною OLAP-системи є OLAP-сервер. Ця складова виконує всю роботу і зберігає в собі всю інформацію, до якої забезпечується активний доступ. Архітектурою сервера управляють різні концепції. Зокрема, основною функціональною характеристикою OLAP-продукту є використання для зберігання даних багатовимірної (ММБД, MDDB) або реляційної (РДБ, RDB) бази даних.

MOLAP. MOLAP - це Multidimensional On-Line Analytical Processing, тобто Багатовимірний OLAP. Це означає, що сервер для зберігання даних використовує ММБД. Оскільки більшість OLAP-продуктів засновані на ММБД, під OLAP часто розуміють також і MOLAP. Сенс використання ММБД очевидний. Вона може ефективно зберігати багатовимірні за своєю природою дані, забезпечуючи засоби швидкого обслуговування запитів до бази даних. Дані передаються від джерела даних в багатовимірну базу даних, а потім база даних піддається агрегації. Попередній розрахунок - це те, що прискорює OLAP-запити, оскільки розрахунок зведених даних вже проведений. Час запиту стає функцією виключно часу, необхідного для доступу до окремого фрагмента даних і виконання розрахунку. Цей метод підтримує концепцію, згідно якої робота проводиться одного разу, а результати потім використовуються знову і знову. Багатовимірні бази даних є відносно новою технологією. Використання ММБД має ті ж недоліки, що і більшість нових технологій. А саме - вони не так стійкі, як РБД, і в тій же мірі не оптимізовані. Інше слабке місце ММБД полягає в неможливості використовувати більшість багатовимірних баз в процесі агрегації даних, тому потрібний час для того, щоб нова інформація стала доступна для аналізу.

«Вибух» бази даних є феномен багатовимірних баз. Не дивлячись на те, що ця проблема досліджувалася фахівцями, проте, важко пояснити, чому і як це відбувається. Представляється, що це пов'язано з розрідженістю бази даних і попередньою агрегацією даних. Якщо багатовимірна база даних містить

невелике число елементів даних, порівнянне з кількістю забезпечуваних нею рівнів агрегації, кожен фрагмент даних вноситиме більший внесок до всіх отримуваних з нього даних. Коли база даних «вибухає», розмір її стає істотно більше, ніж він повинен бути. Складно визначити умови «вибуху» бази даних або передбачити, чи «вибухне» якась конкретна структура. Одним з підходів, який, схоже, може допомогти вирішити проблему «вибуху», є динамічне управління розрідженими даними. Ця методика дозволяє аналізувати свої власні моделі зберігання і оптимізувати їх з метою запобігання «вибуху» бази даних.

ROLAP. ROLAP - це Relational On-Line Analytical Processing, тобто Реляційний OLAP. Термін ROLAP означає, що OLAP-сервер базується на реляційній базі даних. Початкові дані вводяться в реляційну базу даних, зазвичай по схемі "зірка" або схемі "сніжинка", що сприяє скороченню часу витягання. Сервер забезпечує багатовимірну модель даних за допомогою оптимізованих SQL-запитів. Існує ряд причин для вибору саме реляційною, а не багатовимірної бази даних. РБД - це добре відпрацьована технологія, що має безліч можливостей для оптимізації. До того ж, РБД підтримують крупніші об'єми даних, чим ММБД. Вони якраз і спроектовані для таких об'ємів. Основним аргументом проти РБД є складність запитів, необхідних для отримання інформації з великої бази даних за допомогою SQL. Недосвідчений SQL-програміст міг би з легкістю обтяжити цінні системні ресурси спробами виконати який-небудь подібний запит, який в ММБД виконується набагато простіше.

Прикладний OLAP (HOLAP). Безумовно, це найбільша область, і це, загалом, те, з чим зазвичай зв'язують і що зазвичай розуміють під терміном «OLAP». Прикладний OLAP, як правило, складається з багатовимірних баз даних, доступ до яких відбувається через конкретний додаток, або, можливо, через безліч додатків. Постачальники в даній області ринку в основному пропонують клієнти для бази даних. Клієнт може бути як простим засобом перегляду, так і могутнішим додатком.

Настільний OLAP (DOLAP). Представниками настільного OLAP є продукти, що необов'язково з'єднуються з сервером. Вони можуть запускатися в основному на клієнтській частині, хоча дані у формі куба даних можуть завантажувати і з сервера. Той факт, що куб даних будується і зберігається на машині користувача, дозволяє рекомендувати їх тим, хто часто використовує портативні комп'ютери або хто нечасто запускає настільки складні звіти, що для їх формування необхідна вища швидкість клієнта, а, отже, і могутніший сервер для її забезпечення.

Швидка реалізація запитів є імперативом для OLAP. Це один з базових принципів OLAP - здатність інтуїтивно маніпулювати даними вимагає швидкого витягання інформації. В цілому, чим більше обчислень необхідно провести, щоб отримати фрагмент інформації, тим повільніше відбувається відгук. Тому, щоб зберегти маленький час реалізації запитів, фрагменти інформації, звернення до яких зазвичай відбувається найчастіше, але які при цьому вимагають обчислення, піддаються попередній агрегації. Тобто вони підраховуються і потім зберігаються в базі даних як нові дані. Як приклад типу даних, який допустимо розрахувати наперед, можна привести зведені дані - наприклад, показники продажів по місяцях, кварталах або роках, для яких дійсно введеними даними є щоденні показники.

Різні постачальники дотримуються різних методів відбору параметрів, що вимагають попередньої агрегації і числа заздалегідь обчислюваних величин. Підхід до агрегації впливає одночасно і на базу даних і на час реалізації запитів. Якщо обчислюється більше величин, вірогідність того, що користувач запитає вже обчислену величину, зростає, і тому час відгуку скорочується, оскільки не доведеться запрошувати початкову величину для обчислення. Проте, якщо обчислити всі можливі величини - це не краще рішення - у такому разі істотно зростає розмір бази даних, що зробить її некерованою, та і час агрегації буде дуже великим. До того ж, коли в базу даних додаються числові значення, або якщо вони змінюються, інформація ця повинна відбиватися на заздалегідь обчислених величинах, залежних від нових даних.

Клієнт. Клієнт - це якраз те, що використовується для уявлення і маніпуляцій з даними в базі даних. Клієнт може бути і достатньо нескладним - у вигляді таблиці, що включає такі можливості OLAP, як, наприклад, обертання даних (півотінг) і поглиблення в дані (дріллінг). Воно повинно бути спеціалізованим, але мати такий же простий засіб проглядання звітів або бути таким же могутнім інструментом, як створений на замовлення додаток, спроектований для складних маніпуляцій з даними. Клієнт є настільки важливий, що безліч постачальників зосереджують свої зусилля виключно на розробці клієнта. Все, що включається до складу цих додатків, представляє собою стандартний погляд на інтерфейс, наперед задані функції і структуру, а також швидкі рішення для більшості стандартних ситуацій, наприклад, популярні фінансові пакети. Наперед створені фінансові додатки дозволять фахівцям використовувати звичні фінансові інструменти без необхідності проектувати структуру бази даних або загальноприйняті форми і звіти.

Інструмент запитів/генератор звітів. Інструмент запитів або генератор звітів пропонує простій доступ до OLAP-даних. Вони мають простий у використанні графічний інтерфейс і дозволяють користувачам створювати звіти переміщенням об'єктів методом "drag and drop". Тоді як традиційний генератор звітів надає користувачеві можливість швидко випускати форматовані звіти, генератори звітів, підтримуючі OLAP, формують актуальні звіти. Кінцевий продукт є звіт, що має можливості поглиблення в дані до рівня подробиць, обертання (півотінг) звітів, підтримки ієрархій і ін.

Сьогодні в багатьох напрямах бізнесу за допомогою електронних таблиць проводяться різні форми аналізу корпоративних даних. У якомусь сенсі це ідеальний засіб створення звітів і проглядання даних. Аналітик може створювати макроси, що працюють з даними у вибраному напрямі, а шаблон може бути спроектований таким чином, що, коли відбувається введення даних, формули розраховують правильні величини, виключаючи необхідність неодноразового введення простих розрахунків. Проте, все це дає в результаті «плоский» звіт, що означає слідуюче: як тільки він створений, важко

розглядати його в різних аспектах. Наприклад, діаграма відображає інформацію за деякий часовий період, - скажімо, за місяць. І якщо хтось бажає побачити показники за день (в протилежність даним за місяць), необхідно буде створити абсолютно нову діаграму. Належить визначити нові набори даних, додати в діаграму нові мітки і внести безліч інших простих, але трудомістких змін. Крім того, існує ряд областей, в яких можуть бути допущені помилки, що в цілому зменшує надійність. Коли до таблиці додається OLAP, з'являється можливість створювати єдину діаграму, а потім піддавати її різним маніпуляціям з метою надання користувачеві необхідної інформації, не обтяжуючи себе створенням всіх можливих уявлень.

Інтернет в ролі клієнта. Новим членом сімейства OLAP-клієнтів є Інтернет. Існує маса переваг у формуванні OLAP-звітів через Інтернет. Найбільш істотною представляється відсутність необхідності в спеціалізованому програмному забезпеченні для доступу до інформації.

Кожен Інтернет-продукт специфічний. Деякі спрощують створення Web-сторінок, але мають меншу гнучкість. Інші дозволяють створювати представлення даних, а потім зберігати їх як статичні HTML-файли. Все це дає можливість проглядати дані через Інтернет, але не більш того. Активно маніпулювати даними з їх допомогою неможливо. Існує і інший тип продуктів - інтерактивний і динамічний, такий, що перетворює такі продукти на повнофункціональні інструменти. Користувачі можуть здійснювати поглиблення в дані, півотінг, обмеження вимірювань і ін. Перш, ніж вибрати засіб реалізації Інтернет, важливо зрозуміти, які функціональні можливості потрібні від Web-рішення, а потім визначити, який продукт найкращим чином утілить цю функціональність.

Додатки. Додатки - це тип клієнта, що використовує бази даних OLAP. Вони ідентичні інструментам запитів і генераторам звітів, описаним вище, але, крім того, вони вносять до продукту ширші функціональні можливості. Додаток, як правило, володіє більшою потужністю, чим інструмент запиту.

Середовище. Зазвичай постачальники OLAP забезпечують середовище розробки для створення користувачами власних настроєних додатків. Середовище розробки в цілому є графічним інтерфейсом, що підтримує об'єктно-орієнтовану розробку додатків. До того ж, більшість постачальників забезпечують API, який може використовуватися для інтеграції баз даних OLAP з іншими додатками.

Розглянемо деякі напрями діяльності основних виробників програмних засобів підтримки OLAP - технологій з урахуванням вищеприведенною класифікації архітектур.

Різні постачальники реалізують OLAP на основі власних корпоративних уявлень про те, що повинно входити в ідеальний OLAP-продукт. Постачальники, розглянуті нижче, дотримувалися різних підходів, включаючи і ті, що засновані на багатовимірній базі даних, реляційній базі даних і додатках, що реалізують можливості OLAP на різних рівнях. Процес їх реального функціонування може служити кількісним параметром при розгляді різних підходів до OLAP. З цією метою OLAP Council розробив атестаційне завдання APB-1 для кількісного порівняння роботи різних OLAP-продуктів. OLAP Council є консорціумом, утвореним декількома постачальниками OLAP для підтримки ключових принципів OLAP. Вони визнають, що OLAP є найважливішою технологією для забезпечення корпоративних аналітиків інструментами, потрібними для виконання необхідного їм аналізу. У квітні 1996 р. було випущено перше атестаційне завдання в області OLAP - APB-1. Контрольне завдання визначає параметри бази даних і встановлює набір з 10 запитів, що відображають нормальне використання. Постачальник OLAP створює відповідну базу даних і потім запускає запити. Окреме вимірювання - AQT (середній час запиту, Average Query Time), генерується на основі часу, який витрачається на завантаження бази даних, агрегацію даних і подальший запуск запитів. Правила визначають, що легально і що нелегально - наприклад, чи потрібно розраховувати наперед значення даних, які з розрахованих величин можуть зберігатися і ін.

В області MOLAP архітектури лідером є компанія Oracle [43]. Лінійка продуктів Express та Oracle OLAP Services є корпоративним підходом Oracle до OLAP, включаючи сервер, клієнтську частину, можливості ROLAP і Інтернет-рішення. Express Server був головною OLAP-машиною для Oracle. Він надавав багатовимірну базу даних і був машиною для інших OLAP-продуктів фірми. До того ж, Oracle пропонує Personal Express - локально працюючий сервер Express. Він надає користувачам доступ і можливість працювати з базою даних в автономному режимі. Все це ідеально підходить для мобільних комп'ютерів. Головною особливістю Express Server є здатність використовувати різноманітні джерела даних. Дані для багатовимірної бази даних можуть збиратися з реляційної бази (за допомогою Express Relational Access Manager), багатовимірної бази, табличного або плоского файлу. Користувачі можуть розробляти власні OLAP-додатки для Express за допомогою Express Objects, який забезпечує розробника графічним інтерфейсом і об'єктно-орієнтованим підходом для створення додатків. Oracle також забезпечує безліч шляхів доступу до бази даних. Express Analyzer містить графічний інтерфейс до бази даних і дозволяє користувачеві легко формувати звіти. Крім того, Analyzer може мати загальні об'єкти з Express Objects, а також випускати додатки, розроблені з їх допомогою. Discoverer найточніше можна описати як інструмент запитів до даних. Він простіший, ніж Analyzer, проте найбільш популярний серед засобів запитів до даним. Крім цього, Express містить додаткові таблиці, які можна використовувати спільно з Excel.

У Oracle існує два готові додатки. Це Financial Analyzer і Sales Analyzer. Вони використовують машину і кеш даних Express Server, і містять конфігурації звітів, розроблені для потреб фінансових аналітиків і аналітиків продажів. Вони корисні і користувачам, оскільки визначають важливі функції, що зазвичай беруть участь в обох видах аналізу, який реалізовано в Express Server.

Web Agent і Web Publisher - це засоби створення Інтернет-ресурсів, що надаються Express. Використовуючи будь-який з цих засобів, користувач може

створити динамічний і інтерактивний сайт, що забезпечується застосуванням різних можливостей OLAP. Web Agent в більшій мірі інструмент розробника, він є набором заздалегідь певних процедур, включених в Express Server SPL. Сайти можуть будуватися так само, як будуються призначені для користувача інтерфейси до бази даних. Для кінцевого користувача існує Web Publisher. Web Publisher зв'язаний з Express Analyzer і має можливість для створення власних інтерактивних сайтів тими, хто не має серйозного досвіду програміста. Web Publisher в основному є "майстром", який веде користувача через всі етапи побудови сайту і забезпечує графічний інтерфейс для підтримки його створення.

Arbor Software Corporation - це головний суперник Oracle. Її продуктом є Essbase, а найостаннішим його релізом був Essbase Server 5. Дуже популярний сервер Arbor для різних OLAP-продуктів, що говорить про те, що багато постачальників OLAP не обов'язково випускають повні додатки, а можуть використовувати як базу даних Arbor, і потім створювати інтерфейси до бази даних. Як приклад можна привести Comshare і Web-компоненту для Arbor - Crystalinfo, Seagate Software, що випускається. Останнім часом Arbor уклав партнерську угоду з IBM. Багатовимірне зберігання даних в Arbor Essbase Server буде замінено на DB2 від IBM. Передбачається, що це буде ROLAP-система, але фактично це не так. Це просто OLAP-система без однієї з кращих властивостей багатовимірних баз даних, але з перевагами системи РБД.

В доповнення до таблиць Essbase Spreadsheet Add-in, що забезпечують користувачів можливостями OLAP, Arbor пропонує WIRED for OLAP (засіб аналізу і презентацій), Crystal Info for Essbase (генератор звітів і розкладів) і SQL Drill-Through, що дозволяє користувачам проглядати подробиці даних в початкових реляційних базах. Arbor також випустив Arbor Essbase Adjustment Module. Цей додаток допомагає користувачам в підготовці звітів, що регулярно випускаються. Він сприяє автоматизації форматування звітів і процесів розрахунку. Крім того, існує ще Arbor Currency Conversion Module, здатний

конвертувати різні валюти в національну на основі моделі для відстежування обмінних курсів.

В області ROLAP визнаним лідером є компанія MicroStrategy. Їх філософією є відсутність обмежень на розмір сховища даних, так що немає жодних проблем з його збільшенням. Оскільки вони є виробниками реляційного OLAP, рівень їх аналітичних систем достатньо високий. У MicroStrategy немає OLAP-машини, яка могла б працювати локально, що незручно для користувачів, які часто працюють з ноутбуками або для тих, хто просто вважає за краще працювати автономно. Проте, у них існує продукт, DSS Broadcaster, що дозволяє посилати дані на різні вихідні пристрої. DSS Broadcaster посилає дані за запитом або коли відбувається певна подія. Наприклад, менеджерові може відсилатися щоденне оновлення з сумами прибутку за попередній день. Ця інформація може поступати по електронній пошті, на пейджер або мобільний телефон, а також факсом.

DSS Server є центральним продуктом в лінійці продуктів MicroStrategy. Це могутня машина, що дозволяє іншим агентам діставати доступ до реляційної бази даних в багатовимірному режимі. DSS Server містить різноманітні драйвери баз даних для оптимізації їх під необхідну реляційну базу даних (вони підтримують Oracle, DB2, Sybase, Red Brick, Informix, і інші реляційні бази). До того ж, акцент робиться на здатність їх зростання і включає драйвер для адаптації до дуже великих баз даних (Very Large Databases, VLDBs), розмір яких перевищує терабайти. Природа реляційного OLAP-продукту обмежує MicroStrategy в можливості надання дійсно індивідуального сервера для автономної роботи, проте за допомогою DSS Agent набір даних може завантажуватися і аналіз може виконуватися і в автономному режимі. DSS Agent є клієнт або клієнтський інструмент до DSS Server. Однією з переваг DSS Agent є використання інтелектуальних агентів для автоматизації бізнес-процессов. Наприклад, за допомогою DSS Agent можна створити агента, що знає, коли і де необхідно шукати дані і потім що з ними робити потім (тобто, як проводити їх очищення і куди їх помістити). Використовуючи агентів, можна

автоматизувати безліч звичайних, але часто повторюваних завдань. DSS Executive використовує можливості DSS Agent для реалізації високоякісного генератора звітів і засобу аналізу. Він використовує об'єктно-орієнтований підхід і інтерфейс, що працює за технологією "drag-and-drop" для швидкого створення додатків управлінських інформаційних систем силами самих користувачів. І, нарешті, MicroStrategy пропонує доповнення Excel Add-in, яке може використовуватися для додання таблицям функціональних можливостей OLAP.

Нове покоління Інтернет-орієнтованого OLAP від MicroStrategy представлене DSS Web 5.0. Однією з примітних властивостей DSS Web 5.0 є підтримка Microsoft webcasting standart. Це дозволяє автоматично передавати web-сторінки на комп'ютер користувача. У числі найважливіших можливостей DSS Web можна назвати здатність зберігати карти або діаграми, отримані з Інтернет, майстри звітів і пакети звітів, що настроюються.

Як приклад прикладного OLAP-продукту можна узяти Comshare. Не дивлячись на те, що це додаток, воно доповнює продукт функціональними можливостями OLAP. Comshare Decision проявляє гнучкість щодо використовуваного спільно з ним сервера. Arbor Essbase і Oracle Express - всі ці багатовимірні сервери баз даних можуть використовуватися спільно з Decision.

Hyperion Software - це також виробник прикладного OLAP-продукту, що випускає виключно OLAP-клієнти. Останнім продуктом був Hyperion MBA, або Multidimensional Business Analyst, що замінив HyperionOLAP. Згідно з останніми даними, Hyperion займав другий за величиною сегмент ринку. Природа продукту гарантує, що велика частина функціональних можливостей заснована на серверній базі даних. Hyperion популярний завдяки своїм додатковим аналітичним можливостям, що реалізовуються у формі складних вимірювань, заздалегідь певних функцій і звітності. Метою його є формування могутнього фінансового пакету, що включає OLAP.

Hyperion пропонує два клієнтські OLAP-рішення. Перше, HyperionMBA, використовує OLAP для бізнес-аналіза. Як це зазвичай буває в

OLAP-додатках, Hyperion застосовує в своїх рішеннях складні вимірювання і заздалегідь певні функції для розрахунків і маніпуляцій з валютами. Програма Hyperion Analytic Accounting включає властивості OLAP в розрахункові пакет.

Cognos служить непоганим прикладом настільного OLAP-продукту. Це означає, що велика частина обробки проводиться не на сервері, а локально. Impromptu є інструментом запитів, використовуваним для витягання даних з багатовимірної бази даних. Дані потім поміщаються в Powerplay, яка зберігає куб даних на робочому столі комп'ютера користувача.

2.6. OLAP – системи та Інтернет - технології

Щоб ствердження про необхідність і повсюдне розповсюдження OLAP не здавалися голослівними, розглянемо декілька прикладів рішення задач за допомогою OLAP-інструментів великими зарубіжними підприємствами [66, 71].

Наш перший приклад - одна з найбільших американських енергетичних компаній, Duke Energy, чиї активи перевищують 20 мільярдів, а число співробітників складає 22 000 чоловік, що працюють в різних підрозділах компанії по всьому світу. Сферою діяльності компанії є енергетичне забезпечення, обслуговування трубопроводів і постачання електричної енергії, природного газу і зрідженого природного газу. На момент впровадження OLAP компанія припускала вийти на нерегульований урядом енергетичний ринок і відвоювати собі певний шматок світового енергетичного бізнесу. В процесі підготовки до цього нового етапу відділ управління інформацією Duke Energy опрацьовував різні можливості застосування технологій для управління змінними інформаційними потребами підприємства і забезпечення йому гідного місця на ринку і відповідного розвитку. Питання полягало в забезпеченні своїх кінцевих користувачів доступом до інформації, що зберігається в корпоративних базах даних. Фахівці з управління інформацією

з'ясували, що в процесі ухвалення рішень всім підрозділам, починаючи з фінансових відділів і закінчуючи кадровими, важливо мати доступ до такої інформації тоді і там, де вона їм необхідна.

Система, яку Duke Energy використовувала до цього моменту, була складною в плані використання і без серйозного втручання фахівців з управління інформацією не забезпечувала необхідний підприємству рівень точності і своєчасності даних. Плануючи апгрейд систем підтримки прийняття рішень так, щоб співробітники фінансових і інші нетехнічні підрозділи компанії могли якнайповніше використовувати корпоративні дані, Duke Energy ухвалила рішення про впровадження OLAP-системи. Була вибрана система підтримки прийняття рішень від Business Objects. В результаті її впровадження, дані, що знаходяться в оперативній базі PeopleSoft і вітрині даних IBM DB2 HRMS стали доступні для OLAP-аналізу і різноманітних запитів і звітів різних підрозділів компанії. Рішення використовує RDTs (Шаблони швидкого розгортання, Rapid Deployment Templates), що містять зразки звітів, які користувачі без спеціальних технічних знань можуть настроювати на свій розсуд для генерації незапланованих запитів до даним і звітів.

В даному випадку компанія визначила, що пропозиція Business Objects найкращим чином відповідає її потребам. Але це не означає, що інші рішення в чомусь гірше вибраного. Кожне підприємство, що ухвалило рішення про впровадження OLAP, в першу чергу розглядає оптимальність пропозицій різних постачальників стосовно своїх власних проблем і потреб. Наприклад, та ж фірма Cognos успішно поставляє свої продукти таким серйозним клієнтам, як Міністерство оборони США і компанії Boeing. Будучи клієнтом Cognos з 1996 року, Міністерство оборони постійно оновлює наявні системи, доповнюючи їх новими можливостями і розробками. У листопаді минулого року фінансовим підрозділом Міністерства (DFAS) була придбана система бізнес-репортінга і аналізу з новими можливостями візуалізації, вартістю 1 мільйон доларів. Нове програмне рішення об'єднує дані з безлічі розрізнених систем, роблячи їх доступними для тисяч співробітників DFAS і інших агентств Міністерства

через захищений Інтернет-портал. «Візуалізатор» перетворить складні дані в зрозумілу зручну для використання інформацію, відображаючи її з використанням багатої графіки, різноманітних презентаційних можливостей і функцій оцінки даних. Роберт Еш, перший віце-президент Cognos, називає дане рішення «ядром електронного Уряду (e-government)», що забезпечує федеральним агентствам можливість використовувати Інтернет для спрощення і прискорення доступу до необхідної їм інформації. Впровадження OLAP-технології дозволило створити таку картину для кінцевих користувачів, що дістали у результаті можливість сформувати поглиблене уявлення про роботу Агентства, що у свою чергу, сприятиме поліпшенню управління і ухвалення рішень відповідними співробітниками.

Перед Boeing Company стояла декілька інше завдання. Оскільки всі літаки зібрані на замовлення і включають до декількох мільйонів різних компонентів, співробітникам було потрібне серйозне аналітичне рішення для обробки даних таких істотних об'ємів. В рамках спеціальної програми по створенню інфраструктури для репортінга у сфері прийняття рішень в підрозділі комерційної авіації почато впровадження продукту Cognos Impromptu Web Reports, за допомогою якого 11500 співробітників підрозділу зможуть генерувати через Інтернет звіти до бази даних по накладним і специфікаціям на компоненти літальних апаратів. Керівники Boeing Company вважають, що нове рішення заощадить значну кількість людино-годин, що раніше витрачалися на створення таких звітів вручну. Всі разом це закономірно сприятиме поліпшенню процесу прийняття рішень.

Компанія MicroStrategy, як і більшість лідерів цього ринку, також пропонує спеціалізовані рішення для окремих областей бізнесу - фінансів, страхування, охорони здоров'я, урядових організацій і ін. - на базі MicroStrategy Business Intelligence Platform. Одне з таких рішень знайшло застосування в області енергетики. Компанія KeySpan, найбільший на північному сході США постачальник природного газу, що має підрозділи в Брукліне, Бостоні і на Лонг-Айленде, керівник ряде сервісних енергетичних компаній і більш ніж 13

000 співробітників, використовує MicrosSrategy Narrowcast Server для забезпечення Нью-Йоркських домовласників і підприємств достатньою кількістю енергії для опалювання і підігріву води. Narrowcast Server, що є ключовим компонентом MicroStrategy Business Intelligence Platform, надає користувачам могутні аналітичні можливості і інтелектуальну систему попередження. Зокрема, система посилає попередження про зниження або підвищення витрати газу щодо проектного на відповідні адреси електронної пошти і пейджери. Клієнти KeySpan, «продавці», що поставляють газ кінцевим користувачам, можуть підключитися до інформаційних ресурсів компанії для перегляду і аналізу проектного і реального використання газу і ухвалення обґрунтованих рішень щодо об'ємів газу, необхідних для передачі по трубопроводах в кожен конкретний день. KeySpan використовує технології MicroStrategy для аналізу таких чинників, як історичні дані і погодні умови, щоб спланувати об'єми постачань газу. Narrowcast Server чотири рази на день порівнює фактичний потік газу із запланованим раніше того ж дня і посилає звіт про непланові ситуації по електронній пошті або пейджеру відповідному клієнтові KeySpan і оперативним підрозділам, контролюючим вентилі трубопроводу. Таким чином компанії-постачальники можуть оптимізувати об'єми газу, що поставляється споживачам залежно від ряду різних чинників.

Одним із сучасних напрямків розвитку систем інтелектуального аналізу даних є об'єднання OLAP з технологією Data Mining і сховищами даних. Ці всі три технології розвиваються у міру того, як компанії починають усвідомлювати цінність даних. Реляційні бази даних свого часу були революційним рішенням, яке дозволило підприємствам збирати дані з щоденних транзакцій у великомасштабні засоби зберігання. За допомогою SQL було можливо виконувати елементарний аналіз цих даних. Коли ж був потрібен складніший аналіз, з'ясувалося, що SQL і РБД зовсім не ідеальне рішення. Таблиці були в змозі забезпечувати гнучкіший аналіз, але мали ряд істотних недоліків. Дані, що підлягають імпорту в таблицю з бази даних і сама таблиця були не в змозі ефективно оперувати великими об'ємами даних. З часом все більше і більше

компаній почали реалізовувати сховища даних і застосовувати до своїх даних засоби Data Mining. Сховище даних забезпечує зберігання очищених корпоративних даних. Дані по транзакціях перевіряються на коректність, категорізуються і потім поміщаються в сховищі. Інструменти Data Mining дозволяють аналітикам підприємств виявити приховані тенденції даних. Інструмент OLAP дає можливість виконувати швидкий і простій аналіз даних. В цілому, користувач-аналітик має уявлення про те, що він збирається знайти в деякому представленні даних. Він просто хоче мати засіб маніпулювання даними щоб найнаочніше відобразити деякі їх аспекти.

Іншим напрямком розвитку таких систем широке застосування OLAP в корпоративних порталах. Корпоративний портал - це «точка доступу», яка забезпечує зовнішніх і внутрішніх користувачів єдиним, безпечним мережевим інтерфейсом з персоніфікованим контентом. Контент слід розуміти в найширшому сенсі, як все, що об'єднує портал в одному призначеному для користувача інтерфейсі: додатки, інформація і інструменти спільної діяльності. До додатків відносяться внутрішні або зовнішні операційні або аналітичні програмно-апаратні засоби. Інформація є структурованими даними, результатами аналізу (звіти, куби, графіки, таблиці і т. п.), внутрішнім або зовнішнім неструктурованим контентом (документи, вміст цифрових накопичувачів, статистику відвідин сайту). Інструменти спільної діяльності - це веб-чати, мережеві конференції, електронна пошта, служби миттєвих повідомлень і т.п. (рис. 2.20).

Рис. 2.20. Корпоративний OLAP портал.

Кажучи про місце OLAP-технологій в порталі, буде цікавим навести точку зору фахівців з консалтингової компанії Object Systems Group. На їх думку, сфера застосування OLAP-інструментів в широкому сенсі - це «просунутий» бізнес-аналіз (Business Intelligence). Ці інструменти можуть звертатися до даних різних об'єктів, співвідносити їх, а також здійснюють пошук інформації (з тематики) в масштабах, непідвладних більшості користувачів, у випадку якби довелося робити те ж саме вручну. На сьогоднішній день, більшість компаній вже мають в своєму розпорядженні багато з компонентів, необхідних для впровадження OLAP-порталів. Ніхто не стане сперечатися, що можливість витягнути будь-яку бізнес-інформацию, задавшись питанням «як це співвідноситься з тим, чим я займаюся?» - могутній

інструмент підтримки прийняття рішень, про який користувачі можуть тільки мріяти.

Важливим напрямком розвитку систем інтелектуального налізу даних, які набуває широкої популярності є поєднання OLAP – засобів та Web – технологій. Динамічні технології, поява яких стала можлива в результаті розвитку World Wide Web, є прекрасною альтернативою традиційним клієнт-серверним OLAP-методам. За останній час з'явилися цілий ряд OLAP-засобів (їх називають WEB-OLAP або WOLAP), оснащених Web-можливостями. Вони виконують аналітичні функції, такі як агрегація і деталізація (drill-up і drill-down), а також забезпечують високу продуктивність у поєднанні зі всіма перевагами, які дає Web-додаток. Поява Web OLAP-засобів стирає межі, що відокремлюють OLAP-ринок від суміжних категорій програмного забезпечення. Web-платформи інтерактивної звітності по своїй функціональності все більше і більше схожі на стандартні Web OLAP продукти. Більшість WEB-OLAP додатків використовують загальну архітектуру, в якій клієнтський браузер взаємодіє з HTTP-сервером, що пересилає HTML-сторінки. Але крім цього надається ще і проміжне ПО, таке, що зберігається на сервері. Такий компонент може безпосередньо зв'язуватися з Web-браузером або взаємодіяти з HTTP-сервером, який потім повертає браузеру HTML-сторінки з додатковими даним.

WEB-OLAP компонент проміжного рівня виконує набір функцій, які не може забезпечити HTML, а саме:

- взаємодія з базою даних, де знаходиться сховище;
- зберігання станів (попередніх транзакцій бази даних);
- обчислення і буферизація даних, що повертаються на клієнт.

На сьогоднішній день реалізовано декілька різних рішень WEB-OLAP, у тому числі на основі технологій HTML (DHTML), Java, ActiveX, а також їх комбінацій. Розглянемо основні типи таких програмних продуктів:

- HTML (DHTML) - рішення.
- HTML з розширеннями – CGI.

- HTML з використанням Java-апплетів.
- Java або ActiveX - компоненти.

HTML – рішення. Для реалізації OLAP-функціональності в Web-браузері використовується лише HTML. Простим прикладом такого рішення є OLAP-інструмент, що дозволяє користувачеві виконувати обумовлені OLAP-запити або звіти з браузеру. В цьому випадку для здобуття даних з автономних OLAP-машин використовуються планувальники, що формують статичні HTML-звіти по HTML-шаблонам. Шаблони створюються так, щоб всі звіти мали погоджений вигляд. Звіти обробляються і передаються в браузер за допомогою Web-сервера. Статичні звіти характеризуються гарною переносимістю, швидко доставляються в браузер, при цьому взаємодії користувача з браузером практично не відбувається. В деяких випадках імітується перехід по вимірах шляхом навігації по звіту. Планувальник може створити набір звітів, зв'язаних між собою гіперпосиланнями. Наприклад, користувач, клацнувши по посиланню під назвою «Третій квартал», перейде до іншого звіту, що містить дані за липень, серпень і вересень.

Існує і інший підхід, коли OLAP-сервер наповнює HTML-шаблон даними в оперативному режимі, тобто по мірі появи запиту користувача через браузер. В цьому випадку на Web-сервері зберігаються лише шаблони звітів і метадані. Ці метадані містять інформацію, необхідну Web-серверу для передачі тих або інших даних в HTML-файл перед тим, як відправити його до браузера.

При будь-якому способі зберігання метаданих шаблонів і звітів інформація збирається Web-сервером згідно коду звіту, що посилається з браузера. Програмне забезпечення Web-сервера використовує метадані звіту, аби витягує відповідні дані з бази. База може зберігатися як на тому ж комп'ютері, що і додаток Web-сервера, так і на іншому. Отримані з бази дані об'єднуються на основі шаблону в звіт і передаються в браузер. Як правило, звіт вже містить за умовчанням певну OLAP-функціональність. При взаємодії із звітом призначений для користувача код посилається разом з іншою інформацією на Web-сервер, що використовує цей код для відстежування

інформації, яку користувач бачить в браузері. Повний набір OLAP-функцій (обертання, агрегація і поглиблення в дані) при такому підході не передбачений.

HTML з розширеннями – CGI. Одне з найслабших місць HTML – неможливість зберігати стану. Пропонований варіант вирішення проблеми – використання CGI (Common Gateway Interface – загальний шлюзовий інтерфейс) або інших Web API (Application Programming Interface - інтерфейс прикладного програмування) для реалізації єднального ПО (middleware). За допомогою цього методу можна забезпечити зберігання станів, а також буферизацію рядків і виконання деяких обчислень над переданими даними. В разі використання такої архітектури переносимість для клієнтських платформ забезпечується за рахунок використання HTML як інтерфейса. Проте і в цьому випадку використання лише HTML накладає обмеження на інтерфейси. Правильне відображення графіків і звітів буде утруднено. Ця проблема вирішується за допомогою Java-апплетів, які володіють великими можливостями по управлінню виведенням інформації на монітор.

HTML з використанням Java-апплетів. Спільне використання HTML і Java-апплетів дозволяє створити відмінне рішення WEB-OLAP. В цьому випадку HTML застосовується для відображення меню і виконання простих інтерфейсних функцій, а за допомогою Java-апплетів забезпечуються складніші компоненти інтерфейсу програми, а також погоджене відображення діаграм, графіків і таблиць. У таких застосуваннях найчастіше реалізовані функції агрегації і деталізації, а також обертання даних. За рахунок широких графічних можливостей Java в порівнянні з HTML, можна представляти дані у вигляді діаграм і змінювати їх в інтерактивному режимі.

Java або ActiveX - компоненти. Наступний підхід передбачає використання Java або ActiveX - компонентів для мінімізації взаємодії між браузером і Web-сервером, а також розширення можливостей користувача по роботі з даними за рахунок якіснішого інтерфейсу. Існує два способи вживання цих компонентів.

У першому випадку Web-сервер заповнює файл даними для звітів, і інтерфейсні компоненти посилаються в браузер разом з відповідним HTML-файлом. Браузер викачує цей файл, а компонент завантажує дані. Таким чином, компоненти забезпечують виконання таких OLAP-функцій, як агрегація, деталізація і обертання за допомогою зручного інтерфейсу, без звернення до сервера.

У другому випадку, інтерфейсний компонент безпосередньо з'єднується з сервером, який в інтерактивному режимі передає дані на клієнт по мірі появи запитів користувачів. В цьому випадку інтерфейсний компонент запрошує дані з Web-сервера, відкриваючи HTTP-потік. Потім, отримавши результати з сервера, він аналізує отриману інформацію і передає дані на об'єкт для відображення.

Історично склалося так, що інтелектуальний аналіз даних (Data Mining) і онлайнова аналітична обробка інформації (OLAP) були виключно прерогативою людини, тобто саме люди визначали і створювали аналітичні моделі, а потім використовували отримані з їх допомогою результати. Але з появою обчислювальної моделі Web-служб, які розглядаються як універсальний засіб об'єднання різнорідних систем, картина істотно змінилася, аналітику тепер можна легко пов'язати з іншими обчислювальними завданнями. Іншими словами, люди перестали бути єдиними творцями або споживачами аналітичних сервісів. Як ви розумієте, це відкриває захоплюючі можливості.

Що ж таке аналітична взаємодія «без людей»? Візьмемо, наприклад, «інтелектуального агента», обслуговуючого B2B-обмін в процесі пошуку партнерів по операції. Окрім іншого, пошук передбачає кількісне визначення ризків - процес, аналогічний оцінці ризиків при видачі кредиту. Кінцеве завдання агента — відповісти на питання, чи годиться партнер X для включення в операцію. Агент визначає важливі критерії (змінні і алгоритми) в запиті на оцінку риску. Цей запит може включати обчислення тимчасових рядів, ранжирування і схожу на Data Mining процедуру оцінки невідомих параметрів партнерів, але автором і споживачем запиту є агент, а не людина.

Більш того, інтелектуальний агент повинен володіти реальною аналітичною «майстерністю»: на підставі результатів одного запиту створити другий набір запитів, на підставі другого — третій і так далі. З сторони API це нагадує інтерактивний сеанс, а з точки зору користувача — систему з власним «інтелектом». До недавнього часу можливості організації взаємодії аналітичних «движків» були обмежені. Розробникам був доступний ряд створених виробниками API, а також технологія Microsoft OLE DB for OLAP (або ODBO), яку окрім Microsoft підтримували і інші сервери. Але протокол ODBO доступний лише в Win32, що навряд чи могло зробити його універсальним засобом доступу для Web-служб або корпоративних додатків середнього рівня. Java-версію створеного Радою з OLAP (OLAP Council) інтерфейсу MD-API не реалізував жоден постачальник серверних або клієнтських продуктів, тому для серверів додатків на основі відмінних від Win32 технологій мультивендорні API взагалі відсутні. Проте з'являються нові можливості. Перший інтерфейс називається XML for Analysis (XML/A), інший інтерфейс — Java OLAP Interface (JOLAP). Перша перевага цих інтерфейсів — підтримка додатків середнього рівня, створених не на Win32. Зрозуміло, що оба інтерфейси годяться і для інших цілей. Так, для клієнтських застосувань ці API як мінімум забезпечують доступ до раніше недоступній підмножині серверів.

XML/A — це API для доступу до інформації. Як і в OLE DB для OLAP, клієнт може запрошувати багатовимірну інформацію у вигляді набору записів реляційних даних, а також багатовимірних наборів кліток. Проте специфікації досить відкриті і забезпечують підключення інших типів провайдерів і доступ до них без порушення нормативів.

JOLAP заснован на Java, і тому всі аспекти цього API об'єктно-орієнтовані. Важливо, що не визначено жодної текстової мови запитів - замість цього в об'єктній моделі присутні класи, які можна комбінувати для визначення вибірки і розміщення результатів.

Таким чином, закономірною є думка багатьох дослідників та розробників OLAP – систем, що нове покоління засобів інтелектуального

аналізу даних розробляється саме для мережевих користувачів і оптимізовано для застосування в Інтернеті (у зв'язку з чим говорять про Інтернет - аналітику). Ці засоби дозволяють забезпечувати даними і інструментами їх аналізу користувачів мережі, для чого застосовуються особливі технології зберігання багатовимірних даних. До теперішнього часу практично всі провідні постачальники засобів аналізу даних пропонують системи з тонким клієнтом, орієнтовані на роботу в Інтернеті. У деяких з цих продуктів тонкий клієнт використовується для доступу до даних «традиційних» OLAP-серверів, в інших же - для обробки багатовимірних кубів, розміщених на Web-сайтах. Такі куби можна викачати з сайту або переслати по електронній пошті і помістити на локальний або мережевий диск. Об'єм даних, поміщених в мікрокуб стискається приблизно в 40 разів. Ефект досягається за рахунок нетрадиційних для OLAP підходів до зберігання даних. По-перше, в мікрокубах не розміщуються заздалегідь підраховані агрегати. Вони замінюються спеціальними механізмами індексування, які дозволяють обчислювати агрегати «на льоту». По-друге, значення всіх вимірювань зберігаються в мікрокубах в єдиних екземплярах. Нарешті, весь їх вміст архівується.

Одним з найвідоміших прикладів реалізації технологій інтелектуального аналізу даних в Інтернеті є проект провідного американського виробника засобів аналізу компанії Cognos для технологічної біржі Nasdaq. Російський аналог цього рішення - сервісний аналітичний центр для комерційних банків Banklist.ru, що використовує ПО компанії Intersoft Labs.

Рішення Nasdaq & Amex Companies Edition надає аналітичний сервіс учасникам біржі Nasdaq. Підписчики Nasdaq регулярно по електронній пошті отримують локальні куби з біржовими даними. У цих кубах в розрізі індексу Nasdaq, ринкової вартості компаній, річних доходів і інших вимірювань зберігаються показники ліквідності, прибутку і інші дані про діяльність понад 5000 компаній із списку Nasdaq і Amex, акції яких є предметом біржового торгу. Біржові дані поміщаються в куби за допомогою генератора Cognos Powerplay. Для OLAP-аналізу кубів і випуску звітів підписчикам надається

спеціально конфігурований для роботи з кубами OLAP-клієнт Cognos і пакет OLAP-звітів для перегляду і аналізу кубів.

Мета проекту Banklist.ru - забезпечити російським банкам, підприємствам і іншим зацікавленим особам і організаціям швидкий доступ до достовірної фінансової звітності комерційних банків і можливість її аналізу. Це необхідно для підвищення довіри до російських банок як з боку їх клієнтів, так і з боку діючих і потенційних партнерів. Проект створений за ініціативою Асоціації російських банків і реалізований під патронатом ЦБ РФ. Готове рішення організоване як Web-сайт, база даних якого містить мікрокуби аналітичної платформи "Контур". У цих мікрокубах зберігається фінансова інформація: баланси, звіти про прибутки і збитки і розрахунок обов'язкових економічних нормативів діяльності кредитних організацій. Вимірюваннями кубів є регіони, банки, балансові рахунки 1-го і 2-го порядку, статті доходів і витрат, банківські нормативи, а показниками - розгорнені залишки по рахунках, суми доходів, витрат і нормативів. Достовірна звітність 733 російських банків поступає з офіційного сайту ЦБ РФ.

Мікрокуби створюються генератором "Контур CubeMaker". Підписчикам сервісу Banklist.ru надається аналітична складова - OLAP-клієнт "Контур OlapBrowser". Це спеціальний web-браузер для OLAP-аналізу, який забезпечує доступ до мікрокубів через Інтернет і в локальній мережі, перегляд і аналіз даних OLAP-кубів, випуск звітів. З його допомогою можна знайти і відкрити для аналізу будь-який мікрокуб безпосередньо з Web-сайту або з диска призначеного для користувача ПК.

Обидва розглянуті рішення відносяться до класу OLAP-клієнтів з локальним кубом, тобто забезпечують відкладений (off-line) аналіз одного разу підготовлених файлів з даними на ПК користувача. Куби для аналізу можуть зберігатися на Web-сайті (варіант Banklist.ru) або на ПК користувача, як в рішенні для Nasdaq.

Принципова відмінність в технологічній реалізації цих проектів - розміщення метаданих. У разі Nasdaq користувачі отримують персональну

OLAP-систему, жорстко конфігуровану для роботи з PowerCube Nasdaq. Метадані зберігаються у файлах OLAP-звітів, налаштованих на куби Nasdaq. Звіти встановлюються разом із спеціальною запускаючою програмою Cognos Special Edition Launcher і OLAP-клієнтом Cognos Powerplay Special Edition. Це так званий «спеціальний випуск» продуктів Cognos, розроблений для проекту Nasdaq. Передвстановлена конфігурація робочого місця користувача системи аналізу біржових даних Nasdaq є істотним плюсом цього підходу.

«Контур»-мікрокуб містить метадані (опис вимірювань і показників, механізму агрегації і сортування даних, аналітичних інтерфейсів і т. д.) в самому собі. Такий підхід універсальний: зміна складу і даних мікрокубів не вимагає оновлення клієнтського ПО. Один і той же OLAP-клієнт може застосовуватися для аналізу OLAP-кубів різних прикладних сервісів. Це створює передумови для тиражування технології в різних галузях і знижує вартість володіння нею для користувачів.

Тести

1. Які економічні обставини, на Ваш погляд, обумовили появу сховищ даних?

а) Необхідність накопичення та оперативної обробки великих об’ємів даних.

б) Можливість інтегрувати дані з різних інформаційних систем.

в) Потреба формувати довільні запити до системи.

г) Розвиток систем автоматизації рутинних процесів.

д) Важливість аналізувати інформацію в тимчасовому аспекті.

2. Під сховищем даних слід розуміти …

а) предметно орієнтовані, інтегровані, незмінні набори даних, організовані для цілей підтримки управління;

б) предметно орієнтовані, інтегровані, динамічно змінюємі набори даних, організовані для цілей підтримки управління;

в) предметно орієнтовані, різнорідні, динамічно змінюємі набори даних, організовані для цілей підтримки управління.

3. Вашою концепцією при створені сховищ даних є …

а) ідея накопичення корпоративних даних, розсіяних по системах оперативної обробки даних, історичних архівах і інших зовнішніх джерелах;

б) ідея розподілення корпоративних даних, розсіяних по системах оперативної обробки даних, історичних архівах і інших зовнішніх джерелах;

в) ідея об'єднання корпоративних даних, розсіяних по системах оперативної обробки даних, історичних архівах і інших зовнішніх джерелах.

4. Переваги сховища даних порівняно з іншими системами обробки даних полягають в наступному:

а) міститься інформація за весь необхідний часовий інтервал, єдине представлення та єдиний доступ до даних, побудова довільних запитів та аналітичних звітів;

б) міститься інформація за весь необхідний часовий інтервал, єдине представлення та універсалізація доступу до даних, побудова довільних запитів та аналітичних звітів;

в) міститься інформація за весь необхідний часовий інтервал, єдине представлення та універсалізація доступу до даних, побудова наперед заданих запитів та аналітичних звітів.

5. Які процеси над даними відбуваються при створенні сховища даних?

а) дані очищаються, агрегуються, трансформуються, об'єднуються, синхронізуються;

б) дані очищаються, агрегуються, трансформуються, розподіляються, синхронізуються;

в) дані очищаються, накоплюються, трансформуються, об'єднуються, синхронізуються;

г) дані очищаються, накоплюються, трансформуються, розподіляються, зберігаються.

6. В Вашому сховищі даних повинні розміщатися слідуючи категорії даних:

а) оперативні дані, агреговані дані, детальні дані;

б) метадані, корпоративні дані, детальні дані;

в) метадані, агреговані дані, детальні дані.

7. Для отримання єдиної та цільної картини бізнес-даних Ви маєте застосувати слідуючи методи інтеграції даних в сховищі:

а) згортку, федералізацію і розповсюдження даних;

б) консолідацію, федералізацію і розповсюдження даних;

в) консолідацію, агрегацію і розповсюдження даних;

г) консолідацію, федералізацію і зберігання даних.

8. При створенні сховища даних Ви маєте можливість застосування слідуючих архітектур:

а) корпоративна інформаційна фабрика;

б) сховище даних з архітектурою кола;

в) сховище даних з архітектурою шини;

г) сховище даних з архітектурою «зірка»;

д) сховище даних з архітектурою «сніжинка».

9. Коли Ви зустрічаєте поняття вітрина або кіоск даних, то розумієте, що це ..

а) засіб розміщення всіх корпоративних даних, який будується для підрозділів усередині організації;

б) засіб розміщення підмножини корпоративних даних, який будується для всієї організації;

в) засіб розміщення підмножини корпоративних даних, який будується для підрозділів усередині організації.

10. Різницю між сховищем даних і вітриною даних Ви бачите в тому, що…

а) сховище даних створюється для розміщення всіх корпоративних даних, а вітрина даних дише для підмножини цих даних;

б) сховище даних створюється для розміщення частини корпоративних даних, а вітрина даних - для всіх цих даних;

в) як сховище даних так і вітрина даних створюються для розміщення всіх корпоративних даних.

11. Створюя вітрину (кіоск) даних по схемі «зірка», Ви розумієте, що така модель даних характеризується …

а) наявністю таблиці ключів, оточеної пов'язаними з нею таблицями розмірностей;

б) наявністю таблиці даних, оточеної пов'язаними з нею таблицями розмірностей;

в) наявністю таблиці фактів, оточеної пов'язаними з нею таблицями розмірностей.

12. Під розмірністю Ви зазвичай розумієте:

а) аспект, в розрізі якого можна отримувати, фільтрувати, групувати і відображати інформацію в базі даних;

б) аспект, в розрізі якого можна отримувати, фільтрувати, групувати і відображати інформацію про факти;

в) аспект, в розрізі якого можна отримувати, фільтрувати, групувати і відображати інформацію про дані.

13. Якщо при створенні вітрини (кіоска) даних мова іде про параметр факти, то Ви розумієте, що це …

а) текстові дані, що зберігаються в таблиці фактів і є предметом аналізу;

б) якісні величини, що зберігаються в таблиці фактів і є предметом аналізу;

в) числові величини, що зберігаються в таблиці фактів і є предметом аналізу.

14. В яких випадках слід використовувати таблиці покриттів при побудові вітрин даних?

а) таблиці покриття використовуються при моделюванні поєднання розмірностей, для яких відсутні факти;

б) таблиці покриття використовуються з метою моделювання розподілу розмірностей, для яких відсутні факти;

в) таблиці покриття використовуються з метою моделювання поєднання даних, для яких відсутні факти.

15. Ви використаєте схему «сніжинка» в випадках, коли …

а) потрібно нормалізувати схему бази даних;

б) потрібно нормалізувати схему "зірка";

в) потрібно нормалізувати схему сховища даних.

16. Поняття OLAP – технологія означає, що це є …

а) математичний аналіз даних для підтримки прийняття рішень;

б) якісний аналіз даних для підтримки прийняття рішень;

в) багатовимірний аналіз даних для підтримки прийняття рішень. +

17. Застосовуя гіперкуби даних, Ви розумієте, що це є …

а) OLAP – структура, створена з якісних даних;

б) OLAP – структура, створена з метаданих;

в) OLAP – структура, створена з робочих даних.

18. Термін агрегація даних для Вас означає:

а) виконання операцій над різнорідними даними з метою отримання більш загальних показників;

б) виконання операцій над однорідними даними з метою отримання більш загальних показників;

в) виконання операцій над всіма даними з метою отримання більш загальних показників.

19. Таблиця фактів в OLAP - аналізі містить …

а) відомості про об'єкти, сукупність яких надалі аналізуватиметься;

б) відомості про об'єкти, сукупність яких надалі зберігатиметься;

в) відомості про об'єкти, сукупність яких надалі видалятиметься.

20. Таблиця вимірювань в OLAP - аналізі містить …

а) сукупність даних про об’єкти аналізу;

б) незмінні або мало змінні дані;

в) постійно змінні дані.

Розділ 3.

НЕЙРОКОМП'ЮТЕРНІ ТЕХНОЛОГІЇ ТА МЕРЕЖІ

3.1. Поняття та можливості нейрокомп'ютерних технологій.

Впровадження нових наукоємких технологій в комерційній сфері - досить непроста справа, що вимагає, окрім грошей і часу, ще і деякої зміни психології. Проте, практика показує, що ці вкладення окупаються і виводять компанію на якісно новий рівень. В такій діяльності досить часто доводиться вирішувати різні задачі, причому їх постановка неформальна, а рішення неоднозначне. Відомо, що дохід компанії так чи інакше залежить від якості вирішення цих завдань. Навіть якщо строгий алгоритмічний підхід неможливий, а отримати точне рішення принципово не можливо, існують інші ефективні способи рішення. Важливе місце серед них займають нейрокомп'ютерні технології та нейронні мережі [8, 22, 32, 35].

Нейронні мережі - це адаптивні системи для обробки та інтелектуального аналізу даних, які є математичною структурою, що імітує деякі аспекти роботи людського мозку і демонструє такі його можливості, як здібність до неформального навчання, узагальнення і кластеризації некласифікованої інформації, здатність самостійно будувати прогнози на основі спостереження часових рядів. Головною їх відмінністю від інших методів є те, що нейромережі в принципі не потребують заздалегідь відомої моделі, а будують її самі лише на основі інформації, яку отримали. Саме тому нейронні мережі увійшли до практики усюди, де потрібно вирішувати задачі прогнозування, класифікації, управління - іншими словами, в області людської діяльності, де є задачі, що погано алгоритмізуються, для вирішення яких необхідні або постійна робота групи кваліфікованих експертів, або адаптивні системи автоматизації, якими і є нейронні мережі [39, 42, 45, 52].

Нейронна мережа приймає вхідну інформацію і аналізує її способом, аналогічним тому, що використовує наш мозок. Під час аналізу мережа вивчає (набуває досвіду і знання) і видає вихідну інформацію на основі придбаного раніше досвіду. Основне завдання аналітика, що використовує нейронні мережі для вирішення якої-небудь проблеми, - створити найбільш ефективну архітектуру нейронної мережі, тобто правильно вибрати вигляд нейронної мережі, алгоритм її навчання, кількість нейронів і види зв'язків між ними. Ця робота не має формалізованих процедур, вона вимагає глибокого розуміння різних видів архітектури нейронних мереж, включає багато дослідницької і аналітичної роботи, і може зайняти досить багато часу.

Для неформалізованих завдань нейромережеві моделі можуть на порядок перевершувати традиційні методи рішення. Але використання нейронних мереж доцільне, якщо:

- накопичені достатні об'єми даних про попередню поведінку системи;
- не існує традиційних методів або алгоритмів, що задовільно вирішують проблему;
- дані частково спотворені, частково суперечливі або не повні і тому традиційні методи видають незадовільний результат.

Нейронні мережі найкраще проявляють себе там, де є велика кількість вхідних даних, між якими існують неявні взаємозв'язки і закономірності. В цьому випадку нейромережі допоможуть автоматично врахувати різні нелінійні залежності, приховані в даних. Це особливо важливо в системах підтримки прийняття рішень і системах прогнозування. Нейромережі є незамінними при аналізі даних, зокрема, для попереднього аналізу або відбору, виявлення «випадних фактів» або грубих помилок людини, що приймає рішення. Доцільно використовувати нейромережеві методи в задачах з неповною або «зашумленною» інформацією, особливо в задачах, де рішення можна знайти інтуїтивно, і при цьому традиційні математичні моделі не дають бажаного результату.

Методи нейронних мереж можуть використовуватися незалежно або ж служити прекрасним доповненням до традиційних методів статистичного аналізу, більшість з яких пов'язані з побудовою моделей, заснованих на тих або інших припущеннях і теоретичних висновках (наприклад, що досліджуєма залежність є лінійною або що деяка змінна має нормальний розподіл). Нейромережевий підхід не пов'язаний з такими припущеннями - він однаково придатний для лінійних і складних нелінійних залежностей, особливо ж ефективний в розвідувальному аналізі даних, коли ставиться мета з'ясувати, чи є залежності між змінними. При цьому дані можуть бути неповними, суперечливими і навіть свідомо спотвореними. Якщо між вхідними і вихідними даними існує якийсь зв'язок, що навіть не виявляється традиційними кореляційними методами, то нейронна мережа здатна автоматично налаштуватися на його із заданою мірою точності. Крім того, сучасні нейронні мережі володіють додатковими можливостями: вони дозволяють оцінювати порівняльну важливість різних видів вхідної інформації, зменшувати її об'єм без втрати істотних даних, розпізнавати симптоми наближення критичних ситуацій і так далі.

Штучні нейронні мережі породжені біологією, оскільки вони складаються з елементів, функціональні можливості яких аналогічні більшості елементарних функцій біологічного нейрона. Ці елементи потім організовуються за способом, який може відповідати (або не відповідати) анатомії мозку. Не дивлячись на таку поверхневу схожість, штучні нейронні мережі демонструють деякі властивості мозку. Наприклад, вони вивчаються на основі досвіду, узагальнюють попередні прецеденти на нові випадки і витягують істотні властивості з інформації, що поступає та містить зайві дані. Проте, навіть найоптимістичніший їх захисник не передбачить, що в недалекому майбутньому штучні нейронні мережі дублюватимуть функції людського мозку. Реальний «інтелект», що демонструється найскладнішими нейронними мережами, знаходиться нижчим за рівень дощового черв'яка, і ентузіазм має бути помірний відповідно до сучасних реалій. Але рівним чином

було б невірним ігнорувати дивну схожість у функціонуванні деяких нейронних мереж з людським мозком. Ці можливості, як би вони не були обмежені сьогодні, наводять на думку, що глибоке проникнення в людський інтелект, а також безліч революційних застосувань, можуть бути не за горами.

Для повного розуміння сутності нейрокомп'ютерних технологій розглянемо як мозок обробляє інформацію. Мозок побудований з кліток двох типів: гліальних і нейронів. І хоча роль глії в його роботі досить значна, більшість дослідників вважають, що в основному розуміння роботи мозку може бути досягнуте при вивченні нейронів, об'єднаних в єдину зв'язану мережу. Ця парадигма і використовується при побудові, вивченні і вживанні штучних нейронних мереж. Слідує, проте, відмітити, що є і інші точки зору. Зокрема, такі учені як Пенроуз і Хамерофф вважають, що головні події відбуваються не в нейронній мережі, а в самих клітинах, а саме в їх цитоскелетоні, в так званих мікротрубочках. Згідно їх точці зору, і пам'ять, і навіть свідомість визначаються конформаційними змінами білків у внутріклітинних структурах і пов'язаними з ними квантовими ефектами.

Кількість нейронів в мозку оцінюється величиною 10^{10} - 10^{11}. Типові нейрони мають тіло клітини (сому), безліч коротких розгалуженних відростків - дендритів і єдиний довгий і тонкий відросток - аксон. На кінці аксон також розгалужується і утворює контакти з дендритами інших нейронів - синапси (рис. 3.1).

Внутріклітинний простір нейрона має негативний електричний потенціал по відношенню до позаклітинного (-70 mV), тобто клітина в цілому поляризована. Поляризація виникає за рахунок вибіркової проникності клітинної мембрани для іонів натрію і калія, що приводить до різниці їх концентрацій усередині і поза кліткою. Проте, якщо зовні досить сильно змінити потенціал мембрани одного нейрона (передавача) поблизу виходу аксона з його клітинного тіла, то проникність мембрани міняється і перерозподіл іонів у внутріклітинному і позаклітинному просторі аксона приводить до поширення по ньому хвилі короткочасної деполяризації.

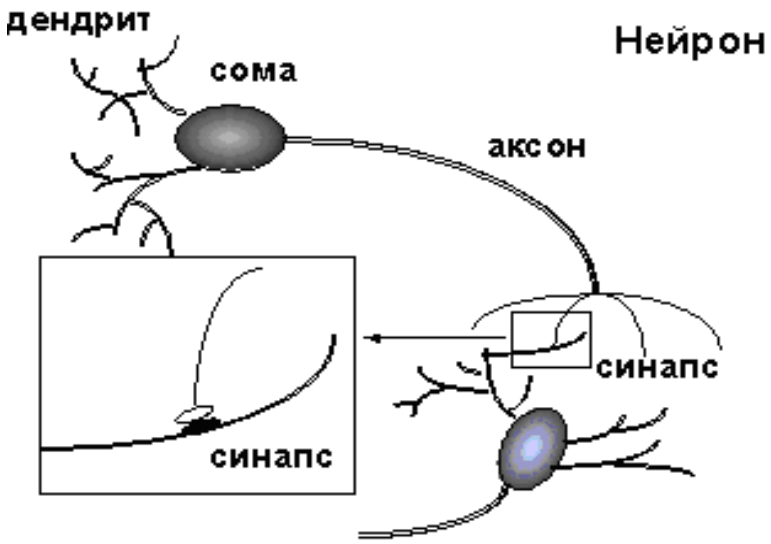


Рис. 3.1. Схема нейрона та міжнейронної взаємодії.

Електричний імпульс, поширившись по всіх галуженнях закінчення аксона з швидкістю близько 100 м/с, досягає синапсів, розташованих в місцях його контакту з дендритом або сомой інших кліток. Під впливом цього імпульсу в синапсах виділяються спеціальні хімічні речовини - нейромедіатори, які, перетинаючи синаптическую щілину, взаємодіють з мембраною нейрона-приймача і змінюють її потенціал. Таким чином дія передається від одного нейрона до інших. Відмітимо, що ця дія може бути як збуджуючою - сприяючою подальшій генерації хвилі деполяризації в нейроні-приймачі, так і такою, що інгібірує - що перешкоджає такій генерації. Тип дії визначається хімічною природою нейромедіатора, що виділяється в синапсі. Після генерації імпульсу нейрон деякий час (період рефрактерності) не може активуватися. Тому частота, з якою нейрон може генерувати імпульси обмежується приблизно 100 Гц. Кожен з нейронів встановлює синаптичні зв'язки в середньому з 104 іншими нейронами. Тому число зв'язків в мозку оцінюється в 10^{14} - 10^{15}. До цих пір невідомо, яким кодом користується нервова система для передачі взаємодії. Можливо, він є бінарним, і значення мають вказані стани нейронів. Можливо, важлива частота електричної активності нейронів, що кодує інтенсивність сигналу. Наприклад, у нейронів кори ця частота може бути

пропорційна вірогідності деякої події. Нарешті, інформація може міститися не в імпульсних процесах, а в повільніших змінах потенціалу мембрани, які не завжди активують клітку (тобто не перевищують порогу активації). Проте при будь-якому припущенні модель мережі взаємодіючих нейронів виявляється виключно багатою і такою, що володіє властивостями, які можна зіставити з реальними можливостями мозку [22, 32, 35].

Нескладно побудувати математичну модель вищеописанного процесу (рис. 3.2).

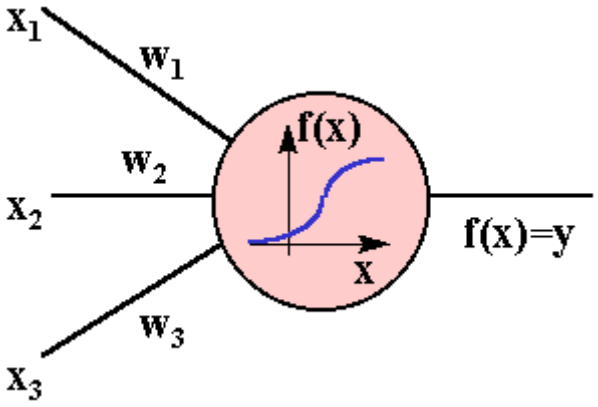


Рис. 3.2. Модель нейрона.

На рисунку зображена модель нейрона з трьома входами (дендритами), причому синапси цих дендритів мають ваги w_1, w_2, w_3. Хай до синапсів поступають імпульси сили x_1, x_2, x_3 відповідно, тоді після проходження синапсів і дендритів до нейрона поступають імпульси w_1x_1, w_2x_2, w_3x_3. Нейрон перетворює отриманий сумарний імпульс $x = w_1x_1 + w_2x_2 + w_3x_3$ відповідно до деякої передавальної функції $f(x)$. Сила вихідного імпульсу дорівнює $y = f(x) = f(w_1x_1 + w_2x_2 + w_3x_3)$. Таким чином, нейрон повністю описується своїми вагами w_k і передавальною функцією $f(x)$. Отримавши набір чисел (вектор) x_k як входи, нейрон видає деяке число y на виході.

Нелінійна функція $f(x)$ також називається активаційною і може мати різний вигляд, як показано на малюнку 3.3. Однією з найбільш поширених є

нелінійна функція з насиченням, так звана логістична функція або сигмоїд (функція S-образного вигляду):

$$f(x) = \frac{1}{1+e^{-\alpha x}}$$

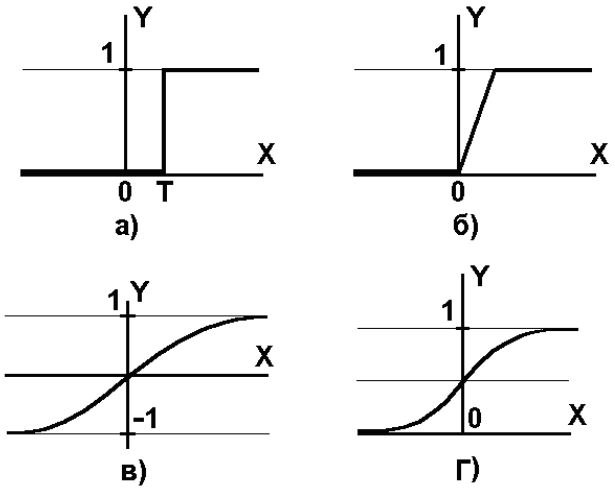


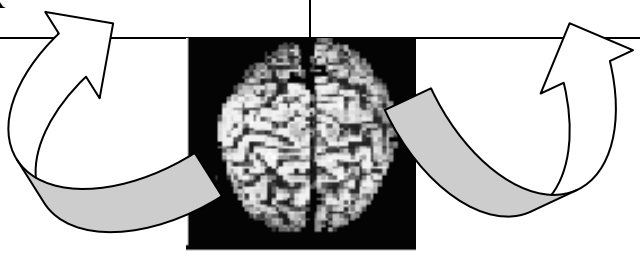
Рис. 3.3. Вигляд активаційної функції: а) функція одиничного стрибка; б) лінійний поріг (гистерезис); в) сигмоїд – гіперболічний тангенс; г) сигмоїд – функція з насиченням.

При зменшенні α сигмоїд стає пологішим і при $\alpha = 0$ вироджується в горизонтальну лінію на рівні 0.5, при збільшенні α сигмоїд наближається по зовнішньому вигляду до функції одиничного стрибка з порогом T в точці $x = 0$. З виразу для сигмоїда очевидно, що вихідне значення нейрона лежить в діапазоні [0,1].

Нейрокомп'ютерні технології, як ми бачимо, зовсім не покликані замінити існуючі традиційні обчислювальні машини. Вони лише заповнюють ті можливості, для яких не вдається побудувати формальних алгоритмічних схем. Подібно до того, як в людському мозку ліва і права півкуля працюють в діалозі і спільно, сучасні інформаційні системи повинні використовувати симбіоз традиційних комп'ютерів і нейротехнологій для повноцінного і продуктивного інтелектуального аналізу інформації (рис. 3.4).

Як відомо, існує дві парадигми обробки інформації – «логічна» і «образна». Перша домінує в існуючих комп'ютерах, друга - лежить в основі роботи мозку, хоча людський мозок відрізняє від мозку інших тварин наявністю обох компонент мислення. Для повноцінного існування в навколишньому світі цінні обидва способи обробки інформації. І вони рано чи пізно виникають - як в ході біологічної еволюції, так і в процесі еволюції комп'ютерів, але, що характерно, - в різній послідовності. Біологічна еволюція йшла «від образів - до логіки». Комп'ютери ж, навпаки, почавши з логіки, лише через декілька десятиліть починають освоювати розпізнавання образів.

Традиційні ЕОМ	**Нейрокомп'ютери**
• Послідовні • Заданий алгоритм • Ієрархична структура алгоритмів та розподіл складної задачі на декілька простих	• Паралельні • Алгоритм формується шляхом навчання • Операції з образами



Ліва півкуля ***Права півкуля***

Рис. 3.4. Симбіоз традиційних та нейрокомп'ютерів як симбіоз правої та лівої півкулі головного мозоку.

Вузьким місцем, що знижує ефективність людино-машинного симбіозу, є нездатність сучасних ЕОМ оперувати сенсорною або, більш загально образною інформацією. Людина, мозок якої орієнтований саме на такого роду інформацію, є зараз єдиною сполучною ланкою між світом абстрактних символів, що переробляються комп'ютерами, і зовнішнім світом. Нездатність комп'ютерів бачити, чути і відчувати не дозволяє їм звільнити людей від їх

теперішніх обов'язків «універсальних маніпуляторів» і контролерів при машинному виробництві. Відсутність сенсорного сприйняття миру комп'ютерами робить доступні їм моделі світу безпорадними. Між тим, вже зараз вартість комп'ютерної обробки інформації і вартість людського мислення майже порівнялися. У мозку людини близько 10^{10} нейронів, з яких одночасно активізоване приблизно 10^{8}, працюючих з характерною частотою 10^{2} Гц. Приймаючи в якості зарплати «білого комірця» $30,000 в рік отримаємо оцінку вартості обробки інформації людиною:

- *Людина*: (10^{10} оп/с*3*10^{7} с/рік) / (30 000 $/рік)=$10^{13}$ оп/$.

Обчислення на сучасному персональному комп'ютері продуктивністю 10^{7} оп/с при амортизації близько $300 в рік стоять лише на порядок менше:

- *Універсальний процесор:* (10^{7} оп/с *3*10^{7} с/рік) / (300 $/рік) = 10^{12} оп/$.

Спеціалізовані процесори, як правило, дають додатковий виграш у вартості обчислень приблизно на порядок:

- *Спеціалізований процесор:* (10^{8} оп/с*3*10^{7} с/рік) / (300 $/рік) = 10^{13} оп/$,

приблизно порівнюючись з людиною при такому способі зіставлення.

Отже, стає економічно доцільним перекласти всі рутинні людські функції на комп'ютери: все, що може бути формалізоване негайно перетворюється на програмні продукти і включається у виробничий процес. Проте, нарощування темпів комп'ютеризації натрапляє на обмежені можливості сучасних комп'ютерів в розв’язанні неалгоритмізованих задач - обробці образів. Це і є те вузьке місце, яке зараз різко звужує можливі області застосування комп'ютерів і, відповідно, - інтелектуальних аналітичних систем. Штучні нейромережі покликані «розшити» це вузьке місце, забезпечивши комп'ютерам здатність оперувати образною інформацією.

Нейрокомп’ютінг є новою парадигмою обчислювальних систем. Основне завдання нейрокомп’ютерів - обробка образів, заснована на навчанні, - те ж, що і у біологічних нейросистем. Подібно до біологічних, штучні нейромережі націлені на паралельну обробку образів. У новій схемотехніці, як і в мозку, відсутні загальні шини, немає розділення на активний процесор і

пасивну пам'ять. Обчислення, як і навчання, розподілені по всіх активних елементах - нейронах, кожний з яких є елементарний процесор образів, оскільки проводить хоч і просту операцію, але відразу над великою кількістю входів. Як обчислення, так і навчання повністю паралельні. У цьому сила природних нейрокомп’ютерів. Це дає можливість вирішувати задачі, непосильні навіть наймогутнішим суперкомп'ютерам, не дивлячись на багато кратну різницю в швидкодії елементної бази.

На думку Anil K. Jain з Мічиганського університету і фахівців Дослідницького центру IBM Jianchang Mao і K. M. Mohiuddin, список таких задач можна класифікувати таким чином [32, 35, 52].

Класифікація образів. Завдання полягає у вказівці приналежності вхідного образу (наприклад, мовного сигналу або рукописного символу), представленого вектором ознак, одному або декільком заздалегідь певним класам. До відомих додатків відносяться розпізнавання букв, розпізнавання мови, класифікація сигналу електрокардіограми, класифікація кліток крові, забезпечення діяльності біометричних сканерів і т.п.

Кластеризація/категоризація. При рішенні задачі кластеризації, яка відома також як класифікація образів «без вчителя» де відсутня навчальна вибірка з мітками класів. Алгоритм кластеризації заснований на подібності образів і розміщує близькі образи в один кластер. Відомі випадки застосування кластеризації для витягання знань, стиснення даних і дослідження властивостей даних.

Апроксимація функцій. Припустимо, що є навчальна вибірка пар даних вхід-вихід, яка генерується невідомою функцією, залежною від деякого аргументу, спотвореною шумом. Завдання апроксимації полягає в знаходженні оцінки невідомої функції. Апроксимація функцій необхідна при рішенні численних інженерних і наукових задач моделювання. Типовим прикладом є шумозаглушення при прийомі сигналу різної природи, незалежно від переданої інформації.

Прогноз. Хай задані n дискретних відліків $\{y(t_1), y(t_2),..., y(t_n)\}$ в послідовні моменти часу $t_1, t_2,..., t_n$. Завдання полягає в прогнозі значення $y(t_{n+1})$ в деякий майбутній момент часу t_{n+1}. Прогнози мають значний вплив на ухвалення рішень в бізнесі, науці і техніці. Прогноз цін на фондовій біржі і прогноз погоди є типовими додатками техніки прогнозу.

Оптимізація. Численні проблеми в математиці, статистиці, техніці, науці, медицині і економіці можуть розглядатися як проблеми оптимізації. Завданням алгоритму оптимізації є знаходження такого рішення, яке задовольняє системі обмежень і максимізує або мінімізує цільову функцію. Задача комівояжера, задача про призначення, транспортна задача є класичними прикладами задач оптимізації.

Пам'ять, що адресується за змістом (асоціативна пам'ять). У моделі обчислень фон Неймана звернення до пам'яті доступно тільки за допомогою адреси, яка не залежить від змісту пам'яті. Більш того, якщо допущена помилка в обчисленні адреси, то може бути знайдена абсолютно інша інформація. Асоціативна пам'ять доступна по вказівці заданого змісту. Вміст пам'яті може бути викликаний навіть по частковому входу або спотвореному змісту. Асоціативна пам'ять може чудово знайти застосування при створенні мультимедійних інформаційних баз даних.

Управління. Розглянемо динамічну систему, задану сукупністю векторів вхідної дії, поточного стану та вихідного вектора системи. Класичною постановкою задачі є розрахунок управляючої дії, щоб в конкретний момент часу система знаходилася в певній бажаній точці.

Слід відзначити, що в загальному випадку найчастіше зустрічаються ситуації, що є комплексом визначених раніше.

Розглянемо парадигми нейрокомп’ютінга, тобто, родові риси, об'єднуючі принципи роботи і навчання всіх нейрокомп’ютерів. Головне, що їх об'єднує - націленість на обробку образів. Сформулюємо ці парадигми в концентрованому вигляді безвідносно до біологічних прототипів, як способи обробки даних.

- Коннекціонізм.

Відмінною рисою нейромереж є глобальність зв'язків. Базові елементи штучних нейромереж - формальні нейрони - спочатку націлені на роботу з широкосмуговою інформацією. Кожен нейрон нейромережі, як правило, пов'язаний зі всіма нейронами попереднього шару обробки даних (рис. 3.5), що ілюструє найбільш широко поширену в сучасних додатках архітектуру багатошарового персептрона. У цьому основна відмінність формальних нейронів від базових елементів послідовних ЕОМ - логічних вентилів, що мають лише два входи.

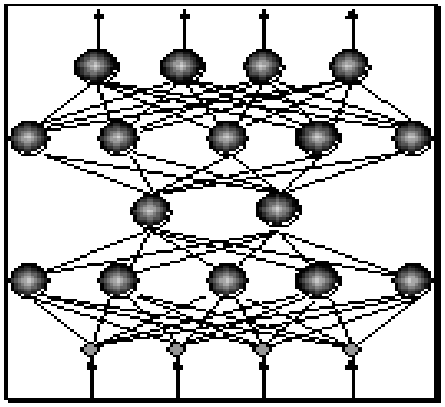


Рис. 3.5. Глобальний зв'язок в штучних нейромережах.

У результаті, універсальні процесори мають складну архітектуру, засновану на ієрархії модулів, кожний з яких виконує строго певну функцію. Навпаки, архітектура нейромереж проста і універсальна. Спеціалізація зв'язків виникає на етапі їх навчання під впливом конкретних даних.

Типовий формальний нейрон проводить просту операцію - зважує значення своїх входів з своїми ж вагами, що локально зберігаються, і проводить над їх сумою нелінійне перетворення (рис. 3.6):

$$y = f(u),\ u = u_0 + \sum_i u_i x_i$$

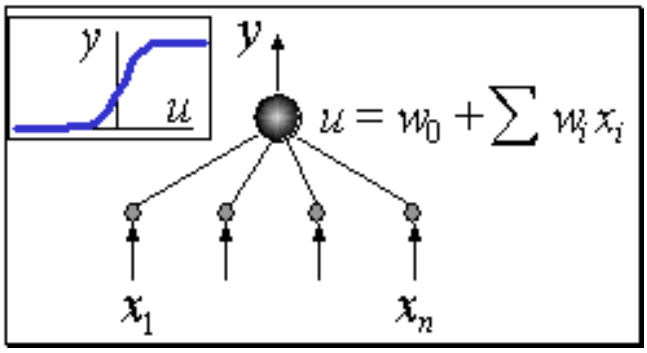


Рис. 3.6. Нелінійна операція над лінійною комбінацією входів.

Нелінійність вихідної функції активації принципова. Якби нейрони були лінійними елементами, то будь-яка послідовність нейронів також проводила б лінійне перетворення, і вся нейромережа була б еквівалентна одному нейрону (або одному шару нейронів - у разі декількох виходів). Нелінійність руйнує лінійну суперпозицію і призводить до того, що можливості нейромережі істотно вищі за можливості окремих нейронів.

- Локальність і паралелізм обчислень.

Масовий паралелізм нейрообчислювань, необхідний для ефективної обробки образів, забезпечується локальністю обробки інформації в нейромережах. Кожен нейрон реагує лише на локальну інформацію, що поступає до нього в даний момент від пов'язаних з ним таких же нейронів, без апеляції до загального плану обчислень, звичайною для універсальних ЕОМ. Таким чином, нейромережеві алгоритми локальні, і нейрони здатні функціонувати паралельно.

- Програмування: навчання, засноване на даних.

Відсутність глобального плану обчислень в нейромережах припускає і особливий характер їх програмування. Воно також носить локальний характер: кожен нейрон змінює свої параметри - синаптічні ваги - відповідно до локальної інформації, що поступає до нього, про ефективність роботи всієї мережі як цілого. Режим розповсюдження такої інформації по мережі і відповідної до неї адаптації нейронів носить характер навчання. Такий спосіб програмування дозволяє ефективно врахувати специфіку потрібного від мережі способу обробки даних, бо алгоритм не задається наперед, а породжується

самими даними - прикладами, на яких мережа навчається. Саме таким чином в процесі самонавчання біологічні нейромережі виробили такі ефективні алгоритми обробки сенсорної інформації.

Характерною особливістю нейромереж є їх здібність до узагальнення, що дозволяє навчати мережу на нікчемній частці всіх можливих ситуацій, з якими їй, можливо, доведеться зіткнутися в процесі функціонування. У цьому їх разюча відмінність від звичайних ЕОМ, програма яких повинна наперед передбачати їх поведінку у всіх можливих ситуаціях. Ця ж їх здатність дозволяє кардинально здешевити процес розробки додатків.

- Універсальність навчальних алгоритмів.

Привабливою рисою нейрокомп'ютінга є єдиний принцип навчання нейромереж - мінімізація емпіричної помилки. Функція помилки, що оцінює дану конфігурацію мережі, задається ззовні - залежно від того, яку мету переслідує навчання. Але далі мережа починає поступово модифікувати свою конфігурацію - стан всіх своїх синаптичних ваг - так, щоб мінімізувати цю помилку. У результаті, в процесі навчання мережа все краще справляється з покладеним на неї завданням.

Не вдаючись до математичних тонкощів, образно цей процес можна уявити собі як пошук мінімуму функції помилки $E(u)$, залежної від набору всіх синаптичних ваг мережі w(рис. 3.7).

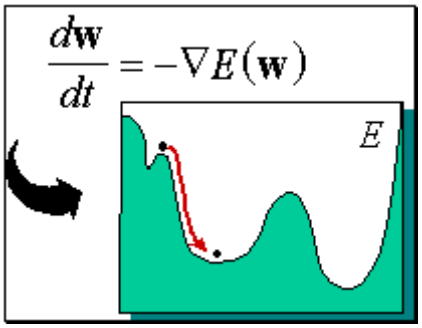


Рис. 3.7. Навчання мережі як задача оптимізації.

Базовою ідеєю всіх алгоритмів навчання є облік локального градієнта в просторі конфігурацій для вибору траєкторії якнайшвидшого спуску по функції помилки. Функція помилки, проте, може мати безліч локальних мінімумів, що представляють суб-оптімальні рішення. Тому градієнтні методи зазвичай доповнюються елементами стохастичної оптимізації, щоб запобігти застряванню конфігурації мережі в таких локальних мінімумах. Ідеальний метод навчання повинен знайти глобальний оптимум конфігурації мережі.

Вписавши появу нейрокомп’ютинга в загальний процес еволюції комп'ютерів, ми дістаємо можливість заглянути в найближче майбутнє - екстраполюючи сьогоднішні тенденції.

1. «Розумні» нейрочіпи.

Сьогоднішній нейрокомп’ютинг проходит «обкатку», в основному, в програмному продукті для задач асоціативної обробки даних, рідко використовуючи при цьому свій «паралельний» потенціал. Епоха паралельного нейрокомп’ютинга почнеться з виходом на ринок широкого асортименту апаратних засобів - спеціалізованих нейрочіпів для обробки зображень, мови, аналітики та сенсорної інформації. Вже сьогодні існують, наприклад, дверні системи, що розпізнають господаря по вигляду, голосу, і мабуть, запаху в сукупності. Системи життєзабезпечення житла стануть адаптивними і зможуть навчатися. Всі побутові прилади порозумнішають і придбають здатність вгадувати, що від них вимагається саме в даний момент. Сенсорні датчики придбають здатність реагувати, а регулюючі системи - відчувати. Розумні контроллери, що розпізнають потенційно небезпечні ситуації і що уміють приймати адекватні превентивні рішення, набудуть поширення в складних електричних і теплових мережах. На них грунтуватимуться системи регулювання транспортними потоками і потоками даних в комп'ютерних мережах і стільниковому зв'язку.

2. Операційні системи нової архітектури.

Проте, найбільші зміни торкнуться самих комп'ютерів. На думку Біла Гейтса, глави відомої компанії Microsoft, висловленому їм на зборах ради

директорів, через 10 років 90% операційних систем буде зайнято вирішенням задач розпізнавання образів. Таким чином, при проектуванні майбутніх поколінь комп'ютерів нейрокомп’ютинг висувається на перший план. Можна навіть уявити собі зразковий сценарій проникнення нейросистем в комп'ютери майбутнього, пов'язаний з розвитком глобальної мережі Internet. Зараз саме вона направляє розвиток комп'ютерних систем, поступово перетворюючи розрізнену мережу персоналок, робочих станцій і мейнфреймів в єдиний світовий мережевий комп'ютер з необмеженими інформаційними ресурсами. І подібно до того, як епоха великих відкриттів XV століття стимулювала розвиток астрономії і точної механіки для вдосконалення навігаційних приладів, освоєння нового інформаційного океану вимагає розвитку нових засобів навігації - асоціативного пошуку, створення адаптивних і автономних агентів. Новий інтерфейс користувача грунтуватиметься на агентах, що представляють його інтереси в мережі. Цей новий вигляд програмного забезпечення, що отримав назву agentware, претендує на центральне місце в майбутній системі людино-машинного спілкування. Тим часом, перші екземпляри agentware вже з'явилися на ринку, і що характерно, багато з них засновані на технології нейромереж. Це, мабуть, сьогодні найкоротша дорога до створення легко навчаємих автономних електронних секретарів. Природно передбачити, що саме на цьому напрямі, через його стратегічну важливість, в найближчому майбутньому буде досягнутий найбільший прогрес.

Розглянемо сучасний стан нейрокомп’ютингу та нейротехнологій.

- Елементна база нейрокомп'ютерів.

Практично всі нейрокомп'ютери, що діють, використовують традиційну елементну базу: мікроелектронні СБІС. Сотні мільярдів доларів, які вже вкладені в розвиток цієї технології, дають їй вирішальну перевагу перед іншими альтернативами, такими, як, наприклад, оптичні обчислення. Сучасна електроніка спирається, в основному, на цифрову обробку сигналів, стійку до перешкод і технологічних відхилень в параметрах базових злементов. Цифрова схемотехніка надає нейро-конструкторам найбільш багатий інструментарій.

Тому недивно, що найбільшого поширення набули саме цифрові нейрокомп'ютери. Це по суті - спеціалізовані матричні прискорювачі, що використовують матричний, пошаровий характер обробки сигналів в нейромережах. Широко використовуються стандартні процесори обробки сигналів (DSP - Digital Signal Processors), оптимізовані під такі операції. Прикладом сучасного DSP-процессора, пристосованого для прискорення нейро-обчислень є продукт Texas Instruments TMS320C80 продуктивністю 2 млрд. операцій в секунду. Цей кристал включає п'ять процесорів і реалізує відразу дві технології - DSP і RISC (рис. 3.8).

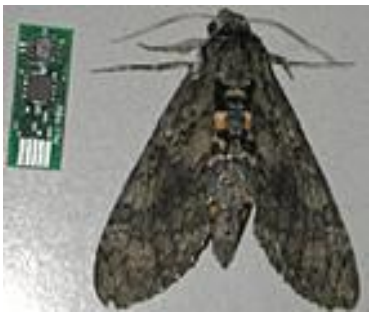


Рис. 3.8. Сучасні цифрові нейрокомп’ютерні елементи.

Проте, сама природа нейромережевої обробки інформації - аналогова, і додаткового виграшу в швидкості обчислень (по деяких оцінках 10^3-10^4) та щільності обчислювальних елементів можна добитися, використовуючи спеціалізовану аналогову елементну базу. Найбільш перспективні, мабуть аналогові мікросхеми з локальними зв'язками між елементами (т.з. клітинні нейромережі, CNN - Cellular Neural Networks), наприклад силіконова ретина фірми Synaptics. З іншого боку, розробка аналогових чіпів з використанням нетрадиційних рішень схемотехнік вимагає додаткових і чималих витрат. В даний час ці роботи розгорнуті широким фронтом, наприклад, в рамках проекту SCX-1 (Silicon Cortex - кремнієва кора) (рис. 3.9).

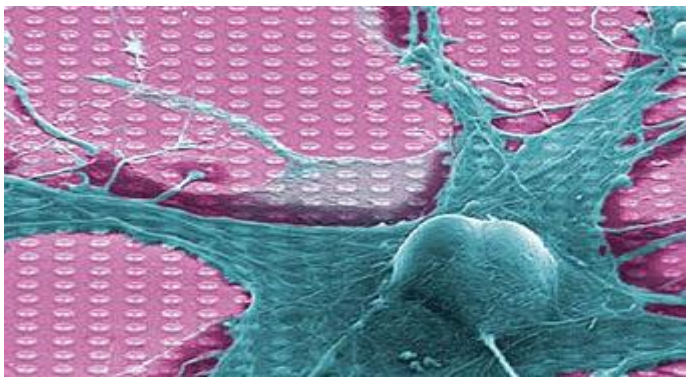


Рис. 3.9. Сучасні аналогові нейрокомп’ютерні елементи.

Переваги обох підходів намагаються поєднати гібридні мікросхеми, що мають цифровий інтерфейс з апаратурою, але виконують найбільш масові операції аналоговим способом.

- Архітектура нейрокомп'ютерів.

Перевага нейрокомп’ютинга полягає в можливості організувати масові паралельні обчислення. Тому базові процесорні елементи зазвичай поєднуються в обчислювальні комплекси: якомога більше - на одному чіпі, а що не помістилося - в мультипроцесорні плати. Ці плати потім або вставляють в персональні комп'ютери і робочі станції як нейро-прискорювачі, або збирають в повномасштабні нейрокомп'ютери.

- Порівняння вартості звичайних і нейрообчислень.

Продуктивність сучасних персональних комп'ютерів складає приблизно 10^7 операцій. Отже, при вартості всього на порядок більше звичайних PC, нейро-прискорювач в декілька сотень разів перевершує їх в швидкості. Таким чином, питома вартість сучасних нейрообчислень приблизно на порядок нижча, ніж в традиційних комп'ютерів. Це всього лише наслідок спеціалізації матричних процесорів (DSP), що мають ту ж елементну базу, що і універсальні мікропроцесори. Проте, виграш на один порядок у вартості обчислень рідко коли здатний стати вирішальним аргументом для використання спеціалізованої апаратури, зв'язаної з додатковими витратами, у тому числі на навчання персоналу. Тому реально нейрокомп'ютери використовуються в

спеціалізованих системах, коли потрібно навчати і постійно перенавчати сотні нейромереж, об'єднаних в єдині інформаційні комплекси, або в системах реального часу, де швидкість обробки даних критична.

- Нейроемулятори.

Більшість же прикладних систем нейромережевого аналізу даних використовують емуляцію нейромереж на звичайних комп'ютерах, зокрема на PC. Такі програми отримали назву нейроемуляторів. Доступність і збільшені обчислювальні можливості сучасних комп'ютерів привели до широкого поширення програм, що використовують принципи нейромережевої обробки даних, але виконуються на послідовних комп'ютерах. Цей підхід не використовує переваг властивого нейрообчисленням паралелізму, орієнтуючись виключно на здатність нейромереж вирішувати завдання, що не формалізуються .

Переваги таких «віртуальних» нейрокомп'ютерів для відносно невеликих завдань очевидні. По-перше, не треба витрачатися на нову апаратуру, якщо можна завантажити вже наявні комп'ютери загального призначення. По-друге, користувач не повинен освоювати особливості програмування на спецпроцесорах і способи їх поєднання з базовим комп'ютером. Нарешті, універсальні ЕОМ не накладають жодних обмежень на структуру мереж і способи їх навчання, тоді як спецпроцесори частенько мають обмежений набір «зашитих» в них функцій активації і досягають пікової продуктивності лише на певному колі задач.

- Готові нейропакети.

Це закінчені незалежні програмні продукти, призначені для широкого класу задач, в основному - для передбачень і статистичної обробки даних. Більшість з нейропакетів, що є на ринку, має дружній інтерфейс користувача, що не вимагає знайомства з мовами програмування. Безліч нейроемуляторів початкового рівня можна знайти в Internet як shareware або freeware. Це, зазвичай, багатошарові персептрони з одним або декількома правилами навчання. Виключення складає повністю професійний Штутгартський

симулятор з великим набором можливостей, що працює, правда, лише на UNIX-машинах. Комерційні пакети відрізняються від вільно поширюваних великим набором засобів імпорту і передобробки даних, додатковими можливостями по аналізу значущості входів і оптимізації структури мережі. Як правило, такі пакети (BrainMaker Professional, NeuroForecaster, Лора-IQ300) мають власний вбудований блок передобробки даних, хоча інколи для цієї мети зручніше використовувати стандартні електронні таблиці. Так, нейро-продукти групи нейрокомп'ютинга ФІАН вбудовується безпосередньо в Microsoft Excel як спеціалізовані функції обробки даних. При цьому всю передобробку даних і візуалізацію результатів можна проводити стандартними засобами Excel, який, крім того, має багатий і розширюваний набір конверторів для імпорту і експорту даних. Такі пакети націлені на вирішення інформаційних задах в діалоговому режимі - при безпосередній участі користувача.

- Інструменти розробки нейрододатків.

Головне, що відрізняє цей клас програмного забезпечення - здатність генерувати «відчуджувані» нейромережеві продукти, тобто генерувати програмний код, що використовує навчені нейромережі для аналізу даних. Такий код може бути вбудований як підсистема в будь-які скільки завгодно складні інформаційні комплекси. Прикладами подібних систем, здатних генерувати вихідні тексти програм є NeuralWorks Professional II Plus фірми NeuralWare і Neural Bench. Остання цікава, окрім іншого, тим, що може генерувати коди на багатьох мовах, включаючи Java. Такі Java-апплеты можуть використовуватися для організація різного роду сервісів в глобальних і локальних мережах. Зручним інструментом розробки складних нейросистем є MATLAB з нейромережевим інструментарієм, що додається до нього. MATLAB надає зручне середовище для синтезу нейромережевих методик з іншими методами обробки даних (wavelet-анализ, статистика, фінансовий аналіз і так далі).

- Готові рішення на основі нейромереж.

В таких системах нейромережі заховані від користувача в надрах готових автоматизованих комплексів, призначених для вирішення конкретних аналітичних завдань. Наприклад, вже згадуваний продукт Falcon вбудовується в банківську автоматизовану систему обслуговування платежів за пластиковими картками. У іншому випадку це буде автоматизована система управління заводом або реактором. Кінцевого користувача, як правило, не цікавить спосіб досягнення результату, йому важлива лише якість продукту. Оскільки багато таких готових рішень володіють унікальними можливостями і забезпечують реальні конкурентні переваги, їх ціна може бути досить висока. Іспанська компанія SEMP займається підвищенням ефективності обслуговування кредитних карт VISA, що емітуються іспанськими банками. Кількість подібних транзакцій - від 500000 до 1000000 в день. Нейромережева система, розроблена для неї вченими з Мадридського Інституту Інженерії Знань (Instituto de Ingenieria del Conocimiento), зменшила вірогідність несанкціонованого використання карт на 30-40% для основних каналів шахрайства.

- Нейромережевий консалтинг.

Опис ринку нейро-продуктів буде не повним без згадки про нейро-консалтинг. Замість того, щоб продавати готові програми або інструменти для їх розробки, можна торгувати і послугами. Деякі завдання, наприклад такі, як передбачення ринкових часових рядів, є настільки складними, що доступні лише справжнім професіоналам. Не кожна компанія може дозволити собі витрати, що асоціюються з передовими науковими розробками. Тому набувають популярності фірми, єдиною продукцією яких є передбачення ринків. При великому числі клієнтів ціна таких передбачень може бути вельми помірною. Прикладом тут може служити Prediction Company, заснована в 1991 році фізиками Дойном Фармером і Норманом Паккардом, - фахівцями в області динамічного хаосу. Продукція компанії користується великим успіхом серед Швейцарських банків, що скуповують прогнози для гри на фондових і валютних ринках.

- Ринок нейропродукції.

Об'єм ринку нейропродукції, структуру зростає стрімкими темпами: по різних оцінках - від 30% до 50% в рік, перевалivши недавно за декілька мільярдів доларів. Таке ж зростання спостерігалося на початку 80-х років на ринку персональних комп'ютерів. Мільярдні доходи тоді з'явилися сигналом для вступу на цей ринок гіганта комп'ютерної індустрії - IBM. Всі ми прекрасно пам'ятаємо як в результаті виниклої конкуренції за гроші кінцевого користувача перетворився весь комп'ютерний світ. Нейрокомп'ютери і їх програмні емулятори, природно, цікаві не самі по собі, а як інструмент вирішення практичних завдань. Лише в цьому випадку нейропродукція володітиме споживчою вартістю і мати відповідний об'єм ринку.

Є багато вражаючих демонстрацій можливостей штучних нейронних мереж. У деяких областях, таких як виявлення фальсифікацій і оцінка ризику, вони стали безперечними лідерами серед використовуваних методів. Їх застосування в системах прогнозування і системах маркетингових досліджень постійно росте. Розглянемо деякі найбільш характерні приклади практичного використання нейронних мереж [41, 53, 70, 77].

Контроль операцій з кредитними картками. Здібності нейромереж до класифікації застосовуються для відстежування операцій з краденими кредитними картами і підробленими чеками. Спеціалізована система Falcon фірми HNC дозволяє по частоті операцій і характеру покупок виділити підозрілі операції і сигналізувати про це. Ця система використовує запатентовану уніфіковану технологію підтримки ухвалення рішення (Unified Decisions Technology), яка комбінує розширену базу даних правил обробки транзакцій, статистичний аналіз і нейронну мережу. Система Falcon також містить спеціальний компонент, що забезпечує можливість досвідченним фахівцям включати в базу даних системи правил, що дозволяють з високою мірою достовірності визначати потенційні випадки шахрайства з банківськими картами по географічному місцю розташування або поштовому індексу отримувача карти. Практичне використання цієї системи показало підвищення показників якості виявлення на 20–60% при значному зниженні помилкових

спрацьовувань. Завдяки цій системі втрати банків від таких операцій помітно зменшилися. В даний час Falcon контролює більше 260 мільйонів рахунків 16 найбільших емітентів кредитних карт.

Аналогічна система, яка розроблена фірмою ITC, використовується для обробки операцій з кредитними картами Visa. У 2005 році за допомогою цієї системи запобігли нелегальні операції на суму більше 100 млн. доларів.

Сімейство систем PRISM, розроблене компанією Nestor, засноване на використанні нейромереж, експертних систем і статистичних методів для виявлення в реальному режимі часу шахрайства з кредитними і дебетовими картами, а також для виявлення інших типів шахрайства при здійсненні фінансових або торгівельних операцій. Нейромережа, використовувана в системах сімейства PRISM, була навчена на основі більш ніж напівмільйона транзакцій з різними типами карт. По деяких оцінках, використання систем сімейства PRISM дозволяє зменшити число шахрайств на 50 %.

Нейромережі на фінансових ринках. Американський Citibank використовує нейромережеві передбачення з 1990 року. У 2002 році, за свідченням журналу The Economist, автоматичний ділінг показував прибутковість 25% річних, що набагато перевищує показники більшості брокерів. Chemical Bank використовує нейро-систему фірми Neural Data для попередньої обробки транзакцій на валютних біржах 23 країн, фільтруючи «підозрілі» операції. Fidelity of Boston використовує нейромережі при управлінні портфелями з сумарним об'ємом 3 мільярда доларів. Повністю автоматизовані системи ведення портфелів з використанням нейромереж застосовують, наприклад, Deere & Co - на суму $100 млн і LBS Capital - на суму $400 млн. У останньому випадку експертна система об'єднується з приблизно 900 нейромережами.

Компанія Alela Corp. займається прогнозуванням зміни біржових індексів. Для передбачення знаків зміни індексів застосовується нейронна мережа, використовуюча РБФ (радіальні базисні функції). На сайті компанії можна безкоштовно отримати прогнози зміни індексів Dow Jones, S&P500 і

Merval, а також переконатися, що доля вірних передбачень складає не менше 80%.

Аналіз страхових позовів. ***Фірмою Neural Innovation Ltd. створена нейромережева система Claim Fraud Analyser, що дозволяє миттєво виявляти підозрілі страхові позови, що відносяться до пошкоджених автомобілів. На вході системи подаються такі параметри, як вік і досвід водія, вартість автомобіля, наявність подібних випадків у минулому та інші. В результаті обробки видається число - вірогідність того, що даний позов пов'язаний з шахрайством. Така система дозволяє не лише заощадити за рахунок виявлення фальсифікацій, але і поліпшити стосунки з клієнтами за рахунок швидшого задоволення чесних позовів***.

Аналіз споживчого ринку. Кілька років тому фірма IBM Consulting виконала замовлення на створення нейромережевої системи, що прогнозує властивості споживчого ринку. Замовник - один з найбільших виробників харчових продуктів, що має величезні ринки збуту. Одним з основних маркетингових механізмів замовника є поширення купонів, що дають право покупки певного товару із знижкою. Так як витрати на розсилку купонів досить великі, вирішальним чинником є ефективність розсилки, тобто доля клієнтів, що скористалися знижкою. Для підвищення ефективності купонної системи важливо було провести попередню сегментацію ринку, а потім адресувати клієнтам кожного сегменту саме ті купони, якими вони з більшою вірогідністю скористаються. В термінах інтелектуального аналізу даних потрібно було вирішити задачу кластеризації, що і було успішно зроблено за допомогою мереж Кохонена. На другому етапі для споживачів кожного з кластерів підбиралися відповідні комерційні пропозиції, а потім будувався прогноз об'єму продажів для кожного сегменту.

Інший варіант рішення цієї ж задачі вибрала компанія GoalAssist Corporation, виконуючи замовлення крупної маркетингової фірми. Потрібно було дослідити стратегію заохочувальних товарів (коли, наприклад, надіславши 5 етикеток від чіпсів, клієнт отримує безкоштовно футболку) для певної

компанії, яка торгувала харчовими продуктами. Звичайні методи прогнозування відгуку споживачів виявилися в даному випадку недостатньо точні. В результаті попит на футболки виявився дуже великий і багатьом покупцям довелося довго чекати здобуття призу, тоді як інші дарунки залишилися незатребуваними. Аби підвищити точність прогнозування, було вирішено використовувати історичні дані і нейронні мережі. Компания GoalAssist Corporation побудувала дві нейромережі для вирішення цього завдання. Перша з них - це мережа з адаптивною архітектурою пакету NeuroShell Classifier компанії Ward Systems Group, на входи якої подавалися різні параметри товарів і рекламної політики. За допомогою цієї мережі, призначеної спеціально для класифікації, було отримано розділення входів на 4 класи, що характеризують відгук споживачів. Ті ж входи разом з відповіддю першої мережі подавалися далі на вхід пакету NeuroShell Predictor, який також містить складну мережу пристосовану для задач кількісного прогнозування. Середня помилка передбачень склала всього біля 4%. Побудова цієї моделі зайняла близько 120 годин, також був потрібен час на передобробку вхідних даних. Експерти GoalAssist Corp. вважають, що ця модель і далі успішно застосовуватиметься для вирішення маркетингових задач.

Компанія Neural Innovation Ltd. використовує при роботі з маркетинговими компаніями конкретну стратегію прямої розсилки. Спочатку розсилається 25% від загального числа пропозицій і збирається інформація про відгуки споживачів. Потім ця інформація поступає на вхід нейрокомп'ютера, який здійснює пошук оптимального сегменту споживчого ринку для даного товару. Останні 75% пропозицій розсилаються у вказаний сегмент. При цьому ефективність розсилки істотно зростає.

Dr. Al Behrens, співробітник Northern Natural Gas в штаті Небраска, навчив нейронну мережу, яка передбачає зміну цін на газ в наступному місяці з середньою точністю 97 %. Щомісячна ціна інколи прив'язується до індексації цін в друкарських виданнях (Inside FERC і Natural Gas Week), включаючи недавню ринкову діяльність компанії, сезонні чинники, погоду і так далі.

Відома корпорація Microsoft також використовує програмні продукти на основі нейромереж в своїй маркетинговій політиці. Щороку Microsoft надсилає більше 40 млн. рекламних пропозицій про покупку своїх продуктів більш ніж 8,5 млн. зареєстрованим користувачам. Велика частина направлена на модернізацію ПО або покупку додаткових пакетів. Мета використання ПО на основі нейроних мереж очевидна - збільшення обсягу продажу. Такими засобами було досягнуте збільшення попиту з 4,9 % до 8,2 %, а відмінність у витратах на рекламу знизилася на 35 %.

Виробництво. Нейронна мережа, застосована на заводі Intel, здатна ідентифікувати брак на виробництві. Спочатку системі давали електричну випробувальну інформацію від готових чіпів і відповідних змінних управління виробничим процесом. Стосунки між цими двома параметрами були визначені числовим експериментом і моделюванням процесу CMOS. Чутлива поверхнева модель використовувалася для збереження результатів достатньої кількості числових експериментів. В результаті нейронна мережа була здатна забракувати нефункціонуючий чіп з точністю 99,5%.

При проведенні випробувань якості бетону використовується велика кількість методів. Одним з них є буріння у пошуках порожнини, що утворилася. Проте за допомогою нейромереж можливо перевірити весь матеріал, а також визначити глибину, на якій знаходиться порожнина. Шляхом подачі звукових хвиль і прийому відбитого сигналу, а потім обробкою нейронною мережею, фахівці з National Institute of Standards and Technology (NIST) здатні перевірити якість бетону при товщині матеріалу до півметра.

Ford Motors Company впровадила у себе нейросистему для діагностики двигунів після невдалих спроб побудувати експертну систему, оскільки хоча досвідчений механік і може діагностувати несправності він не в змозі описати алгоритм такого розпізнавання. На вхід нейро-системи подаються дані від 31 датчика. Нейромережа навчалася різним видам несправностей по 868 прикладам. Після повного циклу навчання якість діагностування несправностей

мережею досягла рівня кращих експертів, і значно перевершувала їх в швидкості.

Медична діагностика. Компанією «НейроПроект» створена система об'єктивної діагностики слуху у грудних дітей. Загальноприйнята методика об'єктивної діагностики полягає в тому, що в процесі обстеження реєструються «викликані потенціали» (відгуки мозку) у відповідь на звуковий подразник, що виявляються у вигляді сплесків на електроенцефалограмі. Для досить упевненої діагностики слуху дитини досвідченному експерту-аудіологу необхідно провести близько 2000 тестів, що займає близько години. Нейромережа здатна з тією ж достовірністю визначити рівень слуху вже за 200 спостереженнями протягом всього декількох хвилин, причому без участі кваліфікованого персоналу.

Ще одним прикладом використання нейронних мереж в програмах медичної діагностики служить пакет кардиодіагностики, розроблений фірмою RES Informatica спільно з Центром кардіологічних досліджень в Мілані. Причому для таких хвороб, як ішемія міокарду і артеріальна гіпертензія, досягається точність постановки діагнозу більш ніж 95 %. Окрім цього, одним з перспективних напрямів в дослідженнях є онлайн-постановка діагнозу. Пацієнт заповнює форму, вказуючи необхідні параметри, а система на основі навчальної вибірки і накопиченої бази даних визначає діагноз. За деякими даними, це дозволить збільшити продуктивність більш ніж в 1000 разів, здешевити дослідження не менше чим в 500 разів, обробити дані від практично необмеженої кількості хворих, при збільшенні діагностичної точності.

Системи безпеки в аеропортах. Американська фірма SAIC (Science Application International Corporation) використовувала нейронні мережі в своєму проекті TNA. TNA є ящиком вартістю $750.000, який здатний виявляти пластикову вибухівку в запакованому багажі. TNA бомбардує багаж повільними нейтронами, що викликають вторинне гамма-випромінювання, спектр якого аналізується нейронною мережею. Система виявляє вибухівку з вірогідністю вище 97% і переглядає 10 місць багажу в хвилину.

Активна реклама в Internet. Нейромережевий продукт SelectCast фірми Aptex Software Inc. виявляє профілі інтересів користувачів Internet і пропонує їм відповідним чином відфільтровану рекламу. Після установки на серверах Excite і Infoseek, нейромережева реклама охопила біля третини всіх користувачів Internet. Згідно проведеним дослідженням, встановлено, що відгук на таку активну рекламу в середньому удвічі вище, ніж на звичайну рекламу, що розміщується в Мережі. А на окремі види реклами відгук зріс вп'ятеро. Відмітимо, що рекламний сектор Internet переживає зараз період бурхливого розвитку. Результати першого півріччя 2007 років свідчать про річний темп зростання 250%, що в грошовому вираженні складає $400 млн.

Політичні технології. Компанія «НейроПроект» досить упевнено передбачає результати президентських виборів в США на підставі анкети з 12 питань. Причому, аналіз навченої нейромережі дозволив виявити п'ять ключових питань, відповіді на яких формують два головні чинники, що визначають успіх президентської кампанії.

Наукові дослідження. Доктор Henrik Lundstedt з Lund Observatory, Швеція, навчив нейронні мережі прогнозувати ефекти від сонячних спалахів, такі як обурення магнітних полів Землі. Як відомо, т.з. магнітні бурі роблять вплив не лише на стан і здоров'я людини, під їх вплив потрапляють і електростанції, трапляються збої в передачі радио- і телепередач, збої в роботі геологічного устаткування, супутників і так далі. Нейронна мережа, що враховує 37 відомих впливаючих чинників раз в чотири дні і аналізує їх зміни, здатна з високою мірою точності визначити «космічну погоду». Відомий випадок, коли стандартний метод прогнозування не визначив сильних магнітних бурь взагалі, а метод на основі нейронних мереж в той же час точно спрогнозував дві з трьох бурь.

Автопілотований гіперзвуковий літак-розвідник. Названий LoFLYTE (Low-Observable Flight Test Experiment) реактивний безпілотний літак завдовжки 2,5 м був розроблений для NASA і Air Force фірмою Accurate Automation Corp. в рамках програми підтримки малого інноваційного бізнесу.

Це експериментальна розробка для дослідження нових принципів пілотування, включаючи нейронні мережі, що дозволяють автопілоту навчатися, копіюючи прийоми пілотування льотчика. З часом нейромережі переймають досвід управління, а швидкість обробки інформації дозволить швидко знаходити вихід в екстремальних і аварійних ситуаціях. LoFLYTE призначений для польотів з дуже високою швидкістю, коли швидкості реакції пілота може не вистачити для адекватного реагування на зміни режиму польоту.

3.2. Архітектура нейронних мереж.

Ідеї нейрокомп'ютинга з'явилися практично одночасно із зародженням послідовних ЕОМ. Основні роботи по нейро-обчисленнях з'явилася на два роки раніше знаменитої доповідної записки фон Неймана про принципи організації обчислень в послідовних універсальних ЕОМ. Вияв цікавості до штучних нейронних мереж був обумовлений роботами піонерів у цій справі - У. Маккалоха і У. Піттса. У 1943 році увагу громадськості привернула робота під назвою «Логічне числення ідей, що відносяться до нервової діяльності», в якій вони запропонували математичну модель нейрона і сформулювали принципи побудови штучних нейронних мереж, згідно розробленої ними моделі функціонування головного мозку. Багато учених з ентузіазмом почали пропонувати свої рішення і нову архітектуру нейронних мереж [32, 33, 52].

Паралельно з прогресом в нейроанатомії і нейрофізіології психологами були створені моделі людського навчання. Одній з таких моделей, що виявилася найбільш плідною, була модель Д. Хебба, який в 1949 році запропонував закон навчання, що виявився стартовою точкою для алгоритмів навчання штучних нейронних мереж. Доповнений сьогодні безліччю інших методів він продемонстрував вченим того часу, як мережа нейронів може навчатися.

Перший експериментальний нейрокомп'ютер Snark був побудований Марвіном Мінським в 1951 році. Проте, він не був пристосований до вирішення практично цікавих задач, і перший успіх нейрокомп’ютинга пов'язують з розробкою іншого американця - Френка Розенблатта - персептроном (від англійського perception - сприйняття). Персептрон був вперше змодельований на універсальній ЕОМ IBM-704 в 1958 році, причому його навчання вимагало біля півгодини машинного часу. Апаратний варіант - Mark I Perceptron, був побудований в 1960 році і призначався для розпізнавання зорових образів. Його рецепторне поле складалося з 400 пікселів (матриця фотоприймачів 20x20), і він успішно справлявся з вирішенням ряду задач, зокрема міг розрізняти деякі букви (рис. 3.10).

Рис. 3.10. Френк Розенблатт з Mark I Perceptron.

Мінський, Розенблатт, Уїдроу та інші розробили мережі, що складаються з одного шару штучних нейронів. Часто звані персептронами, вони були використані для такого широкого класу задач, як передбачення погоди, аналіз електрокардіограм і штучний зір. Протягом деякого часу здавалося, що ключ до інтелекту знайдений і відтворення людського мозку є лише питанням конструювання досить великої мережі. Але ця ілюзія скоро розсіялася. Мережі не могли вирішувати завдання, зовні вельми схожі з тими, які вони успішно вирішували. З цих нез'ясовних невдач почався період інтенсивного аналізу. Мінський, використовуючи точні математичні методи, строго довів ряд теорем,

що відносяться до функціонування мереж. Його дослідження привели до написання книги, в якій він разом з Пайпертом довів, що використовувані у той час одношарові мережі теоретично нездібні вирішити багато простих завдань. Мінський також не був оптимістичний відносно потенційно можливого прогресу.

Їх робота остудила науковий запал багатьох учених-ентузіастів практично на два десятиліття. Утворилося деяке затишшя, пов'язане з розчаруванням в можливостях нейронних мереж, викликаним не в останню чергу психологічним чинником, що виявляється в нездатності людини описати словами те, як він думає. І хоча в цей період було запропоновано багато заслуговуючих уваги розробок і досліджень, науковий світ відносився до них скептично. Дослідження в цьому напрямі були згорнуті аж до 1983 року, коли вони, нарешті, отримали фінансування від Агентства Перспективних Військових Досліджень США, DARPA. Цей факт став сигналом на початок нового нейромережевого буму.

Інтерес широкої наукової громадськості до нейромереж прокинувся після теоретичної роботи фізика Джона Хопфілда (1982 р.), що запропонував модель асоціативної пам'яті в нейронних ансамблях. Холфілд і його багаточисельні послідовники збагатили теорію нейромереж багатьма ідеями з арсеналу фізики, такими як колективні взаємодії нейронів, енергія мережі, температура навчання і так далі Проте справжній бум практичного вживання нейромереж почався після публікації в 1986 році Девідом Румельхартом із співавторами методу навчання багатошарового персептрона, названного ними методом зворотного поширення помилки (error back-propagation). В цей же час з'явилися перші комерційні проекти нейрокомп'ютерів (Mark III фірми TRW, США). Обмеження персептронів, про які писали Мінський і Пейперт, виявилися переборними, а можливості обчислювальної техніки - достатніми для вирішення широкого круга прикладних завдань. У 90-х роках продуктивність послідовних комп'ютерів зросла настільки, що це дозволило моделювати з їх допомогою роботу паралельних нейронних мереж з числом

нейронів від декількох сотень до десятків тисяч. Такі емулятори нейромереж здатні вирішувати багато важливих з практичної точки зору завдань. У свою чергу, нейромережеві програмні комплекси стануть тим носієм, який виведе на технологічну орбіту сьогодення паралельне нейромережеве hardware.

Подальший розвиток нейромережевих технологій підтримувався створенням спеціалізованих центрів нейрокомп'ютерів, проведенням великого числа міжнародних конференцій і форумів. Обсяг продажу комерційних нейромережевих продуктів в період з 1991 по 2007 р. виріс більш ніж в чотирнадцять разів. Зараз, з врахуванням переходу на нанотехнології, появою нових знань про діяльність людського мозку, з'явилися принципово нові архітектури, технологічні рішення, визначені напрями досліджень, з'явився широкий спектр завдань, які не під силу вирішувати алгоритмічно навіть на сучасних ПК.

Означення. Штучні нейронні мережі - вид математичних моделей, які будуються за принципом організації і функціонування їх біологічних аналогів, - мереж нервових клітин (нейронів) мозку. У основі їх побудови лежить ідея про те, що нейрони можна моделювати досить простими автоматами (штучними нейронами), а вся складність мозку, гнучкість його функціонування і інші найважливіші якості визначаються зв'язками між нейронами.

Відомі такі типи нейронних мереж:

- персептрон Розенблатта та багатошаровий персептрон;
- нейронні мережі Джордана, Елмана, Хеммінга;
- мережа Хопфілда та мережа Кохонена;
- когнітрон та неокогнітрон;
- хаотична нейронна мережа та осциляторна нейронна мережа;
- мережі зустрічного розповсюдження та мережі радіальних базисних функцій (RBF – мережі);
- мережі узагальненої регресії та імовірнісні мережі;
- сіамська нейронна мережа та мережа адаптивного резонансу.

Як вже було відмічено, світовий ринок пропонує більше кілька тисяч нейромережевих пакетів. Загальний об'єм ринку нейронних мереж до 2008 року перевищив $10 млрд. І, практично, кожний розробник традиційних аналітичних пакетів сьогодні прагне включити нейронні мережі в нові версії своїх програм. У США нейронні мережі застосовуються в аналітичних комплексах кожного крупного банку. Наприклад, продаж одного тільки нейромережевого пакету «Brain Maker Pro» порівнянна з об'ємами продажів найпопулярнішого пакету технічного аналізу MetaStock.

У зв’язку з великою різноманітністю штучних нейронних мереж и специфікою їх реалізації важливим є проведення класифікацію їх архітектур для подальшого застосування в інтелектуальних системах аналіза даних.

- Нейронні мережі можуть бути *синхронні* і *асинхронні*.

У синхронних нейронних мережах в кожен момент часу свій стан міняє лише один нейрон. У асинхронних - стан міняється відразу у цілої групи нейронів, як правило, у всього шару.

- По типу базової архітектури - *шаровані* і *повнозв'язні* мережі.

Ключовим в шаруватих мережах є поняття прошарка. Шар - один або декілька нейронів, на входи яких подається один і той же загальний сигнал. Шаровані нейронні мережі - нейронні мережі, в яких нейрони розбиті на окремі групи (шари) так, що обробка інформації здійснюється пошарово. У шарованих мережах нейрони і-го шару отримують вхідні сигнали, перетворять їх і через точки рощгалуження передають нейронам (і+1) шару. І так до к-го шару, який видає вихідні сигнали для інтерпретатора і користувача. Число нейронів в кожному шарі не пов'язане з кількістю нейронів в інших шарах і може бути довільним.

В рамках одного шару дані обробляються паралельно, а в масштабах всієї мережі обробка ведеться послідовно - від шару до шару. До шарованих нейронних мереж відносяться, наприклад, багатошарові персептрони, мережі радіальних базисних функцій, когнітрон, некогнітрон, мережі асоціативної пам'яті. Проте сигнал не завжди подається на всі нейрони шару. У когнітроні,

наприклад, кожен нейрон поточного шару отримує сигнали тільки від близьких йому нейронів попереднього шару.

Шаровані мережі, у свою чергу, можуть бути *одношаровими* і *багатошаровими*. Одношарова мережа – мережа складається з одного шару. Багатошарова мережа – мережа має декілька шарів. У багатошаровій мережі перший шар називається вхідним, подальші - внутрішніми або прихованими, останній шар - вихідним. Таким чином, проміжні шари - це всі шари в багатошаровій нейронній мережі, окрім вхідного і вихідного. Вхідний шар мережі реалізує зв'язок з вхідними даними, вихідний - з вихідними.

Таким чином, нейрони можуть бути вхідними, вихідними і прихованими. Вхідний шар організований з вхідних нейронів (input neuron), які отримують дані і поширюють їх на входи нейронів прихованого шару мережі. Прихований нейрон (hidden neuron) - це нейрон, що знаходиться в прихованому шарі нейронної мережі. Вихідні нейрони (output neuron), з яких організований вихідний шар мережі, видає результати роботи нейронної мережі.

У *повнозв'язних мережах* кожен нейрон передає свій вихідний сигнал решті нейронів, включаючи самого себе. Вихідними сигналами мережі можуть бути все або деякі вихідні сигнали нейронів після декількох тактів функціонування мережі. Всі вхідні сигнали подаються всім нейронам.

- По спрямованості зв'язків.

Нейронні мережі бувають із *зворотними зв'язками* і *без зворотних зв'язків*.

Серед мереж без зворотних зв'язків розрізняють *мережі із зворотним розповсюдженням помилки* та *інші мережі*. Мережі першої групи характеризуються фіксованою структурою, ітераційним навчанням, корегуванням вагів по помилках. Перевагами мереж без зворотних зв'язків є простота їх реалізації і гарантоване отримання відповіді після проходження даних по шарах. Недоліком цього виду мереж вважається мінімізація розмірів мережі - нейрони багато разів беруть участь в обробці даних. Менший об'єм мережі полегшує процес навчання.

До мереж із зворотними зв'язками відносять мережі Хопфілда та мережі Кохонена. Перевагами мереж із зворотними зв'язками є складність навчання, викликана великим числом нейронів для алгоритмів одного і того ж рівня складності. Недоліки цього виду мереж - потрібні спеціальні умови, що гарантують сходимість обчислень.

- Мережі прямого розповсюдження і рекурентні мережі.

До мереж *прямого розповсюдження* зазвичай відносять персептрони, мережу Back Propagation, мережу зустрічного розповсюдження та карти Кохонена. Всі зв'язки направлені строго від вхідних нейронів до вихідних.

Характерна особливість *рекурентних мереж* - наявність блоків динамічної затримки і зворотних зв'язків, що дозволяє їм обробляти динамічні моделі, наприклад, мережа Хопфілда або мережа Елмана - мережа, що складається з двох шарів, в якій прихований шар охоплений динамічним зворотним зв'язком, що дозволяє врахувати передісторію спостережуваних процесів і накопичити інформацію для вироблення правильної стратегії управління. Ці мережі застосовуються в системах управління рухомими об'єктами. Окремим випадком рекурентних мереж є двонаправлені мережі. У таких мережах між шарами існують зв'язки як в напрямі від вхідного шару до вихідному, так і в зворотному. Класичним прикладом є нейронна мережа Косько.

- Навчання з вчителем або без нього.

Перед використанням нейронної мережі її необхідно навчити. Процес навчання нейронній мережі полягає в підстроюванні її внутрішніх параметрів під конкретне завдання. Алгоритм роботи нейронної мережі є ітеративним, його кроки називають епохами або циклами.

Епоха - одна ітерація в процесі навчання, що включає пред'явлення всіх прикладів з навчальної множини і, можливо, перевірку якості навчання на контрольній множині. Процес навчання здійснюється на навчальній вибірці. Вона включає вхідні значення і відповідні їм вихідні значення набору даних. В ході навчання нейронна мережа знаходить деякі залежності вихідних полів від

вхідних. Таким чином, перед нами ставиться питання - які вхідні поля (ознаки) нам необхідно використовувати. Спочатку вибір здійснюється евристично, далі кількість входів може бути змінена. Складність може викликати питання про кількість спостережень в наборі даних. І хоча існують деякі правила, що описують зв'язок між необхідною кількістю спостережень і розміром мережі, їх вірність не доведена. Кількість необхідних спостережень залежить від складності вирішуваного завдання. При збільшенні кількості ознак кількість спостережень зростає нелінійно, ця проблема носить назву "Прокліn розмірності". При недостатній кількості даних рекомендується використовувати лінійну модель.

Аналітик повинен визначити кількість шарів в мережі і кількість нейронів в кожному шарі. Далі необхідно призначити такі значення вагів і зміщень, які зможуть мінімізувати помилку рішення. Ваги і зміщення автоматично підстраюються так, щоб мінімізувати різницю між бажаними і отриманими на виході сигналами, яка називається помилка навчання.

Помилка навчання для побудованої нейронної мережі обчислюється шляхом порівняння вихідних і цільових (бажаних) значень. З отриманих різниць формується функція помилок. Функція помилок - це цільова функція, що вимагає мінімізації в процесі керованого навчання нейронної мережі. За допомогою функції помилок можна оцінити якість роботи нейронної мережі під час навчання. Наприклад, часто використовується сума квадратів помилок. Від якості навчання нейронній мережі залежить її здатність вирішувати поставлені перед нею завдання.

Навчання з вчителем передбачає, шо для кожного навчального вхідного прикладу потрібне знання правильної відповіді або функції оцінки якості відповіді. Таке навчання називають *керованим*. Нейронній мережі пред'являються значення вхідних і вихідних сигналів, а вона по певному алгоритму підстроює ваги синаптіческіх зв'язків. В процесі навчання проводиться корегування вагів мережі за наслідками порівняння фактичних вихідних значень з вхідними, відомими наперед.

При *навчанні без вчителя* розкривається внутрішня структура даних або кореляції між зразками в наборі даних. Виходи нейронної мережі формуються самостійно, а ваги змінюються по алгоритму, що враховує тільки вхідні і похідні від них сигнали. Це навчання називають також *некерованим*. В результаті такого навчання об'єкти або приклади розподіляються по категоріях, самі категорії і їх кількість можуть бути наперед не відомі.

При навчанні нейронних мереж часто виникають серйозні труднощі, названі *проблемою перенавчання* (overfitting). Перенавчання, або надмірно близька підгонка - надмірно точна відповідність нейронної мережі конкретному набору навчальних прикладів, при якому мережа втрачає здібність до узагальнення.

Перенавчання виникає в разі дуже довгого навчання, недостатнього числа навчальних прикладів або переускладненої структури нейронної мережі. Перенавчання пов'язане з тим, що вибір навчальної (тренувальної) множини є випадковим. З перших кроків навчання відбувається зменшення помилки. На подальших кроках з метою зменшення помилки (цільовій функції) параметри підстроюються під особливості навчальної множини. Проте при цьому відбувається «підстроювання» не під загальні закономірності ряду, а під особливості його частини - навчальної підмножини. При цьому точність прогнозу зменшується. Один з варіантів боротьби з перенавчанням мережі - ділення навчальної вибірки на дві множини (навчальної і тестової). На навчальній множині відбувається навчання нейронній мережі. На тестовій множині здійснюється перевірка побудованої моделі. Ці множини не повинні перетинатися. З кожним кроком параметри моделі змінюються, проте постійне зменшення значення цільової функції відбувається саме на навчальній множині. При розбитті множини на дві ми можемо спостерігати зміну помилки прогнозу на тестовій множині паралельно із спостереженнями над навчальною множиною. Якась кількість кроків помилки прогнозу зменшується на обох множинах. Проте на певному кроці помилка на тестовій множині починає зростати, при цьому помилка на навчальній множині продовжує зменшуватися.

Цей момент вважається кінцем реального або справжнього навчання, з нього і починається перенавчання. Прогноз на тестовій множині є перевіркою працездатності побудованої моделі. Помилка на тестовій множині може бути помилкою прогнозу, якщо тестова множина максимально наближена до теперішнього моменту.

- Одношарові і багатошарові штучні нейронні мережі.

Хоча один нейрон і здатний виконувати прості процедури аналізу, але для серйозних нейронних обчислень необхідно сполучати нейрони в мережі. Проста мережа складається з групи нейронів, створюючих шар (рис. 3.11). Відзначимо, що вершини-круги зліва служать лише для розподілу вхідних сигналів. Вони не виконують яких-небудь обчислень і тому не вважатимуться шаром. Для більшої наочності позначимо їх кругами, щоб відрізняти їх від обчислюючих нейронів, позначених квадратами. Кожен елемент з множини входів X окремою вагою поєднан з кожним штучним нейроном. А кожен нейрон видає зважену суму входів в мережу. У штучних і біологічних мережах багато з'єднань можуть бути відсутніми, але тут вони показані всі для демонстрації загальної картини. Можуть існувати також з'єднання між виходами і входами елементів в шарі.

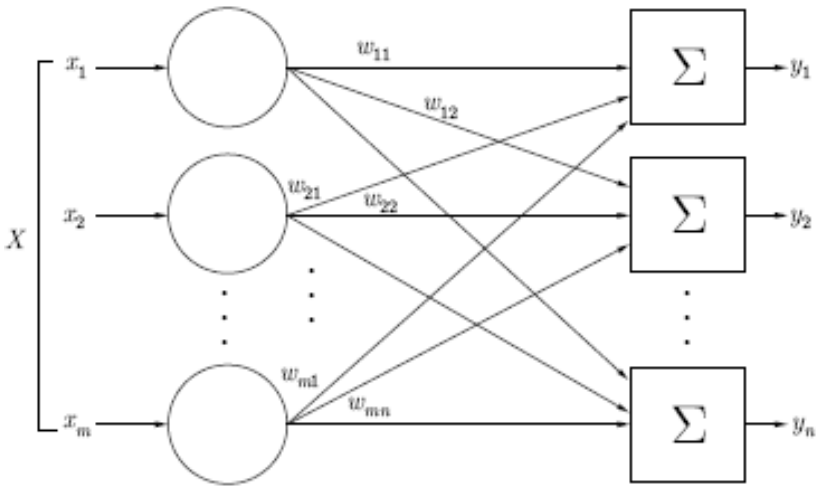


Рис. 3.11. Одношарова нейронна мережа.

Зручно вважати ваги елементами матриці W. Матриця має m рядків і n стовпців, де - m число входів, а - n число нейронів. Наприклад, $w_{2,3}$ - це вага,

що пов'язує другий вхід з третім нейроном. Таким чином, обчислення вихідного вектора N, компонентами якого є виходи OUT нейронів, зводиться до матричного множення $N = XW$, де N і X - вектори-рядки.

Більші і складніші багатошарові нейронні мережі володіють, як правило, і великими обчислювальними можливостями. Хоча створені мережі всіх конфігурацій, які тільки можна собі представити, пошарова організація нейронів копіює шаруваті структури певних відділів мозку. Виявилось, що такі багатошарові мережі володіють більшими можливостями ніж одношарові, і останніми роками були розроблені алгоритми для їх навчання. Багатошарові мережі можуть будуватися з каскадів шарів. Вихід одного шару є входом для подальшого шару (рис. 3.12).

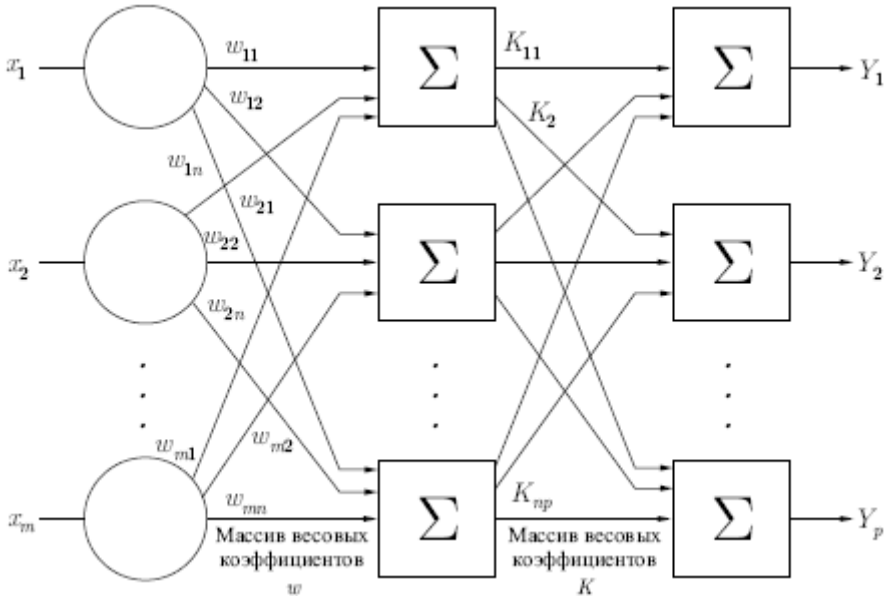


Рис. 3.12. Багатошарова нейронна мережа.

Багатошарові мережі не можуть привести до збільшення обчислювальної потужності в порівнянні з одношаровою мережею, якщо активаційна функція між шарами лінійна. Обчислення виходу шару полягає в множенні вхідного вектора на першу вагову матрицю з подальшим множенням (якщо відсутня нелінійна активаційна функція) результуючого вектора на другу вагову матрицю $OUT = (XW_1)W_2$. Оскільки множення матриць асоціативне, то

$(XW_1)W_2 = X(W_1W_2)$. Це показує, що двошарова лінійна мережа еквівалентна одному шару з ваговою матрицею, рівною множинню двох вагових матриць. Отже, будь-яка багатошарова лінійна мережа може бути замінена еквівалентною одношаровою мережею. Проте одношарові мережі вельми обмежені по своїх обчислювальних можливостях. Таким чином, для розширення можливостей мереж в порівнянні з одношаровою мережею необхідна нелінійна активаційна функція.

На жаль, немає загальноприйнятого способу підрахунку числа шарів в мережі. Багатошарова мережа складається, як показано на малюнку 3.12, з множини нейронів і вагів, що чередуються. Перший шар не приймається до уваги при підрахунку шарів, і мережа, подібна зображеною на малюнку 3.12, вважається двошаровою, оскільки тільки два шару виконують обчислення. Далі, ваги шару вважаються пов'язаними з наступними за ними нейронами. Отже, шар складається з множини вагів з наступними за ними нейронами, що підсумовують зважені сигнали.

Розглянемо найбільш поширені архітектури штучних нейронних мереж.

Персептрон. Персептрон (perceptron) - математична і комп'ютерна модель сприйняття інформації мозком (кібернетична модель мозку), запропонована Френком Розенблаттом і реалізована у вигляді електронної машини «Марк-1» в 1960 році. Персептрон став однією з перших моделей нейромереж, а «Марк-1» - першим в світі нейрокомп’ютером. Не дивлячись на свою простоту, персептрон здатний навчатися і вирішувати досить складні задачі (рис. 3.13).

Персептрон складається з трьох типів елементів, а саме: сигнали, що поступають від сенсорів (S – елементи), передаються асоціативним елементам (A – елементи), а потім реагуючим елементам (R – елементи). Кожен сенсор може знаходитися в одному з двох станів - *спокою* або *збудження*, і лише в останньому випадку він передає одиничний сигнал в наступний шар, асоціативним елементам. А-елементи називаються асоціативними, тому що кожному такому елементу, як правило, відповідає цілий набір (асоціація) S-

елементів. А-елемент активізується, як тільки кількість сигналів від S-елементів на його вході перевищила деяку величину θ.

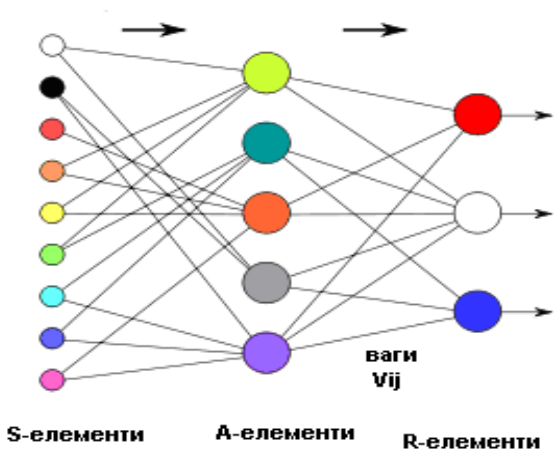


Рис. 3.13. Логічна схема персептрона.

Сигнали від А-елементів, що збудилися, у свою чергу, передаються в суматор R, причому сигнал від і-го асоціативного елементу передається з коефіцієнтом w_i. Цей коефіцієнт називається *вагою* A-R зв'язку. Так само як і А-елементи, R-елемент підраховує суму значень вхідних сигналів, помножених на ваги. R-елемент, а разом з ним і елементарний персептрон, видає «1», якщо лінійна форма перевищує поріг і, інакше на виході буде «-1». Математично, функцію, що реалізовується R-елементом, можна записати так:

$$f(x) = sign(\sum_{i=1}^{n} w_i x_i - \theta)$$

Таким чином, персептрони дозволяють створити набір «асоціацій» між вхідними стимулами і необхідною реакцією на виході. У біологічному плані це відповідає перетворенню, наприклад, зорової інформації у фізіологічну відповідь від рухових нейронів. Згідно сучасної термінології, персептрони можуть бути класифіковані як штучні нейронні мережі:

1. з одним прихованим шаром;
2. з пороговою передавальною функцією;
3. з прямим розповсюдженням сигналу.

Навчання елементарного персептрона полягає в зміні вагових коефіцієнтів w_i зв'язків A-R. Ваги зв'язків S-A (які можуть приймати значення

{-1; 0; +1}) і значення порогів А-елементів вибираються випадковим чином на самому початку і потім не змінюються.

Після навчання персептрон готовий працювати в режимі *розпізнавання* або *узагальнення*. У цьому режимі персептрону пред'являються не знайомі об'єкти, і персептрон повинен встановити, до якого класу вони належать. Робота персептрона полягає в наступному: при пред'явленні об'єкту А-елементи, що збудилися, передають сигнал R-елементу, рівний сумі відповідних коефіцієнтів w_i. Якщо ця сума позитивна, то ухвалюється рішення, що даний об'єкт належить до першого класу, а якщо вона негативна - то другому.

Важливою властивістю будь-якої нейронної мережі є здібність до навчання. Розенблатт намагався класифікувати різні алгоритми навчання персептрона, називаючи їх системами підкріплення. При цьому слід розрізняти поняття представляємість і навчання. Поняття представляємості відноситься до здатності персептрона (або іншій мережі) моделювати певну функцію. Навчання ж вимагає наявність систематичної процедури настройки вагів мережі для реалізації цієї функції.

Навчання з вчителем. Класичний метод навчання персептрона - це *метод корекції помилки*. Він є таким видом навчання з вчителем, при якому вага зв'язку не змінюється до тих пір, поки поточна реакція персептрона залишається правильною. При появі неправильної реакції вага змінюється на одиницю, а знак (+/-) визначається протилежним від знаку помилки.

Допустимо, ми хочемо навчити персептрон розділяти два класи об'єктів так, щоб при пред'явленні об'єктів першого класу вихід персептрона був позитивний (+1), а при пред'явленні об'єктів другого класу - негативним (-1). Для цього виконаємо наступний алгоритм.

1. Випадковим чином вибираємо пороги для А-елементів і встановлюємо зв'язки S-A (далі вони змінюватися не будуть).
2. Початкові коефіцієнти w_i вважаємо рівними нулю.
3. Пред’являємо навчальну вибірку з вкзівкою класу, до якого вона належить.

- Показуємо перцептрону об'єкт першого класу. При цьому деякі А-елементи збудяться. Коефіцієнти w_i, відповідні цим збудженим елементам, збільшуємо на 1.
- Пред'являємо об'єкт другого класу і коефіцієнти w_i тих А-елементів, які збудяться при цьому показі, зменшуємо на 1.

4. Обидві частини кроку 3 здійснимо для всієї навчальної вибірки. В результаті навчання сформуються значення вагів зв'язків w_i.

Теорема збіжності персептрона показує, що елементарний персептрон, який навчається по такому алгоритму, незалежно від початкового стану вагових коефіцієнтів і послідовності появи стимулів завжди приведе до досягнення рішення за кінцевий проміжок часу.

Навчання без вчителя. Окрім класичного методу навчання персептрона Розенблатт також ввів поняття про навчання без вчителя, запропонувавши спосіб навчання, названий альфа-система підкріплення Ця система підкріплення, при якій ваги всіх активних зв'язків c_{ij}, які ведуть до елементу u_j, змінюються на однакову величину r, а ваги неактивних зв'язків за цей час не змінюються. Потім, з розробкою поняття багатошарового персептрона, альфа-система була модифікована, і її сталі називати *дельта-правило*. Модифікація була проведена з метою зробити функцію навчання такою, що диференціюється, що у свою чергу, потрібно для застосування *методу градієнтного спуску*, завдяки якому можливе навчання більш одного шару.

Метод зворотного розповсюдження помилки. Для навчання багатошарових мереж був запропонований градієнтний алгоритм навчання з вчителем, який проводить сигнал помилки, обчислений виходами персептрона, до його входів, шар за шаром. Зараз це найпопулярніший метод навчання багатошарових персептронов. Його перевага в тому, що він може навчити всі шари нейронної мережі, і його легко прорахувати локально. Проте цей метод є дуже довготривалим, до того ж, для його застосування потрібно, щоб передавальна функція нейронів була такою, що диференціюється. При цьому в

персептронах довелося відмовитися від бінарного сигналу, і користуватися на вході безперервними значеннями.

Алгоритм навчання мережі на основі метода зворотного розповсюдження помилки вимагає виконання наступних операцій:

1. Вибрати чергову навчальну пару з навчальної множини та подати вхідний вектор на вхід мережі.
2. Обчислити вихід мережі.
3. Обчислити різницю між виходом мережі і необхідним виходом (цільовим вектором навчальної пари).
4. Підкоригувати ваги мережі так, щоб мінімізувати помилку.

Повторювати кроки з 1 по 4 для кожного вектора навчальної множини до тих пір, поки помилка на всій множині не досягне прийнятного рівня.

Розенблатт виділив два фундаментальні обмеження для тришарових персептронов: відсутність у них здібності до узагальнення своїх характеристик на нові стимули або нові ситуації, а також нездатність аналізувати складні ситуації в зовнішньому середовищі шляхом розчленовування їх на простіші.

В той же час персептрони демонструють великі можливості для свого застосування, зокрема, в задачах класифікації, кластеризації, прогнозування та апроксимації, стиснення даних та асоціативної пам'яті.

Задачі класифікації. Як образи можуть виступати різні за своєю природою об'єкти: економічні показники, зображення, зразки звуків і т.д. При навчанні мережі пропонуються різні зразки образів з вказівкою того, до якого класу вони відносяться. Зразок, як правило, представляється як вектор значень ознак. При цьому сукупність всіх ознак повинна однозначно визначати клас, до якого відноситься зразок. У випадку, якщо ознак недостатньо, мережа може співвіднести один і той же зразок з декількома класами, що невірно. Після закінчення навчання мережі їй можна пред'являти невідомі раніше образи і отримувати відповідь про приналежність до певного класу.

Топологія такої мережі характеризується тим, що кількість нейронів у вихідному шарі, як правило, рівна кількості визначуваних класів. При цьому

встановлюється відповідність між виходом нейронної мережі і класом, який він представляє. Коли мережі пред'являється якийсь образ, на одному з її виходів повинна з'явитися ознака того, що образ належить цьому класу, в той же час на інших виходах повинна бути ознака того, що образ даному класу не належить. Якщо на двох або більш виходах є ознака приналежності до класу, вважається що мережа «не упевнена» в своїй відповіді.

Задачі кластеризації. Під кластеризацією розуміється розбиття множини вхідних сигналів на класи, при тому, що ні кількість, ні ознаки класів наперед невідомі. Після навчання така мережа здатна визначати, до якого класу відноситься вхідний сигнал. Мережа також може сигналізувати про те, що вхідний сигнал не відноситься ні до одного з виділених класів - це є ознакою нових, відсутніх в навчальній вибірці, даних. Таким чином, подібна мережа може виявляти нові, невідомі раніше класи сигналів. Відповідність між класами, виділеними мережею, і класами, що існують в преметній області, встановлюється людиною.

Прогнозування і апроксимація. Здібності нейронної мережі до прогнозування безпосередньо виходять з її здібності до узагальнення і виділення прихованих залежностей між вхідними і вихідними даними. Після навчання мережа здатна передбачити майбутнє значення якоїсь послідовності на основі декількох попередніх значень або деяких існуючих зараз чинників. Слід зазначити, що прогнозування можливе тільки тоді, коли попередні зміни дійсно якоюсь мірою зумовлюють майбутні. Наприклад, прогнозування котирувань акцій на основі котирувань за минулий тиждень може виявитися успішним, тоді як прогнозування результатів завтрашньої лотереї на основі даних за останні 50 років майже напевно не дасть ніяких результатів.

Стиснення даних і асоціативна пам'ять. Здібність нейромереж до виявлення взаємозв'язків між різними параметрами дає можливість виразити дані великої розмірності компактніше, якщо дані тісно взаємозв'язані один з одним. Зворотний процес - відновлення початкового набору даних з частини інформації – називається асоціативною пам'яттю. Асоціативна пам'ять дозволяє

також відновлювати початковий образ із зашумленних вхідних даних. Рішення задачі гетероассоциативной пам'яті дозволяє реалізувати пам'ять, що адресується по змісту.

Багатошаровий персептрон. Багатошаровий персептрон Розенблатта - персептрон з додатковими шарами A - елементів, розташованими між S і R елементами. Оскільки елементарний персептрон вже володів двома шарами зв'язків і трьома шарами елементів (нейронів), то такий персептрон не вважався багатошаровим, і багатошаровість зумовлювалась тільки за наявності мінімум чотирьох шарів елементів. Інша важлива відмінність полягає в тому, що не обов'язково всі зв'язки мають бути навчані, частина з них може бути випадково вибранна і фіксована.

Нейронна мережа Ворда. Штучна нейронна мережа, топологія якої характеризується тим, що внутрішні (приховані) шари нейронів розбиті на блоки (рис. 3.14):

1. Нейрони вхідного шару.
2. Нейрони блоку прихованого шару.
3. Нейрони вихідного шару.

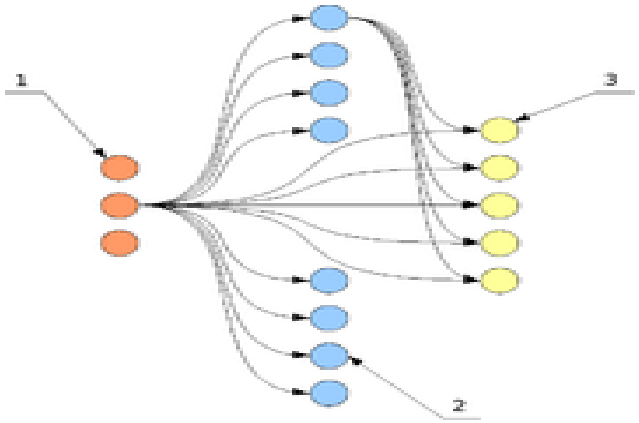


Рис. 3.14. Нейронна мережа Ворда с двома прихованими шарами.

Нейронні мережі Ворда розрізняються кількістю блоків прихованого шару і наявністю або відсутністю обхідних з'єднань. Розбиття прихованих шарів на блоки дозволяє використовувати різні передавальні функції для різних блоків

прихованого шару. Таким чином, одні і ті ж сигнали, отримані від вхідного шару, зважуються і обробляються паралельно з використанням декількох способів, а отриманий результат потім обробляється нейронами вихідного шару. Застосування різних методів обробки для одного і того ж набору даних дозволяє сказати, що нейронна мережа аналізує дані з різних аспектів. Практика показує, що мережа показує дуже добрі результати при рішенні задач прогнозування і розпізнавання образів. Для нейронів вхідного шару, як правило, встановлюється лінійна функція активації. Функція активації для нейронів з блоків прихованого і вихідного шару визначається експериментально.

Для навчання нейронної мережі Ворда можна застосовувати метод зворотного розповсюдження помилки.

Нейронна мережа Джордана. Мережа Джордана - це вид мереж , який виходить з багатошарового персептрона, якщо на його вхід подати крім вхідного вектора вихідний із затримкою на один або декілька тактів. В перших рекурентних мережах головною ідеєю було дати мережі бачити свій вихідний образ на попередньому кроці. У такої мережі тільки частина сенсорів приймає сигнали з навколишнього світу, на інші сенсори приходить вихідний образ з попереднього моменту часу. Розглянемо проходження послідовності сигналів через мережу. Сигнал поступає на групу сенсорів сполучених із зовнішнім світом (INPUT) і проходить в прихований шар (HIDDEN). Перетворений прихованим шаром сигнал піде на вихідний шар (OUTPUT) і вийде з мережі, а його копія потрапить на затримку. Далі в мережу, на сенсори, що сприймають зовнішні сигнали, поступає другий образ, а на контекстну групу сенсорів (CONTEXT) - вихідний образ з попереднього кроку із затримки. Далі зі всіх сенсорів сигнал піде в прихований шар, потім на вихідний (рис. 3.15).

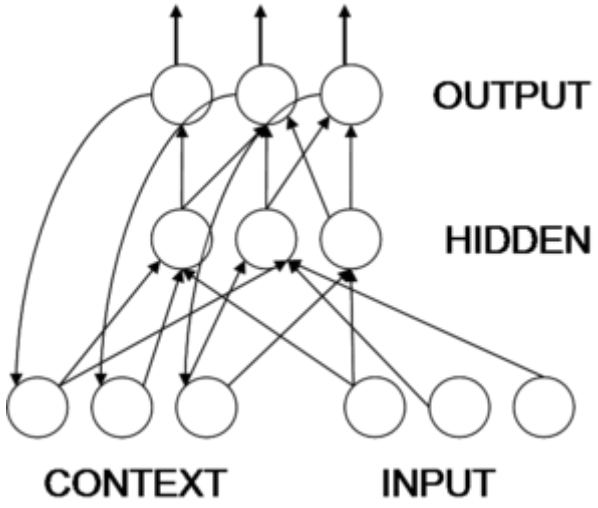


Рис.3.15. Архітектура рекурентної мережі Джордана.

Нейронна мережа Елмана. Нейронна мережа Елмана - один з видів рекурентної мережі, яка так само як і мережа Джордана виходить з багатошарового персептрона введенням зворотних зв'язків, тільки зв'язки йдуть не від виходу мережі, а від виходів внутрішніх нейронів. Це дозволяє врахувати передісторію спостережуваних процесів і накопичити інформацію для вироблення правильної стратегії управління.

Когнітрон та неокогнітрон. Когнітрон - штучна нейронна мережа на основі принципу самоорганізації. Своєю архітектурою когнітрон схожий на будову зорової кори, має ієрархічну багатошарову організацію, в якій нейрони між шарами зв'язані тільки локально. Навчається конкурентним навчанням (без вчителя). Кожен шар мозку реалізує різні рівні узагальнення; вхідний шар чутливий до простих образів, таких, як лінії, і їх орієнтації в певних областях візуальної області, тоді як реакція інших шарів є складнішою, абстрактнішою і незалежною від позиції образу.

Когнітрон конструюється у вигляді шарів нейронів, сполучених синапсами. Предсинаптіческій нейрон в одному шарі пов'язаний з постсинаптичним нейроном в наступному шарі. Є два типи нейронів: збудливі вузли, які прагнуть викликати збудження постсинаптичного вузла, і гальмуючі вузли, які гальмують це збудження. Збудження нейрона визначається зваженою сумою його збудливих і гальмуючих входів, проте насправді механізм є

складнішим, ніж просте підсумовування. Дана нейронна мережа одночасно є як моделлю процесів сприйняття на мікрорівні, так і обчислювальною системою, що застосовується для технічних завдань розпізнавання образів.

Неокогнітрон є подальшим розвитком ідеї когнітрона і точніше відображає будову зорової системи, дозволяє розпізнавати образи незалежно від їх перетворень, обертань, спотворень і змін масштабу. Неокогнітрон може як самонавчатися, так і навчатися з вчителем. Неокогнітрон отримує на вході двовимірні образи, аналогічні зображенням на сітчастій оболонці ока, і обробляє їх в подальших шарах аналогічно тому, як це було виявлено в зоровій корі людини. Звичайно, в неокогнітроне немає нічого, що обмежує його використання тільки для обробки візуальних даних, він достатньо універсальний і може знайти широке застосування як узагальнена система розпізнавання образів.

Хаотичні нейронні мережі. По структурі такі мережі є одношаровими рекурентними мережами, елементи яких зв'язані «кожен з кожним», без утворення зв'язку «сам на себе». Структуру мережі утворюють N нейронів, кожний з яких відповідає за конкретний об'єкт при цьому зв'язки симетричні. Навчання хаотичної мережі полягає у формуванні один раз вагових коефіцієнтів мережі, які визначають міру взаємного впливу пар нейронів один на одного. Основна відмінність від інших мереж полягає в утворенні групової поведінки нейронів і виділенні з первинного хаосу деякого унікального порядку, властивого тільки заданому вхідному набору об'єктів, представлених у вигляді окремих точок деякого умовного зображення. Прийнято вважати, що такі властивості нейронної мережі як великі обчислювальні можливості і стійкість до помилки в нейродинамічних системах обумовлюється колективною поведінкою всіх нейронів. Це стає можливим завдяки сильній взаємодії між нейронами, коли кожен пов'язаний з кожним. Має місце глобальний характер зв'язку: якщо не розрізняються миттєві стани елементів, то не відрізняється і діюче на них поле.

У хаотичній нейромережі на вхід в початковий момент часу подається відразу все зображення (представляються відразу всі дані, що підлягають кластеризації і лише один раз), на відміну від інших нейронних мереж, побудованих на основі шару змагання, де елементи представляються по черзі і неодноразово. Стан мережі задається станом нейронів. Щоб значення функції активації нейронів, коливалися в діапазоні [-1, 1], використовується спеціальна логістична функція

$$f(x) = 1 - 2x^2$$

Мережа відпускається на деякий період часу у вільне функціонування для того, щоб можна було побудувати фазові портрети поведінки, що відображають поведінку кожного нейрона окремо. Подача на вхід мережі набору об'єктів може бути розглянута як вплив навколишнього середовища на спочатку неврегульовану поведінку нейронів і формування під її дією нейронних ансамблів. При цьому сумісне функціонування нейронів є проявом інтелекту на рівні малого колективу, вирішального завдання кластеризації.

Розглядаючи сумісну динаміку поведінки нейронів після проходження перехідного процесу, виділяються пари нейронів, які міняють свої стани синхронно. Формуються дані про сумісну інформацію, яку несе кожна синхронізована пара нейронів і на її основі робиться висновок про те, які нейрони слід вважати такими, що відносяться до одного кластера. Первинний хаос перехідного процесу, з часом зменшується, оскільки за рахунок виявлення взаємозв'язку значень виходів деяких пар нейронів починають змінюватися синфазно. Саме тому робота нейромережі розділяється на дві частини: перехідний період роботи і період спостереження, коли вважається, що хаос мінімальний, і можна починати витягувати інформацію.

Осциляторні нейронні мережі. Осциляторні нейронні мережі - нейронні мережі, основними структурними одиницями яких є осцилятори. Функціонують такі мережі за рахунок коливань окремих елементів або груп елементів і їх взаємодії. Осциляторні мережі представляють науковий інтерес, оскільки істотну роль в розумових процесах людини грають коливання.

3.3. Нейронні мережі Хопфілда та Кохонена.

Багато нейронних мереж не мають зворотних зв'язків, тобто зв'язків, що йдуть від виходів мереж до їх входів. Відсутність зворотного зв'язку гарантує безумовну стійкість мереж. Вони не можуть увійти до режиму, коли вихід безперервно блукає від стану до стану і не придатний до використання. Але це вельми бажана властивість досягається не безкоштовно, мережі без зворотних зв'язків володіють більш обмеженими можливостями в порівнянні з мережами із зворотними зв'язками.

Оскільки мережі із зворотними зв'язками мають шляхи, що передають сигнали від виходів до входів, то відгук таких мереж є динамічним, тобто після появи нового входу обчислюється вихід і, передаючись по мережі зворотного зв'язку, модифікує вхід. Потім вихід повторно обчислюється, і процес повторюється знову і знову. Для стійкої мережі послідовні ітерації приводять до все меншим змінам виходу, поки врешті-решт вихід не стає постійним. Для багатьох мереж процес ніколи не закінчується, такі мережі називають *нестійкими*. Нестійкі мережі володіють цікавими властивостями і вивчалися як приклад хаотичних систем.

Проблема стійкості ставила в безвихідь перших дослідників. Ніхто не був в змозі передбачити, які з мереж будуть стійкими, а які знаходитимуться в постійній зміні. Більш того, проблема представлялася настільки важкою, що багато дослідників було налагоджена песимістично відносно можливості застосування нейронних мереж. На щастя, була отримана теорема, що описала підмножину мереж із зворотними зв'язками, виходи яких врешті-решт досягають стійкого стану. Це чудове досягнення відкрило дорогу подальшим дослідженням і сьогодні багато вчених займаються дослідженням складної поведінки і можливостей цих систем. Дж. Хопфілд зробив важливий вклад як в теорію, так і у вживання систем із зворотними зв'язками. Тому деякі з конфігурацій відомі як мережі Хопфілда [22, 34, 42].

Розглянемо мережу із зворотними зв'язками, що складається з двох шарів (рис. 3.16). Нульовий шар не виконує обчислювальної функції, а лише розподіляє виходи мережі назад на входи. Кожен нейрон першого шару обчислює зважену суму своїх входів, даючи сигнал NET, який потім за допомогою нелінійної функції F перетвориться в сигнал OUT. Ці операції схожі з нейронами інших мереж.

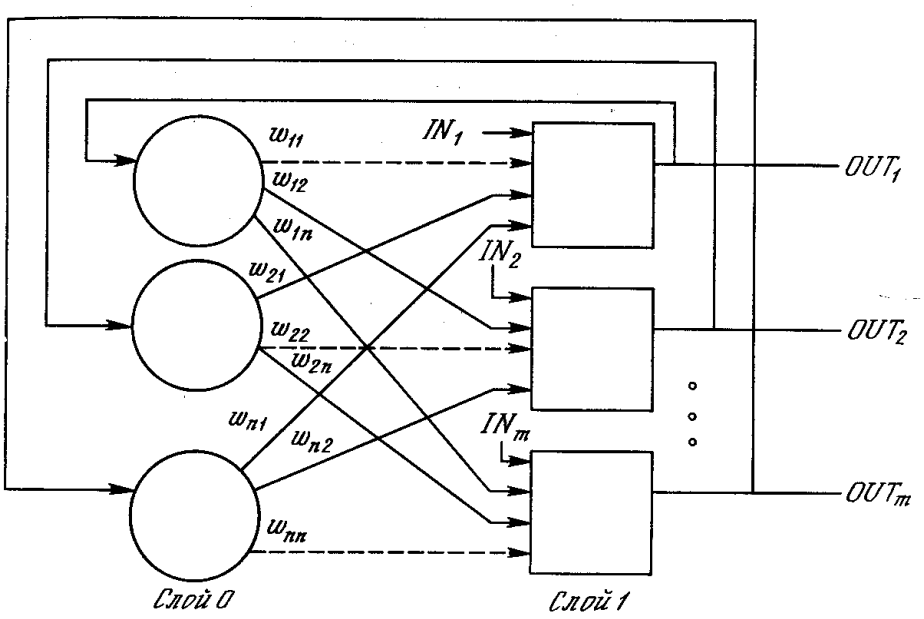


Рис. 3.16. Одношарова мережа з зворотними зв’язками.

В перших варіантах нейронних мереж Хопфілда функція F була просто пороговою функцією. Вихід такого нейрона дорівнює одиниці, якщо зважена сума виходів з інших нейронів більше порогу T_j, інакше вона дорівнює нулю. Він обчислюється таким чином:

$$NET_j = \sum_{i \neq j} w_{ij} OUT_i + IN_j,$$

$$OUT = 1, якщо NET_j > T_j,$$

$$OUT = 0, якщо NET_j < T_j$$

OUT не вимірюється, якщо $NET_j = T_j$.

Стан мережі – це просто множина поточних значень сигналів OUT від всіх нейронів. Оскільки виходом бінарного нейрона може бути лише нуль або одиниця (проміжних рівнів немає), то поточний стан мережі є двоїчним числом, кожен біт якого є сигналом OUT деякого нейрона.

Функціонування мережі легко візуалізується геометрично. На рисунку 3.17 показаний випадок двох нейронів у вихідному шарі, причому кожній вершині квадрата відповідає один з чотирьох станів системи (00, 01, 10, 11). На рисунку 3.18 показана трьохнейронна система, представлена кубом, що має вісім вершин, кожна з яких помічена трьохбітовим бінарним числом. У загальному випадку система з n нейронами має $2n$ різних станів і представляється n - мірним гіперкубом.

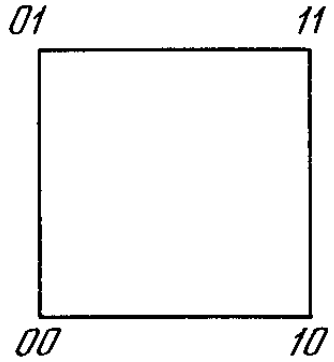


Рис. 3.17. Два нейрона породжують систему з чотирьмя станами.

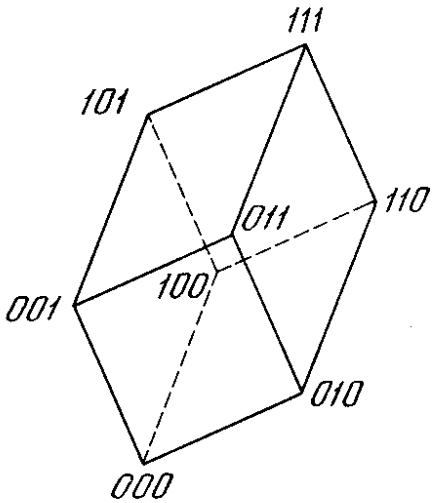


Рис. 3.18. Три нейрона породжують систему з восьма станами.

Коли подається новий вхідний вектор, мережа переходить з вершини у вершину, поки не стабілізується. Стійка вершина визначається мережевими вагами, поточними входами і величиною порогу. Якщо вхідний вектор частково неправильний або неповний, то мережа стабілізується у вершині, найближчій до бажаної.

Людська пам'ять асоціативна, тобто деякий спогад може породжувати велику пов'язану з ним область. Наприклад, декілька музичних тактів можуть викликати цілу гамму спогадів, включаючи пейзажі, звуки і запахи. Навпаки, звичайна комп'ютерна пам'ять є локально адресуємою, пред'являється адреса і витягується інформація за цією адресою. Мережа із зворотним зв'язком формує асоціативну пам'ять. Подібно до людської пам'яті по заданій частині потрібної інформації вся інформація витягується з «пам'яті». Аби організувати асоціативну пам'ять за допомогою мережі із зворотними зв'язками, ваги повинні вибиратися так, щоб утворювати енергетичні мінімуми в потрібних вершинах одиничного гіперкуба. Хопфілд розробив асоціативну пам'ять з безперервними виходами, що змінюються в межах від +1 до –1, відповідних двоїчним значенням 0 і 1. Інформація, що запам'ятовується, кодується двоїчними векторами і зберігається у вагах згідно наступній формулі:

$$w_{ij} = \sum_{d=1..m} (OUT_{i,d} OUT_{j,d}),$$

де m – число вихідних векторів, що запам'ятовуються; d – номер вихідного вектора, що запам'ятовується; $OUT_{i,j}$ – і-компонента вихідного вектора, що запам'ятовується.

Цей вираз може стати яснішим, якщо відмітити, що ваговий масив W може бути знайдений обчисленням зовнішнього множиння кожного вектора, що запам'ятовується, з самим собою і підсумовуванням матриць, отриманих таким чином. Це може бути записано у вигляді

$$W = \sum_i D_i^T D_i ,$$

де D_i – і-й вектор-рядок, що запам'ятовується.

Як тільки ваги задані, мережа може бути використана для здобуття вихідного вектора, що запам'ятався, по даному вхідному вектору, який може бути частково неправильним або неповним. Для цього виходам мережі спочатку надають значення цього вхідного вектора. Потім вхідний вектор забирається і мережі надається можливість «розслабитися», опустившись в найближчий глибокий мінімум. Мережа йде по локальному нахилу функції енергії і може бути захоплена локальним мінімумом, не досягнувши найкращого в глобальному сенсі рішення.

Недоліком мереж Хопфілда є їх тенденція стабілізуватися в локальному, а не глобальному мінімумі функції енергії. Ця трудність долається в основному за допомогою класу мереж, відомих під назвою машин Больцмана, в яких зміни станів нейронів обумовлені статистичними, а не детермінованими закономірностями. Існує тісна аналогія між цими методами і відпалом металу, тому і самі методи часто називають імітацією відпалу.

Розглянемо застосування нейронної мережі Хопфілда на прикладі розв’язання відомої задачі комівояжера.

Задача комівояжера є оптимізаційною задачею, що часто виникає на практиці. Вона може бути сформульована таким чином: для деякої групи міст із заданими відстанями між ними потрібно знайти найкоротший маршрут з відвідинами кожного міста один раз і з поверненням у вихідну точку. Було доведено, що ця задача належить великій множині задач, названих «NP-повними». Для NP-повних задач не відомо кращого методу розв’язання, ніж повний перебір усіх можливих варіантів. Оскільки такий повний пошук практично нездійснений для великого числа міст, то евристичні методи використовуються для знаходження прийнятних, хоча і неоптимальних рішень.

Запропоноване рішення, засноване на мережах із зворотними зв'язками, є типовим в цьому відношенні. Відповідь знаходиться так швидко, що в певних випадках метод може виявитися корисним. Припустимо, що міста, які необхідно відвідати, помічені буквами A, B, C і D, а відстані між парами міст є d_{ab}, d_{bc} і т. д. Рішенням є впорядкована множина з n міст. Завдання полягає у

відображенні його в обчислювальну мережу з використанням нейронів. Кожне місто представлене рядком з n нейронів. Вихід одного і лише одного нейрона з них дорівнює одиниці (всі інші дорівнюють нулю). Цей вихід нейрона показує порядковий номер, в якому дане місто відвідується при обході. На рисунку 3.19 показаний випадок, коли місто C відвідується першим, місто A – другим, місто D – третім і місто B – четвертим.

Порядок дотримування

Місто	1	2	3	4
A	0	1	0	0
B	0	0	0	1
C	1	0	0	0
D	0	0	1	0

Рис. 3.19. Варіант розвязання задачі комівояжера

Для такого варіанта потрібно n^2 нейронів – число, яке швидко зростає із збільшенням числа міст. Довжина такого маршруту була б рівна $d_{ca} + d_{ad} + d_{db} + d_{bc}$. Оскільки кожне місто відвідується лише один раз і в кожен момент відвідується лише одне місто, то в кожному стовпці є по одній одиниці. Для задачі з *n* містами всього є *n***!** різних маршрутів обходу. Якщо *n* = 60, то існує $6934155*10^{78}$ можливих маршрутів. Якщо прийняти до уваги, що в нашій галактиці є лише 10^{11} зірок, то стане зрозумілим, що повний перебір всіх можливих маршрутів для 1000 міст навіть на найшвидшому в світі комп'ютері займе час, порівнянний з геологічною епохою.

Продемонструємо тепер, як сконструювати мережу для вирішення цієї NP-повної проблеми. Кожен нейрон забезпечений двома індексами, які відповідають місту і порядковому номеру його відвідин в маршруті. Наприклад, $OUT_{X_j} = 1$ показує, що місто *x* було *j-им* по порядку містом маршруту.

Функція енергії повинна задовольняти двом вимогам: по-перше, повинна бути малою тільки для тих рішень, які мають по одній одиниці в кожному рядку і в кожному стовпці; по-друге, повинна надавати перевагу рішенням з короткою довжиною маршруту. Перша вимога задовольняється введенням наступної, складеної з трьох сум, функції енергії:

$$E = \frac{A}{2}\sum_{x}\sum_{i}\sum_{j\neq i}\mathrm{OUT}_{xi}\mathrm{OUT}_{xj} + \frac{B}{2}\sum_{i}\sum_{x}\sum_{y\neq x}\mathrm{OUT}_{xi}\mathrm{OUT}_{yi} + \frac{C}{2}\left[\left(\sum_{x}\sum_{i}\mathrm{OUT}_{xi}\right) - n\right]^2,$$

де A, B і C - деякі константи. Цим досягається виконання наступних умов:

1. Перша потрійна сума дорівнює нулю в тому і лише в тому випадку, якщо кожен рядок (місто) містить не більш за одну одиницю.
2. Друга потрійна сума дорівнює нулю в тому і лише в тому випадку, якщо кожен стовпець (порядковий номер відвідин) містить не більш за одну одиницю.
3. Третя сума дорівнює нулю в тому і лише в тому випадку, якщо матриця містить рівно *n* одиниць.

Друга вимога - перевага коротким маршрутам - задовольняється за допомогою додавання наступного члена до функції енергії:

$$E = \frac{D}{2}\sum_{x}\sum_{y\neq x}\sum_{i} d_{xy}\mathrm{OUT}_{xi}(\mathrm{OUT}_{y,i+1} + \mathrm{OUT}_{y,i-1}).$$

Відмітимо, що цей член є довжиною будь-якого допустимого маршруту. Для зручності індекси визначаються по модулю *n*, тобто $OUT_{n+j} = OUT_j$, а D - деяка константа. При достатньо великих значеннях A, B і C низькоенергетичні стани представлятимуть допустимі маршрути, а великі значення D гарантують, що буде знайдений короткий маршрут.

Тепер задамо значення вагів, тобто встановимо відповідність між членами у функції енергії і членами загальної форми. Отримуємо

$w_{xi,yi} = -A\delta_{xy}(1 - \delta_{ij})$ (не допускає більш за одну одиницю в рядку)

$-B\delta_{ij}(1 - \delta_{xy})$ (не допускає більш за одну одиницю в стовпці)

$-C$ (глобальне ообмеження)

$-Dd_{xy}(\delta_{j,i+1} + \delta_{j,i-1})$ (член, що відповідає за довжину циклу),

де $\delta_{ij} = 1$, якщо $i = j$, інакше $\delta_{ij} = 0$.

Результати експерименту, в якому задача комівояжера була вирішена для 10 міст продемонстрували, що 16 з 20 прогонів зійшлися до допустимого маршруту і близько 50% рішень виявилися найкоротшими маршрутами, як це було встановлено за допомогою повного перебору. Цей результат стане більш вражаючим, якщо усвідомити, що є 181440 допустимих маршрутів.

Розглянемо практичне застосування нейромереж на прикладі розв'язання задачі «Видача кредиту клієнтові» в аналітичному пакеті Deductor. В якості навчального набору даних виступає база даних, що містить інформацію про клієнтів, зокрема: Сума кредиту, Термін кредиту, Мета кредитування, Вік, Рід, Освіта, Приватна власність, Квартира, Площа квартири. На основі цих даних необхідно побудувати модель, яка зможе дати відповідь, чи входить клієнт, охочий отримати кредит, в групу ризику неповернення кредиту, тобто користувач повинен отримати відповідь на питання чи «Видавати кредит?». Завдання відноситься до групи завдань класифікації, тобто навчання з вчителем.

Дані для аналізу знаходяться у заздалегідь підготовленому файлі. Імпортуємо дані з файлу за допомогою майстра імпорту. Запускаємо майстер обробки і вибираємо метод обробки даних - нейронна мережа. Задаємо призначення початкових стовпців даних. Вихідний стовпець в нашому завданні – «Давати кредит», всі інші – вхідні (рис. 3.20).

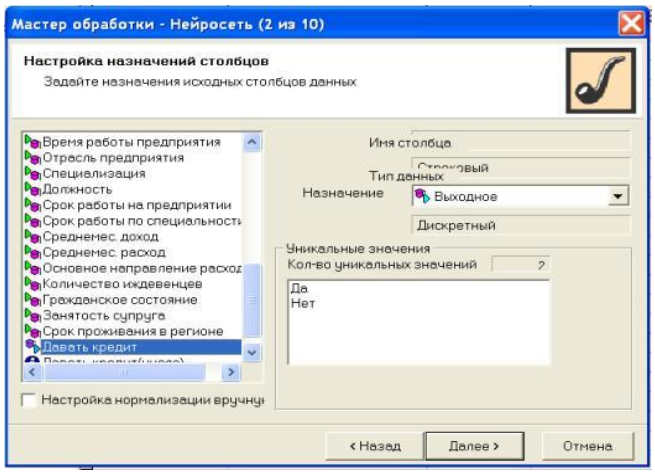


Рис. 3.20. Настройка призначень стовпців.

На наступному кроці розбиваємо початкову множину даних на навчальну і тестову та визначаємо структуру нейронної мережі, тобто вказуємо кількість нейронів у вхідному шарі - 33 (кількість вхідних змінних), в прихованому шарі - 1, у вихідному шарі - 1 (кількість вихідних змінних). Активаційна функція - Сигмоїда, і її крутизна рівна одиниці (рис. 3.21).

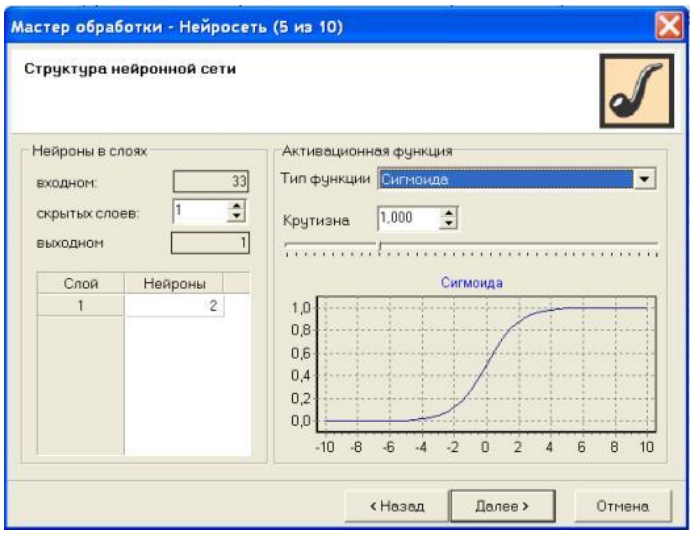


Рис. 3.21. Структура нейтронної мережі.

Далі вибираємо алгоритм і параметри навчання нейронної мережі і настроюємо умови зупинки навчання. Вважатимемо приклад розпізнаним, якщо помилка менше 0.005, і вкажемо умову зупинки навчання досягши епохи 10000. Запускаємо процес навчання і спостерігаємо за зміною величини помилки і відсотком розпізнаних прикладів в навчальній і тестовій множинах. У нашому випадку ми бачимо, що на епосі № 4536 в навчальній множині розпізнано 83,10% прикладів, а на тестовому - 85,71% прикладів (рис. 3.22).

Після закінчення процесу навчання для інтерпретації отриманих результатів ми маємо можливість вибрати візуалізатори із списку запропонованих, зокрема, таблицю зв'язаності, граф нейросеті, аналіз "що, якщо", і за допомогою їх проаналізувати отримані дані.

На рисунку 3.23 показана таблиця зв'язаності. По її діагоналі розташовані приклади, які були правильно розпізнані, тобто 55 клієнтів, яким можна видавати кредит, і 89 клієнтів, яким видавати кредит не варто. У решті

осередків розташовані ті клієнти, які були віднесені до іншого класу (1 і 4). Можна вважати, що правильно класифіковані практично всі приклади - 96,64%.

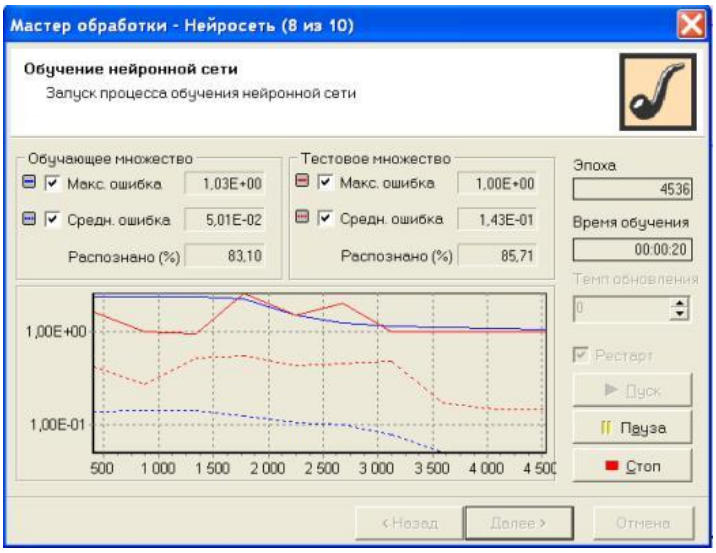
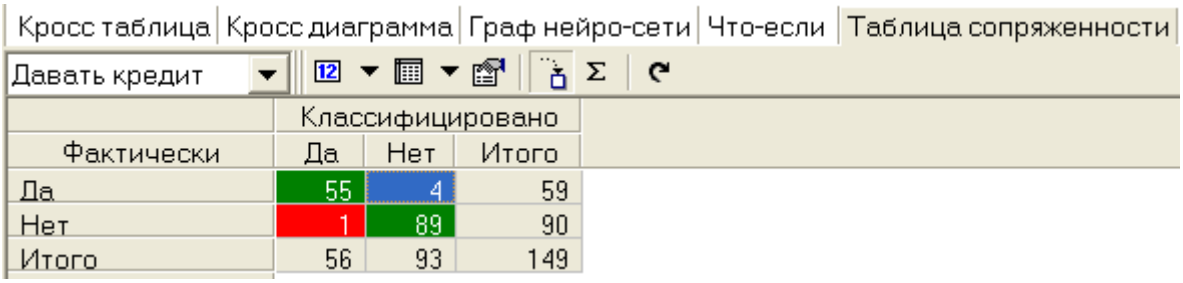


Рис. 3.22. Навчання нейтронної мережі.



	Классифицировано		
Фактически	Да	Нет	Итого
Да	55	4	59
Нет	1	89	90
Итого	56	93	149

Рис. 3.23. Данні таблиці зв'язаності.

Тепер нейронна мережа підготовлена для практичного використання.

Важливим напрямком в розвитку нейронних мереж є самоорганізуючи карти (Self-Organizing Maps, SOM) або мережи (карти) Кохонена [21, 41, 53].

Мережі, названі картами Кохонена, - це один з різновидів нейронних мереж, проте вони принципово відрізняються від розглянутих вище, оскільки використовують неконтрольоване навчання. Нагадаємо, що при такому навчанні навчальна множина складається лише із значень вхідних змінних, в процесі навчання немає порівняння виходів нейронів з еталонними значеннями. Можна сказати, що така мережа вчиться розуміти структуру даних.

Ідея мережі Кохонена належить фінському ученому Тойво Кохонену (1982 рік). Основний принцип роботи мереж - введення в правило навчання

нейрона інформації щодо його розташування. У основі ідеї мережі Кохонена лежить аналогія з властивостями людського мозку. Кора головного мозку людини є плоским листом і згорнута складками. Таким чином, можна сказати, що вона володіє певними топологічними властивостями (ділянки, відповідальні за близькі частини тіла, примикають один до одного і все зображення людського тіла відображається на цю двовимірну поверхню).

Карти, що самоорганізовуються, можуть використовуватися для вирішення таких задач, як моделювання, прогнозування, пошук закономірностей у великих масивах даних, виявлення наборів незалежних ознак і стиснення інформації. Найбільш поширене застосування мереж Кохонена - рішення задачі класифікації без вчителя, тобто кластеризації. Нагадаємо, що при такій постановці задачі нам дан набір об'єктів, кожному з яких зіставлений рядок таблиці (вектор значень ознак). Потрібно розбити початкову множину на класи, тобто для кожного об'єкту знайти клас, до якого він належить. В результаті отримання нової інформації про класи можлива корекція існуючих правил класифікації об'єктів.

Ось два з найбільш поширених застосувань карт Кохонена: розвідувальний аналіз даних і виявлення нових явищ.

Розвідувальний аналіз даних. Мережа Кохонена здатна розпізнавати кластери даних, а також встановлювати близькість класів. Таким чином, користувач може поліпшити своє розуміння структури даних, щоб потім уточнити нейромережеву модель. Якщо в даних розпізнані класи, то їх можна позначити, після чого мережа зможе вирішувати задачі класифікації. Мережі Кохонена можна використовувати і в тих завданнях класифікації, де класи вже задані, - тоді перевага буде в тому, що мережа зможе виявити схожість між різними класами.

Виявлення нових явищ. Мережа Кохонена розпізнає кластери в навчальних даних і відносить всі дані до тих або інших кластерів. Якщо після цього мережа зустрінеться з набором даних, несхожим ні на один з відомих

зразків, то вона не зможе класифікувати такий набір і тим самим виявить його новизну.

Мережа Кохонена, на відміну від багатошарової нейронної мережі, дуже проста, вона представляє собою два шари: вхідний і вихідний. Її також називають самоорганізуючою картою. Елементи карти розташовуються в деякому просторі, як правило, двовимірному (рис. 3.24).

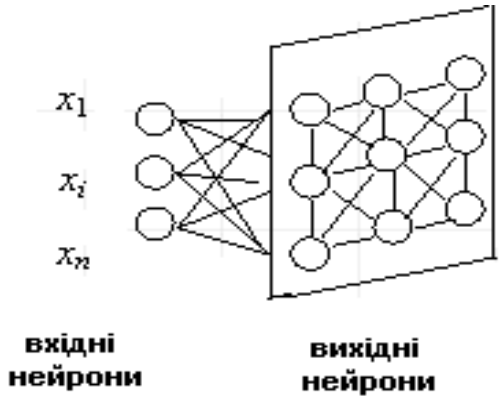


Рис. 3.24. Мережа Кохонена.

Мережа Кохонена навчається методом послідовних наближень. В процесі навчання таких мереж на входи подаються дані, але мережа при цьому підстроюється не під еталонне значення виходу, а під закономірності у вхідних даних. Починається навчання з вибраного випадковим чином вихідного розташування центрів. В процесі послідовної подачі на вхід мережі навчальних прикладів визначається найбільш схожий нейрон (той, у якого скалярне множення вагів і поданого на вхід вектора мінімально). Цей нейрон оголошується переможцем і є центром при підстроюванні вагів у сусідніх нейронів. Таке правило навчання припускає «змагальне» навчання з урахуванням відстані нейронів від «нейрона-переможця». Навчання при цьому полягає не в мінімізації помилки, а в підстроюванні вагів (внутрішніх параметрів нейронної мережі) для найбільшого збігу з вхідними даними.

Основний ітераційний алгоритм Кохонена послідовно проходить ряд епох, на кожній з яких обробляється один приклад з навчальної вибірки. Вхідні сигнали послідовно пред'являються мережі, при цьому бажані вихідні сигнали

не визначаються. Після пред'явлення достатнього числа вхідних векторів синаптіческіє ваги мережі стають здатні визначити кластери. Ваги організовуються так, що топологічно близькі вузли чутливі до схожих вхідних сигналів. В результаті роботи алгоритму центр кластера встановлюється в певній позиції, задовільним чином кластерізующей приклади, для яких даний нейрон є «переможцем». В результаті навчання мережі необхідно визначити міру сусідства нейронів, тобто околицю нейрона-переможця. Околицею є декілька нейронів, які оточують нейрон-переможець. Спочатку до околиці належить велике число нейронів, далі її розмір поступово зменшується. Мережа формує топологічну структуру, в якій схожі приклади утворюють групи прикладів, що близько знаходяться на топологічній карті. Отриману карту можна використовувати як засіб візуалізації при аналізі даних. В результаті навчання карта Кохонена класифікує вхідні приклади на кластери (групи схожих прикладів) і візуально відображає багатовимірні вхідні дані на площині нейронів. Унікальність методу карт, що самоорганізовуються, полягає в перетворенні n-мірного простору в двомірне. Застосування двомірних сіток пов'язане з тим, що існує проблема відображення просторових структур більшої розмірності. Маючи таке представлення даних, можна візуально визначити наявність або відсутність взаємозв'язку у вхідних даних. Нейрони карти Кохонена розташовують у вигляді двомірної матриці, розфарбовують цю матрицю залежно від аналізованих параметрів нейронів (рис. 3.25).

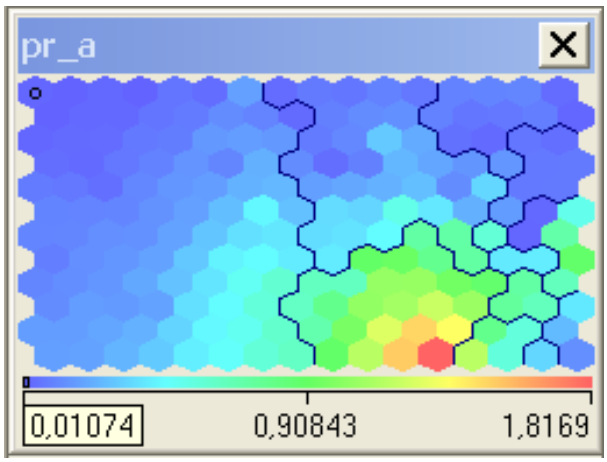


Рис. 3.25. Приклад карти Кохонена.

Що ж означає її розфарбовування? На рисунку 3.26 приведено розфарбовування карти, а точніше, її деякої ознаки, в тривимірному уявленні. Як ми бачимо, темно-сині ділянки на карті відповідають найменшим значенням показника, червоні - найвищим.

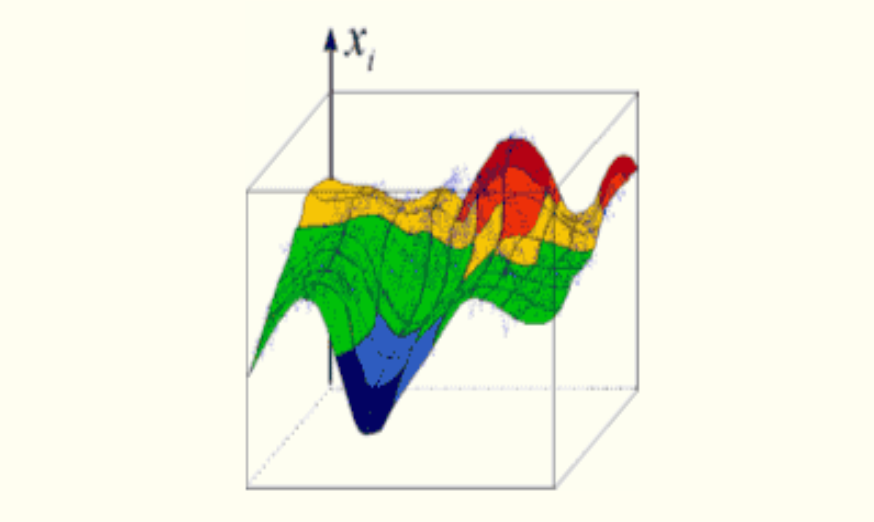


Рис. 3.26. Розфарбування ознаки в тривимірному просторі.

Тепер ми можемо сказати, які об'єкти мають найбільші значення даного показника (група об'єктів, позначена червоним кольором), а які - найменші значення (група об'єктів, позначена синім кольором). Таким чином, карти Кохонена (як і географічні карти) можна відображати:

- у двомірному вигляді, тоді карта розфарбовується відповідно до рівня виходу нейрона;
- у тривимірному вигляді.

В результаті роботи алгоритму отримуємо такі карти:

- карта входів нейронів;
- карта виходів нейронів;
- спеціальні карти.

Координати кожної карти визначають положення одного нейрона. Так, координати [15:30] визначають нейрон, який знаходиться на перетині 15-го стовпця з 30-м поряд в матриці нейронів. Розглянемо, що ж є ці карти.

Карта входів нейронів. Ваги нейронів підстроюються під значення вхідних змінних і відображають їх внутрішню структуру. Для кожного входу

малюється своя карта, розфарбована відповідно до значення конкретної ваги нейрона. При аналізі даних використовують декілька карт входів. На одній з карт виділяють область певного кольору - це означає, що відповідні вхідні приклади мають приблизно однакове значення відповідного входу. Колірний розподіл нейронів з цієї області аналізується на інших картах для визначення схожих або відмітних характеристик.

Карта виходів нейронів. На карту виходів нейронів проектується взаємне розташування досліджуваних вхідних даних. Нейрони з однаковими значеннями виходів утворюють кластери - замкнуті області на карті, які включають нейрони з однаковими значеннями виходів.

Спеціальні карти. Це карта кластерів, матриця відстаней, матриця щільності попадання і інші карти, які характеризують кластери, отримані в результаті навчання мережі Кохонена.

Важливо розуміти, що між всіма розглянутими картами існує взаємозв'язок - всі вони є різними розфарбовуваннями одних і тих же нейронів. Кожен приклад з навчальної вибірки має одне і те ж розташування на всіх картах.

Програмне забезпечення, що дозволяє працювати з картами Кохонена, зараз представлене безліччю інструментів. Це можуть бути як інструменти, що включають тільки реалізацію методу карт, що самоорганізовуються, так і нейропакети з цілим набором структур нейронних мереж. Також даний метод реалізований в деяких універсальних інструментах аналізу даних.

До інструментарію, що включає реалізацію методу карт Кохонена, відносяться SoMine, Statistica, NeuroShell, NeuroScalp, Deductor і багато інших. Для вирішення задач використовуватимемо аналітичний пакет Deductor. Хай є база даних комерційних банків з показниками діяльності за поточний період. Необхідно провести їх кластеризацію, тобто виділити однорідні групи банків на основі показників з бази даних, всього показників - 21. Початкова таблиця знаходиться у файлі «banks.xls». Вона містить показники діяльності

комерційних банків за звітний період. Спочатку імпортуємо дані з xls-файлу в середу аналітичного пакету.

На першому кроці запускаємо майстер обробки і вибираємо із списку метод обробки «Карта Кохонена». Далі слід настроїти призначення стовпців, тобто для кожного стовпця вибрати одне з призначень: вхідне, вихідне, не використовується і інформаційне. Вкажемо всім стовпцям, відповідно до показників діяльності банків, призначення «Вхідні». «Вихідний» не призначаємо. Наступний крок пропонує розбити початкову множину на навчальну, тестову і валідаційну. За умовчанням, програма пропонує розбити множину на навчальне - 95% і тестове - 5%. Ці кроки аналогічні крокам в майстрові обробки для нейронних мереж. Далі задаємо параметри карти: кількість ячеек по X і по Y, їх форму (шестикутну або чотирикутну), встановлюємо параметри зупинки навчання і встановлюємо епоху, по досягненню якої навчання буде припинено.

На сьомому кроці настроюються інші параметри навчання: спосіб початкової ініціалізації, тип функції сусідства. Можливі два варіанти кластеризації: автоматичне визначення числа кластерів з відповідним рівнем значущості і фіксована кількість кластерів (визначається користувачем). Оскільки нам невідома кількість кластерів, виберемо автоматичне визначення їх кількості (рис. 3.27).

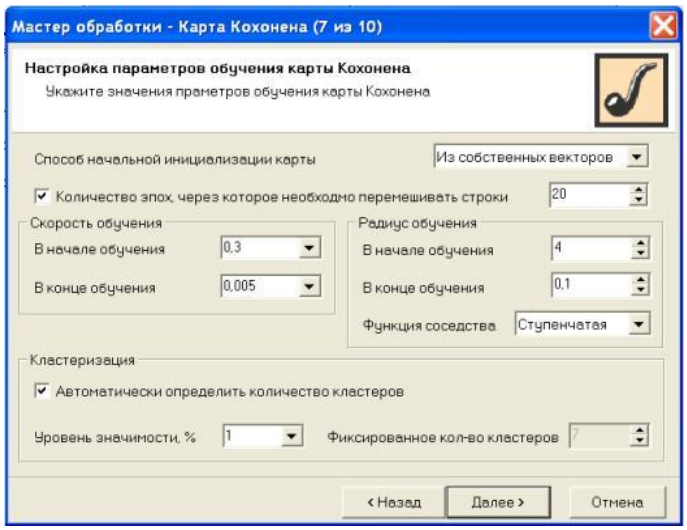


Рис. 3.27. Настройка параметрів навчання.

На восьмому кроці запускаємо процес навчання мережі. Під час навчання можемо спостерігати зміну кількості розпізнаних прикладів і поточні значення помилок. Цей процес аналогічний тому, що ми розглядали при навчанні нейронних мереж. Після закінчення навчання в списку візуалізаторов виберемо «Карту Кохонена» і візуалізатор «Що-коли». На останньому кроці настроюємо відображення карти Кохонена.

Вкажемо відображення всіх вхідних, вихідних стовпців, кластерів. При аналізі карт входів рекомендують використовувати відразу декілька карт. Досліджуємо фрагмент карти, що складається з карт трьох входів, який приведений на рис. 3.28.

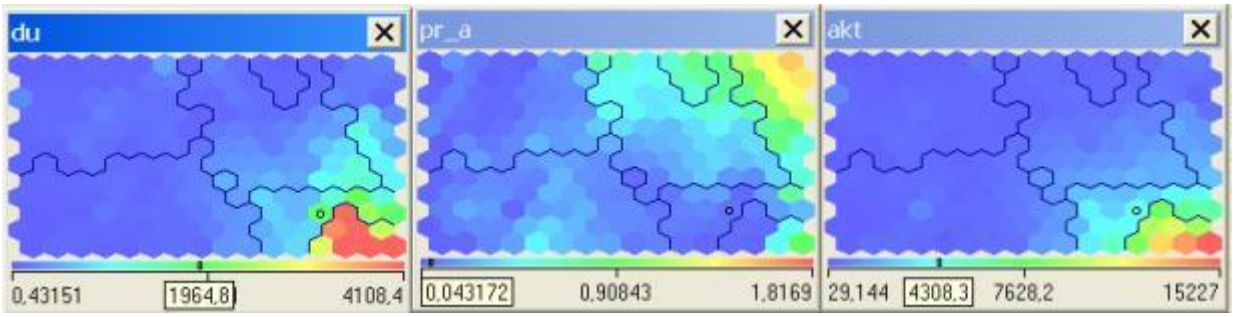


Рис. 3.28. Карти трьох входів.

На одній з карт виділяємо область з найбільшими значеннями показника. Далі має сенс вивчити ці ж нейрони на інших картах. На першій карті найбільші значення мають об'єкти, розташовані в правому нижньому кутку. Розглядаючи одночасно три карти, ми можемо сказати, що ці ж об'єкти мають найбільші значення показника, зображеного на третій карті. Також по розфарбовуванню першої і третьої карти можна зробити висновок, що існує взаємозв'язок між цими показниками. Також ми можемо визначити, наприклад, таку характеристику: кластер, розташований в правому верхньому кутку, характеризується низькими значеннями показників «депозити юридичних осіб» і «активи банку» і високими значеннями показників «прибутковість активів». Ця інформація дозволяє так охарактеризувати кластер, що знаходиться в правому верхньому кутку: це банки з невеликими активами, невеликими привернутими депозитними засобами від юридичних осіб, але з найбільш

прибутковими активами, тобто це група невеликих, але найбільш прибуткових банків. Це лише фрагмент висновку, який можна зробити, досліджуючи карту.

На рисунку 3.29 приведена ілюстрація карт входів і виходів, остання - це карта кластерів. Ми бачимо декілька карт входів (показників діяльності банків) і сформовані кластери, кожен з яких виділений окремим кольором.

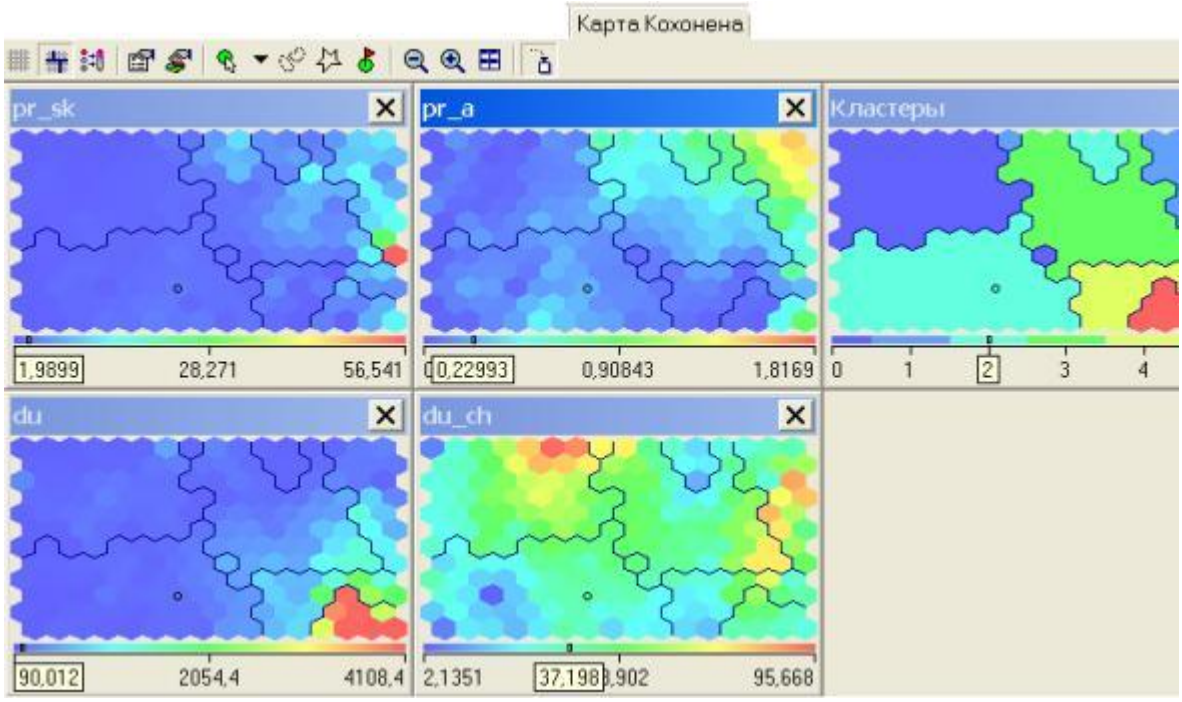


Рис. 3.29. Карти входів та виходів.

Для знаходження конкретного об'єкту на карті необхідно натискувати правою кнопкою миші на досліджуваному об'єкті і вибрати пункт «Знайти ячейку на карті» (рис. 3.30). В результаті ми можемо бачити як сам об'єкт, так і значення того виміру, який ми переглядаємо. Таким чином, ми можемо оцінити положення аналізованого об'єкту, а також порівняти його з іншими об'єктами.

В результаті застосування карт, що самоорганізуються, багатовимірний простір вхідних чинників був представлений в двомірному вигляді, в якому його досить зручний аналізувати. Зокрема, банки були класифіковані на 7 груп, для кожної з яких можливе визначення конкретних характеристик, виходячи з розфарбовування відповідних показників.

Таким чином, мережі Кохонена дозволяють спростити багатовимірну структуру, їх можна розглядати як одним з методів проектування багатовимірного простору в простір з нижчою розмірністю. Інтенсивність кольору в певній точці карти визначається даними, які туди попали: ячейка з мінімальними значеннями зображаються темно-синім кольором, ячейка з максимальними значеннями - червоним.

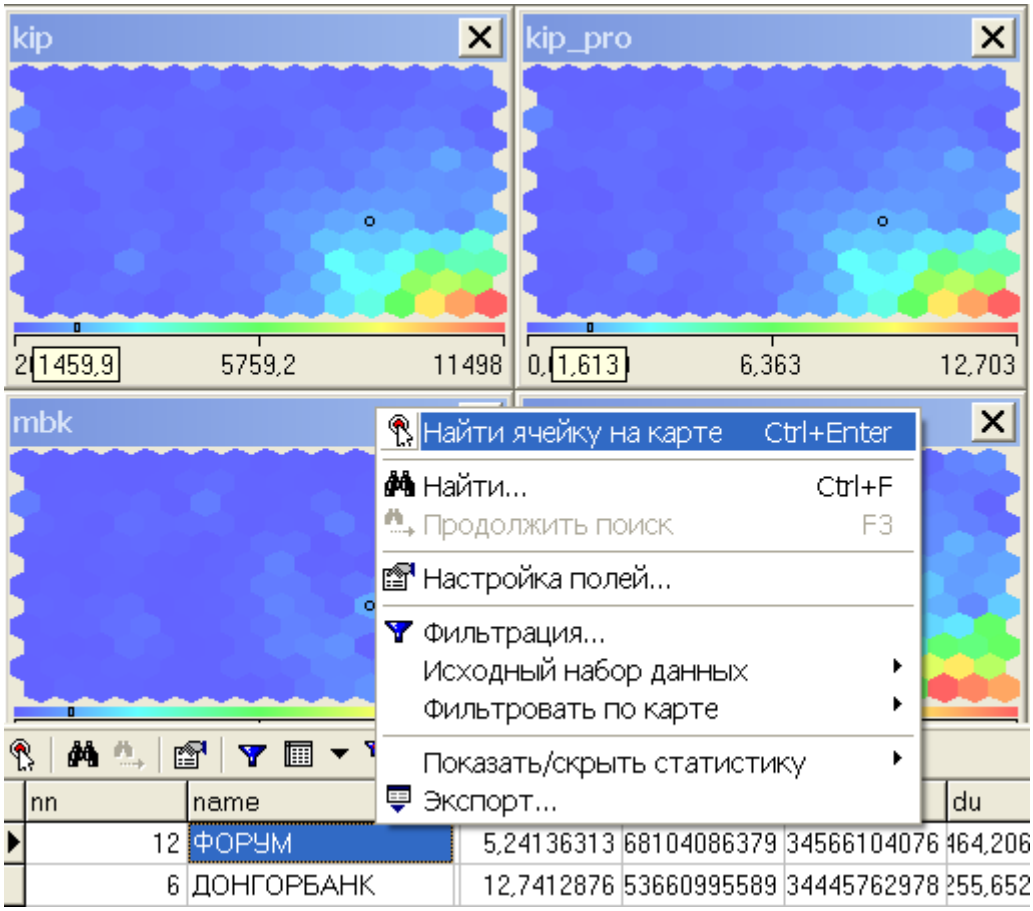


Рис. 3.30. Ячейка на карті Кохонена.

Інша принципова відмінність карт Кохонена від інших моделей нейронних мереж - інший підхід до навчання, а саме - некероване або неконтрольоване навчання. Цей тип навчання дозволяє даним навчальної вибірки містити значення лише вхідних змінних. Мережа Кохонена вчиться розуміти саму структуру даних і вирішує завдання кластеризації.

3.4. Програмні засоби реалізації нейромережевих технологій.

Нейронні мережі активно застосовуються в задачах аналізу вже більше 10 років. На фоні постійно змінної динаміки ринку, інтерес до нейромережевих технологій не лише не слабшає, а зростає з кожним роком. Зараз вже накопичений багатий досвід успішного використання нейронних мереж в практичних застосуваннях. По кількості реальних впроваджень лідирують системи інтелектуального аналізу даних в бізнесі і в управлінні процесами.

Більшість програмних комплексів, заснованих на нейромережевих технологіях, включають наступну послідовність дій:

- створення мережі (вибір користувачем параметрів або схвалення встановлених за умовчанням);
- навчання мережі;
- видача користувачеві рішення.

Існує величезна різноманітність нейропакетів, можливість використання нейромереж включена також практично у всі відомі статистичні пакети. Критерії порівняння нейропакетів: простота вживання, наочність інформації, що представляється, можливість використовувати різні структури, швидкість роботи, наявність документації. Вибір визначається кваліфікацією і вимогами користувача [2, 17, 34, 41, 53,70].

Для застосування методів нейронних мереж в процесі інтелектуального аналізу даних в бізнес-додатках розроблений ряд інструментальних засобів високого рівня. До них відносяться в першу чергу системи MathLab, STATISTICA Neural Networks, NeuroSolutions, BrainMakerPro, NeuroShell 2, 4Thought, SENN Sales та інші.

З метою більшої наочності розглянемо особливості застосування різних програмних комплексів підтримки нейромережевих технологій на конкретних практичних прикладах.

Пакет MathLab. Пакет MathLab надає користувачам можливість роботи з нейронними мережами. Стандартне постачання MathLab «Neural

Network Toolbox» надає широкі можливості для роботи з нейронними мережами всіх типів. Перевага цього пакету полягає в тому, що при його використанні користувач не обмежений моделями нейронних мереж і їх параметрами, жорстко закладеними в нейросимуляторі, а має можливість самостійно сконструювати ту мережу, яку рахує оптимальною для вирішення поставленого завдання. Пакет містить функції командного рядка і графічний інтерактивний майстер для швидкого покрокового створення нейромереж. Окрім цього Neural Network Toolbox забезпечує підтримку Simulink, що дозволяє моделювати нейромережі і створювати блоки на основі розроблених нейромережевих структур. Ключовими можливостями Neural Network Toolbox є:

- графічний інтерфейс користувача і майстер покрокового створення нейронних мереж;
- підтримка найбільш поширених мережевих парадигм;
- повний набір засобів для тренування нейромереж з вчителем і без;
- динамічно навчаємі нейромережі, включаючи нейромережі з запізненням, нелінійні і авторегресійні (NARX);
- підтримка Simulink для моделювання нейромережі, створення блоків на основі розроблених нейромережевих структур для адаптивних систем управління;
- модульне представлення мережі, що дозволяє створювати необмежене число шарів і міжмережевих зв'язків;
- візуалізація топології нейронної мережі.

Розглянемо приклад конструювання нейронної мережі в пакеті Matlab. Хай є 15 незалежних змінних - показників діяльності фірми і одна залежна змінна - об'єм продажів. Маємо базу даних за минулий рік. Необхідно побудувати потижневий прогноз об'ємів продажів на місяць. Для вирішення завдання пропонується використовувати тришарову мережу зворотного розповсюдження.

Сформуємо таку мережу, яка включає 15 нейронів у вхідному шарі (по кількості вхідних змінних), 8 нейронів в другому шарі і 1 нейрон у вихідному шарі (по кількості вихідних змінних). Для кожного шару виберемо передавальну функцію: перший шар - logsig, другий - logsig, третій - purelin. У середовищі MatLab синтаксис такої нейронної мережі виглядає таким чином:

Net=netff (PR, [S1, S2, : , Sn], {TF1, TF2 : , TFn}, btf, blf, pf),

де PR - масив мінімальних і максимальних значень для R векторів входу; Si - кількість нейронів в i-му шарі; TFi - функція активації шару i; btf - навчальна функція, що реалізовує метод зворотного розповсюдження; blf - функція настройки, що реалізовує метод зворотного розповсюдження; pf - критерій якості навчання.

Активаційною функцією може виступати будь-яка функція, що диференціюється, наприклад, tansig, logsig, purelin.

Net=netff (minmax (P), [n, m, l], { logsig, logsig, purelin }, trainpr),

де P - множина вхідних векторів; n - кількість входів нейромережі; m - кількість нейронів в прихованому шарі; l - кількість виходів нейромережі.

Необхідно також встановити метод розрахунку значення помилки. Наприклад, якщо вибраний метод найменших квадратів, то ця функція виглядатиме так: Net.performFcn='SSE'. Для встановлення максимальної кількості епох рівної 10000 скористаємося функцією: net.train Param.epochs=10000. Запустити процес навчання можна таким чином:

[net, tr]=train (net, P, T).

Після закінчення навчання мережі її можна зберегти у файлі, наприклад, з ім'ям pvz.mat. Для цього необхідно виконати команду: save pvz net.

Таким чином, в пакеті можливе конструювання мережі будь-якої складності і немає необхідності прив'язуватися до обмежень, що наклада.ться нейросимуляторами. Проте для роботи з нейронними мережами в пакеті MatLab необхідно вивчити як саме середовище, так і більшість функцій Neural Network Toolbox.

Розглянемо інший більш складніший приклад застосування нейронних мереж в пакеті MathLab.

Прогноз фінансових результатів - необхідний елемент будь-якої інвестиційної діяльності. Сама ідея інвестицій - вкладення грошей зараз з метою отримання доходу в майбутньому - грунтується на ідеї прогнозування майбутнього. Відповідно, прогноз фінансових результатів лежить в основі діяльності всієї індустрії інвестицій - всіх бірж і небіржових систем торгівлі цінними паперами. У останнє десятиліття спостерігалося стійке зростання популярності технічного аналізу - набору емпіричних правил, заснованих на різного роду індикаторах поведінки ринку. Технічний аналіз зосереджується на індивідуальній поведінці даного фінансового інструменту, поза його зв'язком з рештою цінних паперів. Але технічний аналіз дуже суб'єктивний і погано працює на правому краю графіка - саме там, де потрібно прогнозувати напрям ціни. Тому все більшої популярності набуває нейромережевий аналіз, оскільки на відміну від технічного, не припускає ніяких обмежень на характер вхідної інформації. Це можуть бути як індикатори даного тимчасового ряду, так і відомості про поведінку інших ринкових інструментів. Недаремно нейромережі активно використовують саме інституційні інвестори (наприклад, крупні пенсійні фонди), що працюють з великими портфелями, для яких особливо важливі кореляції між різними ринками.

Як відомо, для хорошого прогнозу потрібно користуватися по-перше, дуже якісно підготовленими даними, а по-друге, нейропакетами з великою функціональністю. Дані можна отримати стандартними засобами MetaTrader (рис. 3.31). Для отримання прогнозу на основі нейронної мережі застосуємо пакет MathLab. З командного рядка по команді *anfisedit* запускаємо пакет ANFIS. Редактор складається з чотирьох панелей - для даних (Load data), для генерації мережі (Generate FIS), для тренування (Train FIS) і для її тестування (Test FIS). Верхня панель призначена для проглядання структури отриманої нейромережі (ANFIS Info).

Для початку роботи завантажуємо дані, підготовлені на попередніх етапах. Для цього натискаємо кнопку Load Data і указуємо файл з даними вибірки. Після цього створюємо нейромережу натисненням кнопки Generate EIS (рис. 3.32).

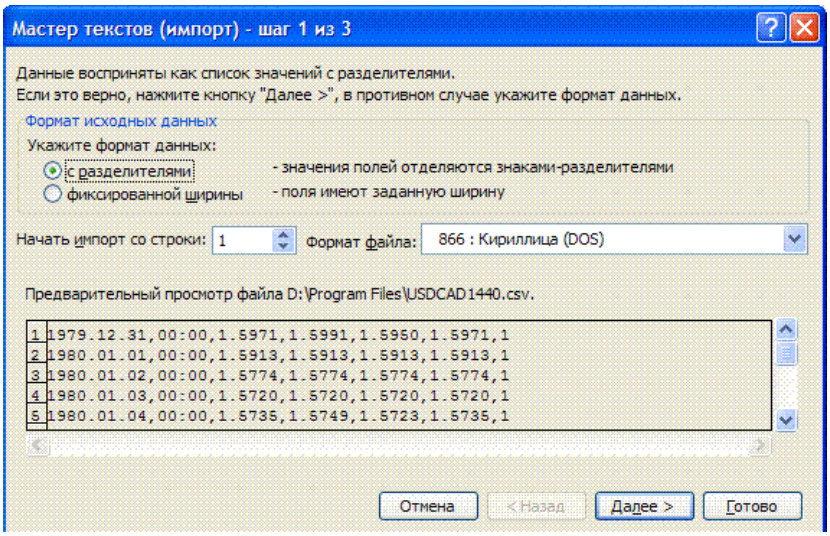


Рис. 3.31. Підготовка даних засобами пакету MetaTrader.

Для кожної з вхідних змінних задамо по 3 лінгвістичних змінних з трикутною функцією приналежності. В якості функції приналежності вихідної функції задамо лінійну функцію.

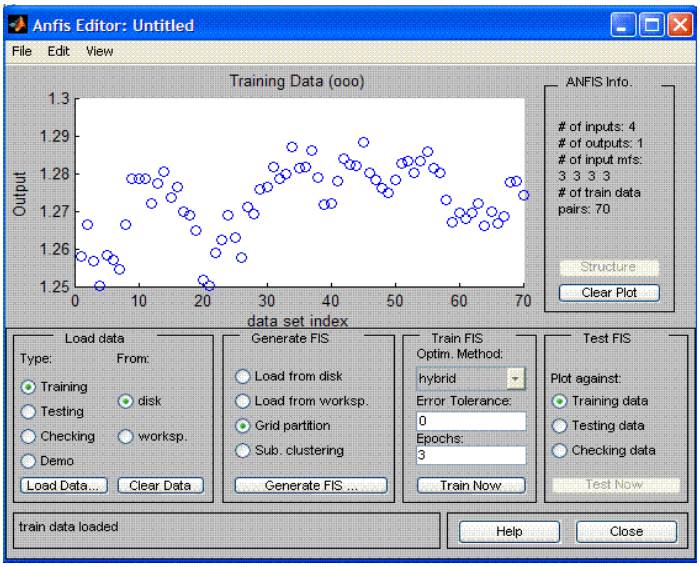


Рис. 3.32. Застосування пакету MathLab.

Для навчання нейромереж в пакеті передбачено 2 алгоритми навчання - зворотного розповсюдження і гібридний. При гібридному способі навчання мережа навчається буквально за пару-трійку проходів. На тренувальній вибірці (60 значень) після навчання прогноз мережі відрізняється від реального на декілька пунктів (рис. 3.33).

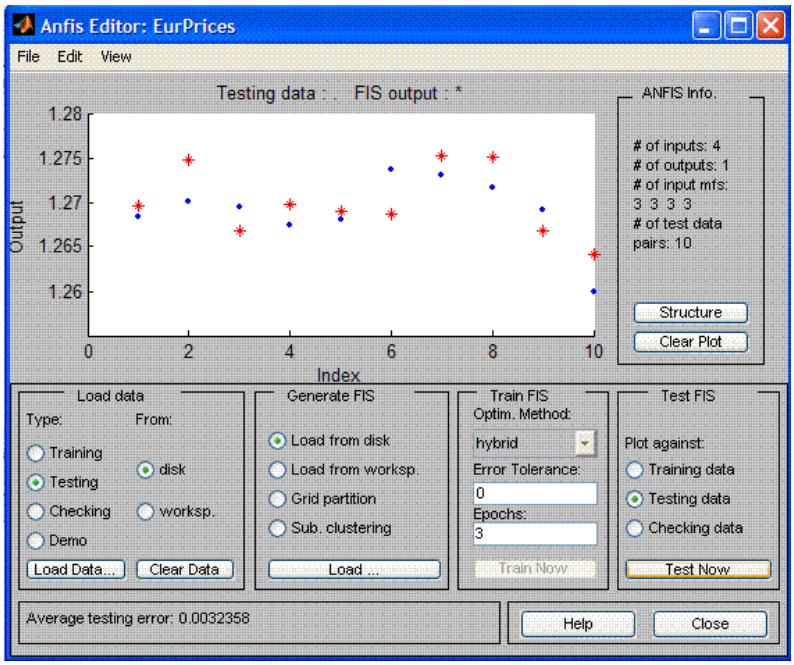


Рис.3.33. Прогнозування фінансових результатів.

Результатом роботи є багатошарова гібридна нейронна мережа, яка здатна прогнозувати абсолютні значення цін на невелике майбутнє. Отриману нейромережу можна побачити при натисненні кнопки Structure (рис. 3.34).

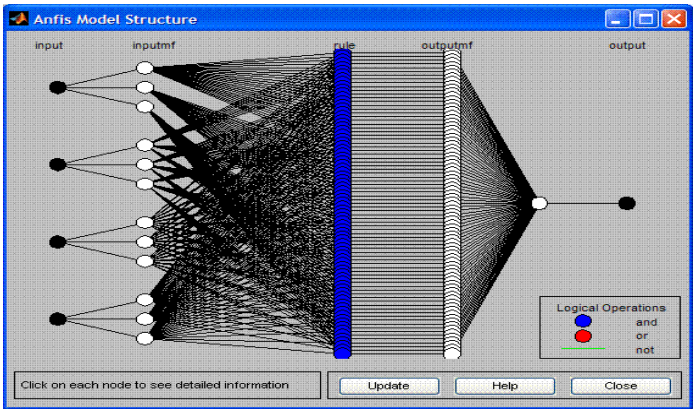


Рис. 3.34. Архітектура нейтронної мережі.

Пакет STATISTICA Neural Networks. Ця система є могутнім і надзвичайно швидким середовищем аналізу нейросетевих моделей, що надає наступні можливості:

пре- і пост-процесування, включаючи вибір даних, кодування номінальних значень, шкалірованіє, нормалізація, видалення пропущених даних з інтерпретацією для класифікації, регресії і завдання тимчасових рядів;

могутні методи розвідувальних і аналітичних технологій, зокрема *Аналіз головних компонент* і *Пониження розмірності* для вибору потрібних вхідних змінних в розвідувальному (нейромережевому) аналізі даних;

найсучасніші, оптимізовані і могутні алгоритми навчання мережі, повний контроль над всіма параметрами, що впливають на якість мережі, такими як функції активації і помилок, складність мережі;

підтримка комбінацій нейромереж практично необмеженого розміру, створених в *Наборах мереж* - *Network Sets*, вибіркове навчання нейромережевих сегментів; об'єднання, і збереження наборів мереж в окремих файлах;

повна інтеграція з системою STATISTICA, всі результати, графіки, звіти можуть бути надалі модифіковані за допомогою могутніх графічних і аналітичних інструментів STATISTICA (рис. 3.35).

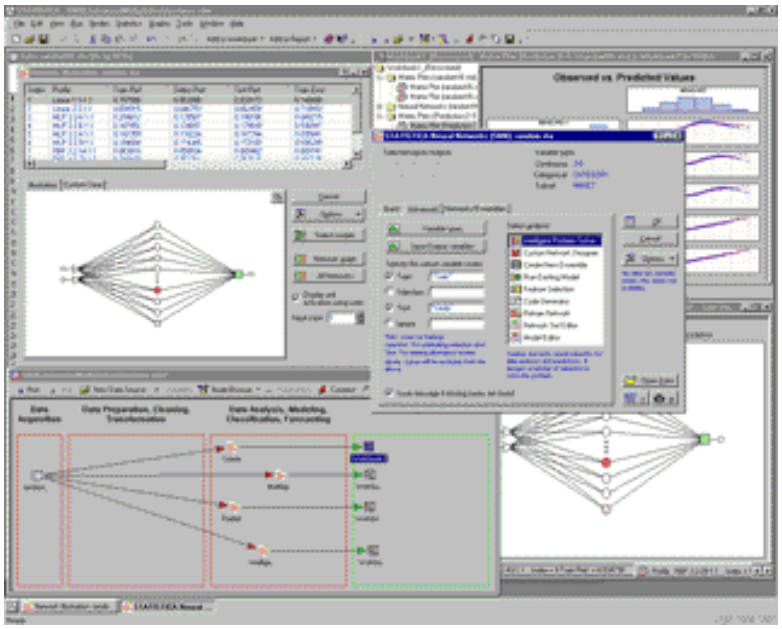


Рис. 3.35. Вигляд меню пакету STATISTICA Neural Networks.

В пакеті STATISTICA Neural Networks реалізовані зворотний і прямий алгоритми покрокового вибору. Крім того, нейро-генетичний алгоритм відбору вхідних даних поєднує в собі можливості генетичних і PNN/GRNN алгоритмів (PNN - імовірнісні нейронні мережі, GRNN - узагальнено-регресійні нейронні мережі) для автоматичного пошуку оптимальних комбінацій вхідних змінних, у тому числі і в тих випадках, коли між ними є кореляції і нелінійні залежності.

Перед тим, як дані будуть введені в мережу, вони повинні бути певним чином підготовлені. Так же важливо, щоб вихідні дані можна було правильно інтерпретувати. У пакеті є можливість автоматичного масштабування вхідних і вихідних даних, а також можуть бути автоматично перекодовані змінні з номінальними значеннями (наприклад, Стать ={ Чоловіча, Жіноча}). *STATISTICA Neural Networks* містить також засоби роботи з пропущеними даними. Реалізовані такі функції нормування, як «одинична сума», «переможець отримує все» і «вектор одиничної довжини». Є засоби підготовки і інтерпретації даних, спеціально призначені для аналізу часових рядів. У задачах класифікації є можливість встановити довірчі інтервали, які система використовує потім для віднесення спостережень до того або іншого класу. У поєднанні із спеціальною реалізованою в STATISTICA Neural Networks функцією активації Софтмакс і кросс-ентропійними функціями помилок це дає принциповий теоретико-імовірнісний підхід до задач класифікації (рис. 3.36).

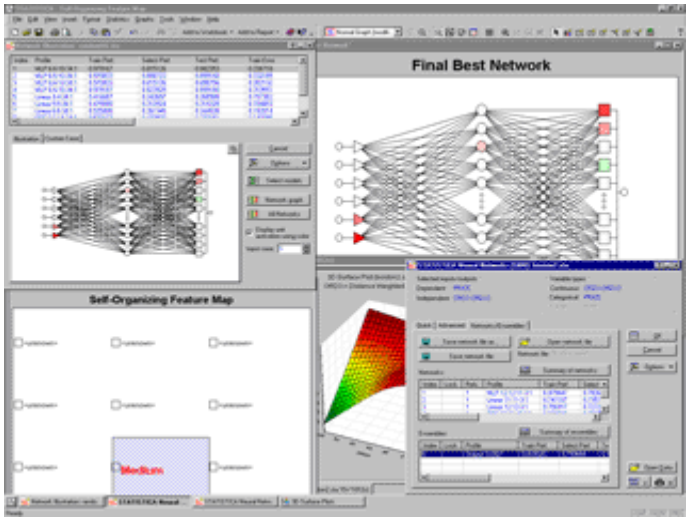


Рис.3.36. STATISTICA Neural Networks в задачах класифікації.

Різноманіття моделей нейронних мереж і безліч параметрів, які необхідно встановити (розміри мережі, параметри алгоритму навчання і т.д.), може поставити іншого користувача в безвихідь. Саме для цього в пакеті існує *Майстер рішень*, який може автоматично провести пошук відповідної архітектури мережі будь-якої складності. У системі STATISTICA Neural Networks реалізовані всі основні типи нейронних мереж, що використовуються при рішенні практичних задач, зокрема: багатошарові персептрони, мережі на радіальних базисних функціях, карти Кохонена, імовірнісні нейронні мережі; мережі для кластеризації, лінійні мережі і т. д. (рис. 3.37).

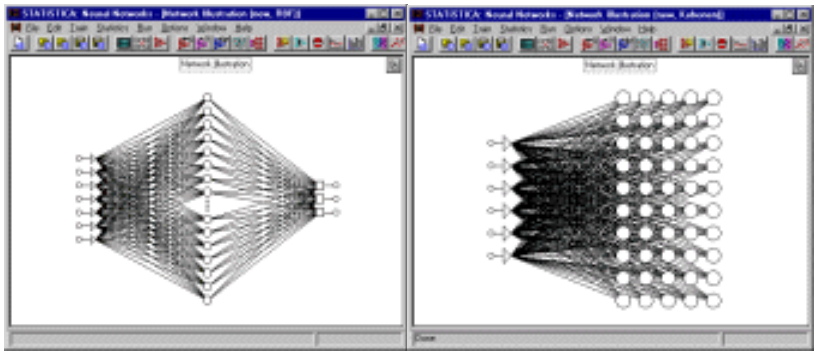


Рис. 3.37. Типи нейромереж в пакеті STATISTICA Neural Networks.

Також, в системі реалізовані мережеві ансамблі, що формуються з випадкових (але значущих) комбінацій вищеперелічених мереж. Існує ще один зручний засіб: ви можете зв'язати мережі, щоб вони запускалися послідовно. Це корисно при препроцесуванні для знаходження рішень з мінімальною вартістю (рис. 3.38).

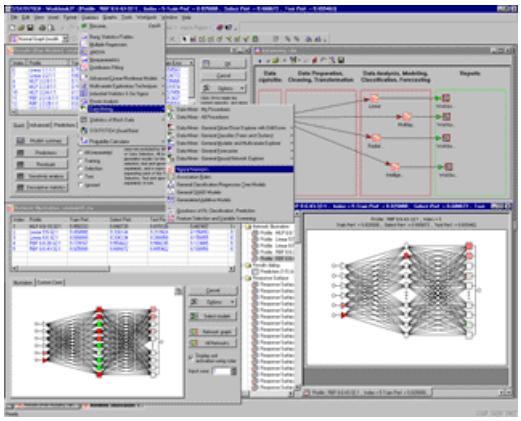


Рис. 3.38. Мережеві ансамблі.

У пакеті STATISTICA Neural Networks є численні засоби, що полегшують користувачеві вибір відповідної архітектури мережі. Статистичний і графічний інструментарій системи включає гістограми, матриці і графіки помилок для всієї сукупності і за окремими спостереженнями, підсумкові дані про вірну або невірну класифікацію, а всі важливі статистики - наприклад, пояснена частка дисперсії - обчислюються автоматично. Для візуалізації даних в пакеті реалізовані діаграми розсіяння і тривимірні поверхні відгуку, що допомагають користувачеві зрозуміти «поведінку» мережі (рис. 3.39).

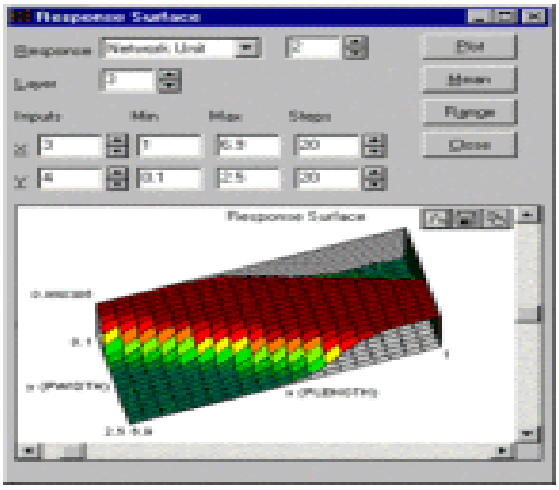


Рис.3.39. Візуалізація даних в пакеті STATISTICA Neural Networks

Для навчання багатошарових персептронів в системі STATISTICA Neural Networks реалізований, перш за все, метод зворотного розповсюдження - із змінною в часі швидкістю навчання і коефіцієнтом інерції, перемішуванням спостережень перед черговим кроком алгоритму і додаванням адитивного шуму для робастного узагальнення. Крім цього, в системі реалізовано два швидкі алгоритми другого порядку - методи зв'язаних градієнтів і Левенберга-Маркара. Останній є незвичайно могутнім сучасним алгоритмом нелінійної оптимізації. Ітеративний процес навчання супроводжується автоматичним відображенням поточної помилки навчання і обчислюваної незалежно від неї помилки на перевірочній множині, при цьому показується і графік сумарної помилки. Якщо перенавчання має місце, це не повинно турбувати користувача:

пакет автоматично запам'ятовує екземпляр якнайкращої мережі, отриманої в процесі навчання, і до цього варіанту мережі завжди можна звернутися, натиснувши відповідну кнопку. Після того, як навчання мережі завершене, Ви можете перевірити якість її роботи на окремому тестовому множині (рис. 3.40).

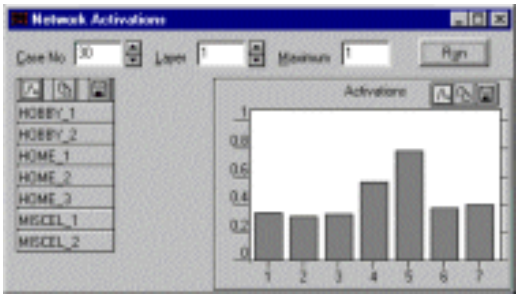


Рис. 3.40. Ітеративний процес навчання.

У системі STATISTICA Neural Networks є інтелектуальні засоби, що дозволяють відрізувати шматки від вже наявних мереж і сполучати декілька мереж воєдино. Так, можна видаляти або додавати окремі нейрони, видаляти з мережі цілком деякий шар, а мережі, узгоджені по числу входів/виходів, послідовно сполучати одна з одною. Завдяки цим можливостям пакет дозволяє використовувати такі засоби, як пониження розмірності за допомогою асоціативних мереж і матриця втрат (для ухвалення рішень з найменшими втратами). Матриця втрат автоматично використовується при роботі з імовірнісними нейронними мережами (рис. 3.41).

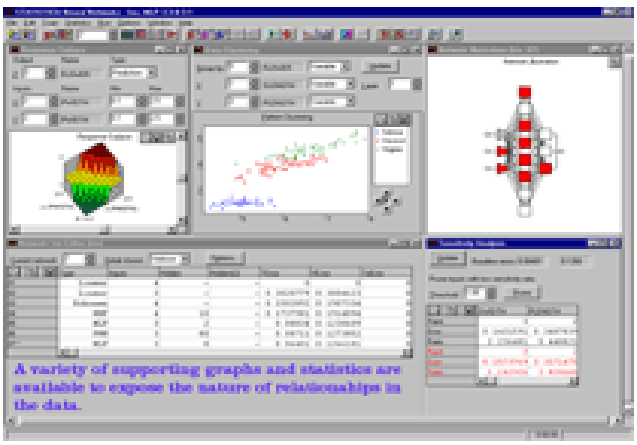


Рис. 3.41. Переформатування нейронних мереж.

Нейропакет NeuroSolutions. Програмний комплекс призначений для моделювання великого набору нейронних мереж. Основне його достоінство полягає в гнучкості: крім традиційних нейросетевих парадигм (повнозв'язних і багатошарових нейронних мереж, карт Кохонена, що самоорганізовуються) нейропакет включає могутній редактор візуального проектування нейронних мереж, що дозволяє створювати будь-які нейронні структури і алгоритми їх навчання, а також вводити власні критерії навчання. Пакет має хороші засоби візуалізації структур, процесів і результатів навчання і функціонування нейронних мереж. Це ставить даний нейропакет на рівень CAD-систем (систем автоматизованого проектування) і моделювання нейронних мереж.

Крім засобів взаємодії з операційною системою (OLE), нейропакет забезпечений генератором початкового коду і дозволяє використовувати зовнішні модулі при створенні і навчанні нейронної мережі. Пакет підтримує програми, написані на мові C++ для компіляторів Microsoft Visual C++ і Borland C++, а також у вигляді DLL-коду. Таким чином, NeuroSolutions є гнучкою відкритою системою, яку можна при необхідності доповнювати і модифікувати. Пакет містить вбудовану макромову, що дозволяє проводити практично будь-яку настройку під конкретне завдання.

У пакеті реалізується великий перелік нейронів, включаючи зважений суматор (нейрон першого порядку), нейрони вищих порядків (з перемножуванням входів), а також безперервний інтегруючий нейрон. Функція активації нейрона може бути вибрана з п'яти стандартних функцій, а також задана користувачем. Зв'язки між нейронами задаються довільно на етапі проектування і можуть бути змінені в процесі роботи. Підтримуються всі типи зв'язків: прямі, перехресні і зворотні. При цьому добре реалізована схема організації зв'язків: можна задати один векторний зв'язок із заданою ваговою матрицею, а не набір скалярних зв'язків з ваговими коефіцієнтами.

Нейропакет містить могутні засоби для організації навчальних вибірок. Вбудовані конвертори даних підтримують графічні зображення, текстові файли з числовими або символьними даними, а також функції безперервного

аргументу (наприклад, часу), задані в аналітичному вигляді або у вигляді вибірки значень. Нейропакет дозволяє використовувати будь-які зовнішні конвертори даних.

На етапі навчання може бути використаний широкий круг критеріїв навчання, як дискретних, так і безперервних. Крім цього можна вводити власні критерії. Також можливо використовувати як вбудований алгоритм навчання типу back-propagation або дельта-правіла, так і використовувати власний. Система візуалізації процесу навчання дозволяє проводити аналіз зміни вагів безпосередньо в процесі навчання і вносити корективи. Пакет містить генератор (майстер) стандартних нейромережевих архітектор (Neural Wizard), за допомогою якого швидко задається архітектура, підбирається навчальна вибірка, критерії і методи навчання нейронної мережі.

Нейропакет NeuralWorks Professional II/Plus. NeuralWorks Professional є могутнім засобом для моделювання нейронних мереж. У ньому реалізовано 28 нейронних парадигм, а також велику кількість алгоритмів навчання. Додатковий модуль UDND (User Define Neural Dynamics) дозволяє створювати власні нейронні структури. Як і NeuroSolutions, NeuralWorks Professional має хорошу систему візуалізації даних: структури нейронної мережі, зміни помилки навчання, зміни вагів і їх кореляції в процесі навчання. Останнє є унікальною властивістю пакету і корисна при аналізі поведінки мережі. У NeuralWorks Professional можна інтегрувати зовнішні програмні модулі. Він має вбудований генератор коду, що підтримує компілятор Microsoft Visual C++.

Нейропакет Brain Maker Pro. Одним з поширених зарубіжних нейросистем є пакет Brain Maker. Хай необхідно вирішити задачу прогнозу ціни закриття на сьогоднішніх торгах по валютних тисячедоларових тримісячних ф'ючерсних контрактах. Хай нас влаштовує точність прогнозу, при якій правильно вказується ціновий тренд і зміна ціни з точністю не нижче 90% від останнього стрибка. Застосування нейронної мережі починається з підготовки вхідних даних: курс долара, індекс інфляції, ставка межбанка,

біржові індекси, об'єм торгів, кількість операцій, максимальні і мінімальні ціни і ін. Після попередньої настройки мережі починається ітераційний процес навчання, в результаті якої нейромережа настроює свою логічну структуру для точної реакції ринку на ті або інші дії. Для цього в пакеті Brain Maker передбачений могутній аналітичний блок, який дозволяє побачити, які параметри роблять позитивний вплив на ситуацію, а які - негативний. Потім мережа знову навчається і далі тестується на якість і адекватність, і після вдалого тестування використовується для прогнозів. За десять біржових днів мережа жодного разу не помилилася в знаку відхилення ф'ючерсних котирувань, а дев'ять днів з десяти відхилення прогнозу від реальної ціни склало менше 10 доларів. Brain Maker - це програма, з якою почалася історія застосування нейронних мереж в Росії та Україні. У цьому пакеті на професійному рівні реалізована класична багатошарова нейронна мережа. Це єдина програма, в якій є можливість настройки всіх параметрів нейронних мереж і алгоритмів навчання. Останнім часом Brain Maker найчастіше використовується не як самостійна програма, а як надбудова до програми TradeStation для аналізу в режимі реального часу.

Нейропакет NeuroShell 2. Пакет є однією з трьох програм, що входять до складу пакету The Al Trilogy і є універсальним нейропакетом для моделювання декількох найбільш відомих нейронних парадигм: багатошарових мереж, мереж Кохонена і т.д. З його допомогою можна вирішувати широкий спектр завдань, починаючи з широко поширених задач прогнозування курсів акцій і закінчуючи менш поширеними зворотніми задачами геофізиці. Він включає систему для початківця, систему для професіонала, засоби автономного використання. Крім того, існують пакети доповнень до NeuroShell 2, які дозволяють істотно спрощувати рішення окремих задач. Зокрема:

Пакет ринкових індикаторів з оптимізацією містить близько 150 найбільш відомих і могутніх технічних індикаторів, які можна застосовувати для фінансових даних або інших часових рядів.

Прогноз результатів скачок. Виконує передобробку статистичних даних про результати скачок для того, щоб зробити прогнози точнішими. Програма бере дані про результати всіх коней в одному забігу і перетворить їх у файл з великою кількістю рядів, в кожному з яких приведені результати тільки для 2 коней. При такій формі представлення даних точність прогнозів нейромережі значно підвищується. Це доповнення застосовне не тільки для прогнозу результатів скачок, але і для вирішення інших задач, пов'язаних з ранжируванням даних.

Прогнозування результатів виборів (ранжирування). Дозволяє успішно застосовувати нейронні мережі для прогнозування результатів виборів за наслідками ряду передуючих їм регулярних соціологічних опитів. Вхідні дані можуть включати дані про результати опитів, однакові для всіх кандидатів, а також дані для кожного кандидата. Кожен тренувальний приклад може включати дані одного опиту для декількох кандидатів, що беруть участь у виборах. Кількість кандидатів в різних опитах може бути разним. Це доповнення здійснює передобробку даних так, щоб мережа могла порівнювати за один раз тільки по двох кандидатів. Це спрощує тренування мережі і збільшує точність прогнозів.

Складовою частиною нейропакету є програмний комплекс **NeuroShell Trader** - система, призначена для прогнозування і пошуку ефективних торгових стратегій на фінансових ринках. NeuroShell Trader - це сімейство продуктів, розроблене спеціально для трейдерів і покликане допомогти їм в ухваленні рішень при торгівлі. В них реалізовані технології штучного інтелекту, що дозволяють прогнозувати фінансові часові ряди, будувати і оптимізувати торгові стратегії. Пакет спочатку розроблявся як інструмент для нейромережевого аналізу біржових даних, тому побудова в ньому прогнозів і торгових стратегій за допомогою нейронних мереж і генетичних алгоритмів проста і зрозуміло навіть для користувача, що не є професіоналом в цій області. Крім того, будучи спеціалізованим інструментом для трейдерів, NeuroShell Trader має дружній графічний інтерфейс, багаті можливості для імпорту даних і

могутню бібліотеку індикаторів. Він дозволяє відображати біржові дані на робочих листах у вигляді графіків, діаграм, японських свічок, дозволяє використовувати більше 800 вбудованих індикаторів і створювати свої індикатори, а також одночасно працювати з декількома фінансовими інструментами. Нейронна мережа або індикатор, побудовані для одного фінансового інструменту, автоматично застосовуються до всіх інструментів, які вказуються. NeuroShell Trader дозволяє користувачеві отримати важливу статистику та іншу інформацію, що стосується застосування торгових стратегій і нейронних мереж до даних (рис. 3.42).

Рис. 3.42. Нейромережевий аналіз біржових даних.

NeuroShell Series. Нейромережева архітектура, яка лежить в основі програм даної серії, є найостаннішими досягненнями наукового пошуку, результатом якого з'явилося створення алгоритму «самопобудови» нейронної мережі, що володіє рекордними швидкостями навчання. Ці програми надзвичайно прості у використанні. Тепер користувач повинен зосередитися тільки на формулюванні завдання, все інше програми даної серії зроблять самі. До складу серії входять:

- **NeuroShell Predictor** – Провісник.
- **NeuroShell Classifier** – Класифікатор.

- **NeuroShell Run-time Server** - Засоби автономного використання мереж, отриманих в NeuroShell Predictor і NeuroShell Classifier.

Пакет NeuroShell Predictor дає можливість створювати системи для вирішення задач прогнозування на основі наявної бази даних. Це можуть бути прогнози наступних значень параметрів часового ряду, наприклад, прогноз курсу акцій, або оцінка якої-небудь величини, визначуваної набором незалежних чинників, наприклад, оцінка вартості квартир або вживаних автомобілів (рис. 3.43).

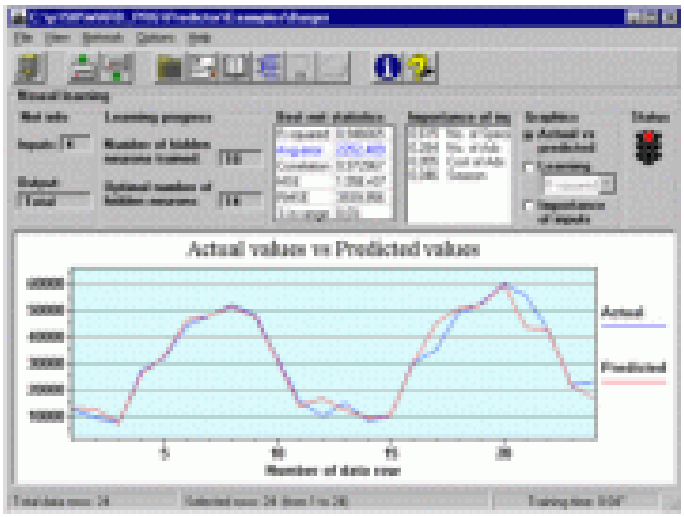


Рис. 3.43. Застосування пакету NeuroShell Predictor.

NeuroShell Predictor настільки простий у використанні, що відпадає необхідність в керівництві користувача. Замість нього в програмі є «Інструктор», який проведе через всі етапи створення моделі для прогнозів. Інструктор дає можливість вибрати стовпці у файлі даних, які використовуватимуться як входи, і вказати один стовпець як вихід.

Пакет NeuroShell Classifier призначений для вирішення задач розпізнавання образів, пов'язаних з визначенням приналежності образу (ситуації) до тієї або іншої категорії. Наприклад, по набору біржових показників виробляти сигнал для покупки або продажу акцій тієї або іншої компанії. Пакет був розроблений, подібно NeuroShell Predictor, для тих, хто не мав попереднього досвіду роботи з нейронними мережами. На відміну від

програми NeuroShell Predictor, яка на виході нейронної мережі дає безперервне значення величини, що передбачається, нейронна мережа NeuroShell Classifier має декілька виходів, які визначають вірогідність приналежності пред'явленого образу до кожної з декількох категорій. Як приклади категорій можна привести такі, як { купити, продати, утриматися від операцій} або { ракова пухлина, доброякісна пухлина}. У NeuroShell Classifier реалізовано два ефективні алгоритми класифікації : один є спеціальною нейронною мережею, а інший - статистичний класифікатор, керований генетичним алгоритмом. Дані алгоритми були розроблені спеціально для того, щоб зробити моделі класифікації точнішими, а також щоб «не сушити голову» кожного разу при створенні моделі, підбираючи її параметри. (рис. 3.44).

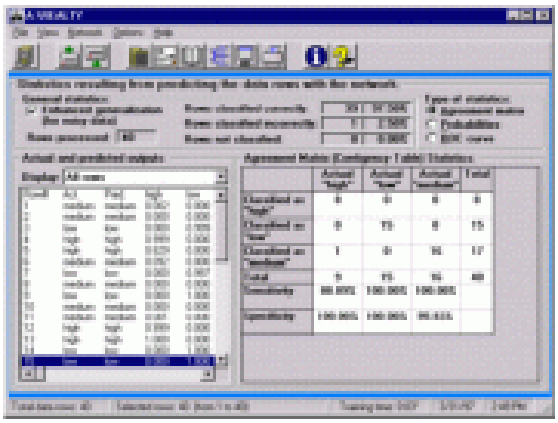


Рис. 3.44. Задача класифікації в пакеті NeuroShell Classifier.

Пакет NeuroShell Run-time Server містить ряд програм, які дозволяють використовувати мережі, створені за допомогою NeuroShell Predictor і NeuroShell Classifier, або з робочих листів Microsoft Excel, або у власних програмах.

Програмний комплекс DEDUCTOR. Deductor Studio є аналітичним ядром платформи Deductor. Він містить повний набір механізмів імпорту, обробки, візуалізації і експорту даних для швидкого і ефективного аналізу інформації. В пакеті використовуються наймогутніші технології, такі як

багатовимірний аналіз, нейронні мережі, дерева рішень, карти Кохонена, спектральний аналіз і множина інших. При цьому акцент зроблений на самонавчальні методи і машинне навчання, що дозволяє будувати адаптивні системи, тобто здатні реагувати на зміну ситуації. Використання самонавчальних методів і майстрів для настройки, дозволяє понизити вимогу до підготовки персоналу, роблячи сучасні технології доступними широкому кругу користувачів. Всі механізми уніфіковані і виконуються за допомогою майстрів (рис. 3.45).

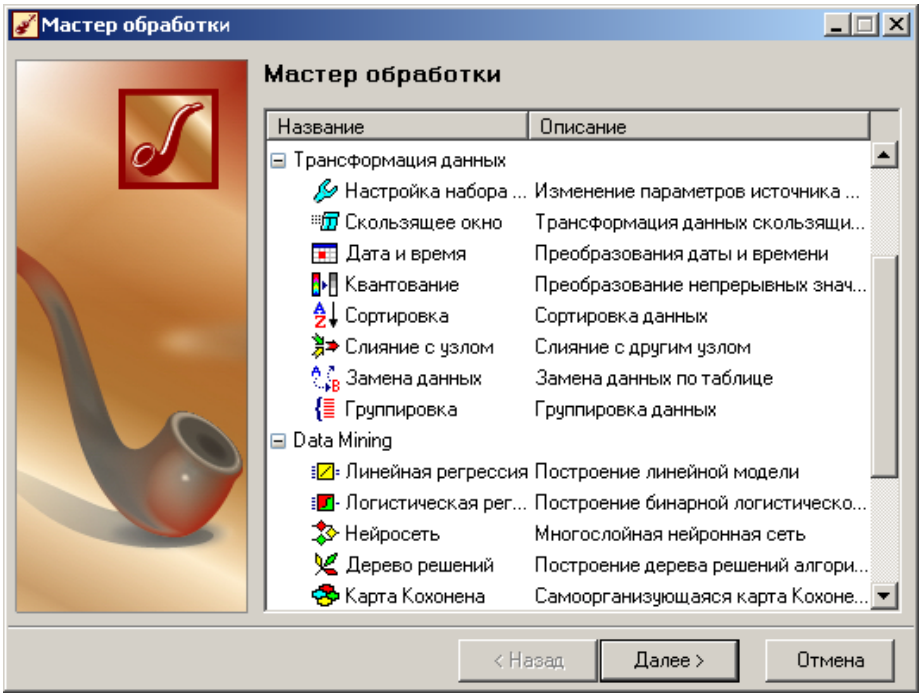


Рис. 3.45. Аналітичні технології платформи Deductor.

Нейронні мережі в пакеті підтримуються за допомогою бібліотеки компонентів NeuralBase. Як приклад, створені компоненти, що реалізовують дві нейромережеві парадигми: рекурентну нейронну мережу (мережу Хопфілда) і багатошарову нейронну мережу навчену по алгоритму зворотного розповсюдження помилки. Основним призначенням бібліотеки є інтеграція нейронних мереж в інформаційні системи, для розширення аналітичних можливостей систем. Реалізація нейронних мереж у вигляді компонентів, наявність відкритого коду дозволяє легко вбудовувати їх в інші програми.

Існує три базові класи TNeuron, TLayer, TNeuralNet. Всі інші є похідними від них. TNeuron є базовим класом для нейронів, несе всю основну функціональність, має індексовану властивість Weights, що є ваговими коефіцієнтами (синапсами), властивістю Output, яка є виходом нейрона (результатом обчислень) і суматор, роль якого, виконує метод ComputeOut. TNeuronHopf, нащадок TNeuron, реалізує нейрон використовуваний в нейронній мережі Хопфілда, єдиною відмінністю від базового класу, є використання активаційної функції в перекритому методі ComputeOut. Наступним породженим класом, є TneuronBP, яка служить для програмної реалізації багатошарових нейронних мереж. Основним призначенням базового класу TLayer і його нащадків TLayerHopf і TLayerBP є об'єднання нейронів в шар, для спрощення роботи з нейронами. Компонент TNeuralNetHopf реалізує нейронну мережу Хопфілда.

Система 4Thought. Даний продукт входить в сімейство засобів Business Intelligence, пропонованих одним з провідних розробників інструментальних засобів для створення аналітичних додатків в бізнесі, канадсько-американською компанією Cognos. Засоби аналізу результатів, передбачені в даній системі роблять її зручним інструментом як для кінцевого користувача-аналітіка, так і для фахівця-математика. Інтеграція із засобами OLAP (система PowerPlay) дозволяє застосовувати нейромережеві методи аналізу і прогнозування безпосередньо до гіперкубів PowerPlay.

Використовуючи технологію нейронних мереж, 4Thought дозволяє аналізувати ефективність роботи підприємств і їх підрозділів, віддачу від інвестицій або різних видів реклами і т.д. Параметр, що характеризує ефективність, - це цільовий показник, а всі інші, що впливають на нього ми називатимемо чинниками (причому значення показника і чинників можуть бути як числовими, так і символьними). Початкові дані аналізу представляються у вигляді таблиці, один із стовпців якої - цільовий показник (наприклад, прибуток або об'єм продажів), а інші - впливаючі чинники (витрати на рекламу, пора року, регіон і ін.). Пакет будує кількісну модель залежності значень

показника від значень чинників, після чого дозволяє проводити перехресний аналіз (проглядати в графічній і аналітичній формі залежність модельованого показника від будь-якого з вибраних чинників при фіксованих або усереднених значеннях інших чинників), перевіряти гіпотези «що якщо», оцінювати значущість чинників за мірою їх впливу на цільовий показник, а також використовувати отриману модель для прогнозування значення показника виходячи з відомих значень чинників. Розглянемо декілька прикладів, в основі яких лежать реальні додатки.

Аналіз роботи філіалів. В даному прикладі досліджувався вплив різних чинників (регіон, масштаб і види діяльності) на ефективність роботи філіалу компанії. До регіональних чинників можна віднести категорію регіону і тип населеного пункту, де розташований торговий філіал: міський або сільський. Масштаб діяльності може характеризуватися величиною обороту і потоками наявних засобів, а види діяльності - видами товарів. Як цільовий показник можна вибрати величину невиробничих витрат філіалу. Для ефективного управління компанією необхідно розуміти, від чого залежать витрати в кожному філіалі та що є вирішальним чинником. На підставі таблиці, що характеризують дані по філіалах, 4Thought дозволяє будувати модель впливу вибраних стовпців-чинників на величину витрат філіалу. Це досить складне завдання, яке неможливо вирішити традиційними статистичними методами. Система використовує дані про роботу філіалів для навчання нейронної мережі, яка моделює залежність витрат філіалів від різних чинників. Готову модель можна використовувати для кількісного аналізу впливу чинників на витрати філіалів. Зокрема, побудувавши залежність модельних значень витрат від обороту (перехресний аналіз), можна оцінити результат економії на масштабах: чим більше оборот, тим менше витрачання. Модель дозволяє досліджувати навіть ті чинники, які не були в неї введені. Наприклад, якість управління філіалом не можна пов'язувати безпосередньо з його прибутковістю - на прибутковість впливають і інші чинники. Проте можна порівняти фактичні показники ефективності (наприклад, витрати) з очікуваними, які обчислюються

за допомогою побудованої моделі, а отже, в них враховується вплив регіональних і інших чинників. Якщо очікувані витрати вище фактичних, значить, ефективність роботи філіалу вище середнього, а це, швидше за все, викликано високою якістю управління (і навпаки).

Окупність капіталовкладень. Сьогодні багато компаній вкладають величезні засоби в розвиток і використання інформаційних технологій. Але як оцінити, наскільки ефективно використовуватиметься обчислювальна техніка і складне програмне забезпечення? Чи окупляться капіталовкладення в цю сферу? Яким чином вплине впровадження нової технології на доходи підприємства і на продуктивність співробітників? Відповісти на ці питання складно - термін окупності часто дуже великий, а на ефективність роботи компанії впливають і інші чинники. Як приклад візьмемо мережу магазинів роздрібної торгівлі, де встановлювалася інформаційна система для касових терміналів в 2000 точок збуту. Об'єм інвестицій склав 75 млн. дол. Передбачалося, що завдяки впровадженню нової системи вдасться підвищити рівень обслуговування клієнтів. Проте після вкладення чверті грошових коштів виникли сумніви в окупності проекту. Задача оцінки окупності інвестицій виявилася складною. Вплив саме цих капіталовкладень на зростання валового прибутку в кожному окремому підрозділі необхідно було відокремити від впливу інших чинників: кваліфікації менеджерів, розширення філіалу, недавніх витрат на ремонт і т.д. По всіх торгових точках були зібрані дані про перераховані чинники, а також інформація про зростання доходів і про те, скільки місяців пройшло з моменту впровадження інформаційної системи. Щоб оцінити ефективність впроваджуваної системи потрібно було побудувати модель зростання доходів залежно від комбінації даних чинників. За допомогою системи 4Thought був проведений аналіз взаємозалежності з метою отримання кількісної оцінки дії кожного чинника окремо (рис. 3.46).

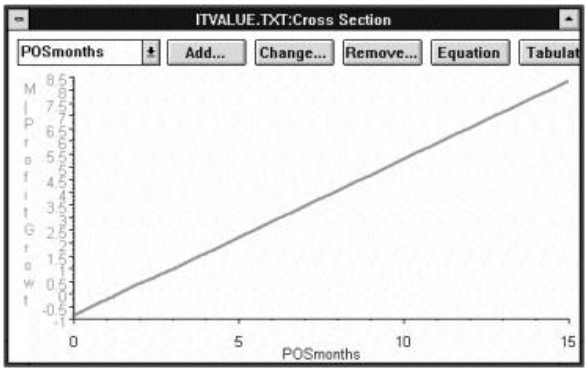


Рис. 3.46. Залежність прибутку компанії від терміну впровадження.

Побудований системою графік показує, що чим довше працює інформаційна система (горизонтальна вісь), тим більше зростання прибутку (вертикальна вісь). Зокрема, видно, що інвестиції повністю окупляться протягом 12 місяців. Отримавши таку інформацію, менеджери без коливань вирішили продовжити вкладення тих, що залишилися 75% коштів.

Оптимізація цін. Один з найкорисніших додатків нейроаналітіки - оптимізація цін. Очевидно, чим більше ціна на товар, тим вище прибуток від його продажу. Проте у міру підвищення ціни об'єм продажів починає знижуватися, і, може так трапитися, що товар перестануть купувати зовсім. Але якщо ціна дуже низька, то отримати прибуток від збуту не вдасться. Необхідно встановити деяке проміжне, оптимальне значення. У системі 4Thought для вирішення цього завдання шляхом аналізу даних про продажі за минулий період з'ясовується, як міняються витрати і об'єми продажів залежно від ціни на товар. Величина оптимальної ціни залежить від реакції клієнтів на зміну цін, від собівартості об'єму продукції, а також від зовнішніх чинників.

Компанія, що займається ремонтом автомобілів, використовувала 4Thought для виставляння оптимальних цін на свої послуги. Щоб визначити чутливість клієнтів до зміни цін, система повинна встановити, яким чином у минулому мінявся попит на продукт залежно від різних чинників. Що стосується компанії по ремонту автомобілів, то тут необхідно було промоделювати зміна попиту залежно від цін на послуги самої фірми, середнього рівня цін конкурентів, пори року, витрат на рекламу, а також певних

змін в законодавстві, які вплинули на структуру ринку. Були зібрані дані за три роки. Інформацію про ціни конкурентів вдалося отримати від менеджерів філіалів, які добре знайомі з діяльністю інших фірм. Щоб визначити, які чинники впливають на попит, була побудована модель зміни об'єму продажів відповідно до цін на послуги, рівнем цін конкурентів, порою року і витратами на рекламу. Ця інформація використовувалася для оцінки співвідношення ціна/об'єм продажів.

На рисунку 3.47 показано, як росте попит на послуги (вертикальна вісь) при зниженні цін (горизонтальна вісь). Щоб визначити оптимальну ціну, необхідно побудувати модель витрат, що показує, яким чином міняються витрати залежно від об'єму продажів. Витрати - розрахункова величина, яку можна ввести в модель за допомогою обчислюваного стовпця. Комбінуючи моделі продажів і витрат, можна отримати повну картину впливу різних чинників на прибутковість бізнесу. Зокрема, можна встановити залежність між ціною і прибутком.

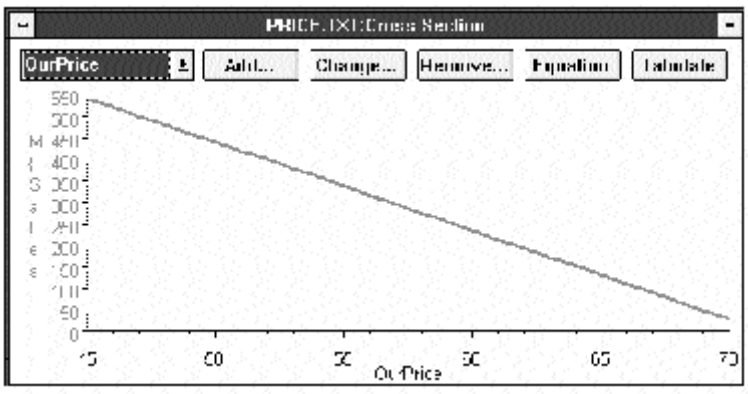


Рис. 3.47. Залежність попиту від цін на послуги.

На рисунку 3.48 показана модель зміни прибутку (вертикальна вісь) залежно від зміни цін на послуги (горизонтальна вісь). У такий спосіб вдається визначити оптимальну ціну для будь-якої іншої галузі бізнесу. Компанія по ремонту автомобілів використовувала отриману інформацію для встановлення цін на свої послуги, при цьому тільки за рахунок правильного ціноутворення їй вдалося підвищити прибуток на декілька млн. дол.

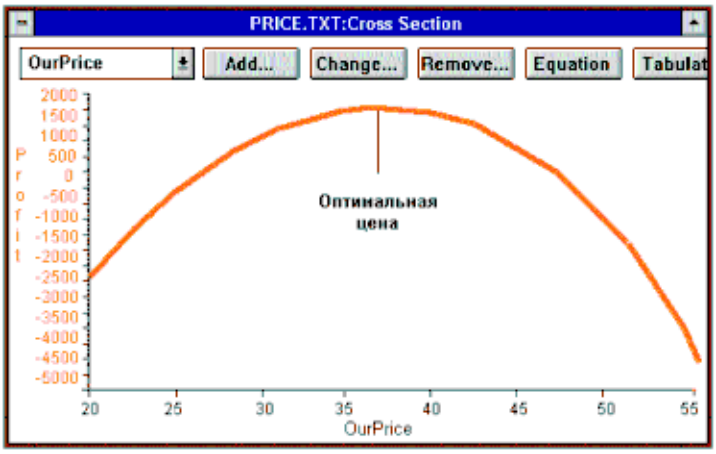


Рис. 3.48. Залежність прибутку компанії від цін на послуги.

Система SENN Sales. Даний продукт є спеціалізованим засібом для інтелектуального аналізу даних у фінансовій і комерційній сферах. Як і 4Thought, ця система також може завантажувати інформацію з корпоративних баз даних без обмежень на кількість стовпців і рядків в таблицях. SENN Sales використовується для вирішення задач профілізації клієнтів, маркетингового аналізу, прогнозування попиту і т.д. Розглянемо досвід її застосування для прогнозування фінансових рядів.

Надійне і точне прогнозування ф'ючерсних курсів обміну валют, кредитних ставок, цін і динаміки продажів дозволяє укладати вигідні міжнародні валютні угоди, формувати фінансові і інвестиційні пакети, давати точну оцінку поточної ситуації на ринку і т.д. Дані за минулі періоди (індекси, курси обміну, кредитні ставки і криві продажів) містять структурні залежності, виявивши які, можна визначити поведінку системи в майбутньому. Використовуючи метод моделювання, що забезпечує точне відтворення динаміки поведінки системи, можна описати залежності в наявних даних і побудувати прогноз.

Оптимальне управління портфелем цінних паперів. При розв’язанні даної задачі за допомогою системи SENN результати прогнозування доходів від капіталовкладень оброблялися методами ухвалення оптимальних рішень. Спочатку для визначення якнайкращого способу розміщення капіталу розглядався індекс фондової біржі, а також довгострокові і короткострокові

кредитні ставки для кожної з країн (Англія, Франція, Німеччина, Японія і США). При цьому були прийняті до уваги зміни в системі рахунків в Європі і країнах великої сімки, викликані появою на фінансових ринках нової європейської валюти. При традиційному підході до управління портфелем прогнозується дохід на кожну із статей капіталовкладень: за допомогою методів нейроаналітіки була отримана ставка доходу 21%. На основі побудованих моделей була розроблена стратегія розміщення фондів, що оптимізує дохід від капіталовкладень для портфеля активів і що оцінює можливі ризики.

3.5. Сучасна практика та перспективні напрямки застосування нейротехнологій.

Нейронні мережі все частіше застосовуються в реальних бізнес-додатках. У деяких областях, таких як виявлення фальсифікацій і оцінка ризику, вони стали безперечними лідерами серед використовуваних методів. Їх використання в системах прогнозування і системах маркетингових досліджень постійно росте. Варто відзначити, що оскільки, економічні, фінансові і соціальні системи дуже складні і є результатом дій і протидій різних процесів, то є дуже складним створити повну математичну модель з урахуванням всіх можливих дій і протидій. Практично неможливо детально апроксимувати модель, засновану на таких традиційних параметрах, як максимізація корисності або максимізація прибутку.

У системах подібної складності є природним і найбільш ефективним використовувати моделі, які безпосередньо імітують поведінку суспільства і економіки. А це якраз те, що здатна запропонувати методологія нейронних мереж. Розглянемо деякі важливі області, в яких ефективність застосування нейронних мереж доведена на практиці [9, 17, 22, 45, 70, 77].

Застосування нейронних мереж в менеджменті. Протягом останнього десятиліття в журналах і газетах, таких як «Management Science», «Man and Cybernetics», «Decision Sciences», «Computers & Operations Research», «European Journal of Operational Research» були надруковані незліченні пропозиції по застосуванню нейронних мереж в бізнесі і дослідженні операцій. Більшість варіантів застосування нейронних мереж в менеджменті стосуються задач, що потрапляють в наступні чотири категорії: класифікація, побудова емпіричної кривої і аналіз часових рядів, кластеризація і оптимізація. Нижче приведені приклади кожної категорії.

1. Класифікація. Належним чином розроблена нейронна мережа може використовуватися як класифікатор. Після навчання історичним даним нейронна мережа може визначати клас приналежності деякої характерної межі. Нейронні мережі можна використовувати при аналізі кредитоспроможності, щоб передбачити банкрутство фірми. Нейронні мережі можуть також оцінити активи і зобов'язання. У багатьох банках нейронні мережі можна використовувати для виявлення підробки кредитної картки.

2. Побудова емпіричної кривої і аналіз часових рядів. Процес навчання в багатьох типах нейронних мереж може розглядатися як побудова емпіричної кривої. Крім того, нейронні мережі можуть використовуватися для визначення моделі коливань часового ряду. Аналітики сфери маркетингу, використовуючи нейронні методи мережі, можуть визначати ринкові функції відгуку, засновані на часових даних. Керівники виробництва можуть передбачати продуктивність фірми, грунтуючись на кривих, представлених навченими нейронними мережами. Багато фінансових установ використовують нейронні мережі для фінансового прогнозу і управління інвестиціями.

3. Кластеризація. Неконтрольовані нейронні мережі, що навчаються, зазвичай використовуються в кластерному аналізі для групування об'єктів без апріорного знання класів. Ідентифікація споживчих сегментів і групування технологічних деталей можуть служити прикладом в цій прикладній категорії.

4. Оптимізація. Оскільки процес навчання в нейронних мережах повинен мінімізувати наперед певну помилку або енергію, то нейронні мережі можуть використовуватися для вирішення задач оптимізації. Задачі на зразок оптимального планування робіт, оптимального планування роботи магазина і мінімізації втрат можуть бути вирішені з використанням нейронних мереж.

Прогноз ризиків і рейтингування. Існують дві базові інвестиційні стратегії: активна, заснована на прогнозах прибутковості тих або інших активів, і пасивна, в якій ринок вважають непередбачуваними, і головною метою ставлять мінімізацію ризиків. Оцінка інвестиційного ризику, таким чином, є одним з наріжних каменів фінансового аналізу. Ці методики використовують два основні підходи: навчання з вчителем - на прикладах експертних оцінок або збанкрутілих фірм, і навчання без вчителя - шляхом категоризації наявних даних.

Спочатку розглянемо перший, прямолінійніший підхід. З цією метою систематизуємо поняття рейтингу цінних паперів.

Рейтинг корпоративних облігацій. Істотну частину ринку цінних паперів складають корпоративні облігації - позики корпорацій під фіксований відсоток. Тільки на Нью-Йоркській Фондовій біржі в 2005 році оберталися облігації близько 15000 компаній із загальною номінальною вартістю понад $260 млрд. Для оцінки ризику невиплати відсотків або неповернення грошей по облігації практично для всіх таких корпорацій існують і періодично оновлюються рейтинги, що складаються незалежними рейтинговими агентствами.

У рейтинговому бізнесі домінують декілька компаній, наприклад, Standard & Poor's і Moody's. Понад 4000 боргових емітентів поставляють свої фінансові звіти цим двом організаціям. Рейтинги цих агентств надзвичайно авторитетні, від них безпосередньо залежать процентні ставки по облігаціях: чим нижче рейтинг емітента - тим дорожче обходиться емітенту обслуговування свого боргу, оскільки інвестори бажають отримати плату за додатковий ризик. Більш того, в США деяким категоріям інвесторів, таким, як

банки і страхові компанії, законодавчо заборонено купувати облігації з рейтингом Standard & Poor's і Moody's нижче певного рівня.

Алгоритм складання описаних вище рейтингів невідомий, більш того, агентства стверджують, що він не заснований в чистому вигляді на статистичному аналізі фінансової інформації, а містить ще оцінки експертів, наприклад для таких параметрів, що важко формалізуються, як «якість менеджменту». Така ситуація цілком влаштовує самі рейтингові агентства, перетворюючи їх продукцію на унікальний товар. Проте багато інвесторів зацікавлено у володінні своїми власними алгоритмами рейтингування, що «емюлюють» рейтинги великої двійки, - принаймні по трьом причинам.

- По-перше, не для кожної облігації є офіційний рейтинг. Багато паперів, обійдених увагою крупних рейтингових агентств, можуть у результаті виявитися вельми привабливими для інвестицій, якщо зуміти грамотно оцінити ступінь їх ризикованої.
- По-друге, оновлення офіційних рейтингів відбувається не так часто, як хотілося б. Уміння заздалегідь, до того як це стане загальнодоступною інформацією, передбачити зміну рейтингів, очевидно, дає інвесторам додаткові конкурентні переваги.
- Нарешті, розгадавши стратегію «офіційного» рейтінгування, інвестори можуть сподіватися поліпшити якість оцінки фінансового стану емітентів шляхом інтенсивнішого статистичного аналізу, отримавши, таким чином, перевагу над тими, хто користується офіційними рейтингами.

Приведені вище доводи обгрунтовують наступну постановку задачі для нейроаналіза: на основі загальнодоступної фінансової звітності компаній-емітентів постаратися відтворити рейтинги Standard & Poor's або Moody's. Не дивлячись на наявність неформальної компоненти, представляється вірогідним, що алгоритмічна складова цих рейтингів досить велика. Врешті-решт, загальна чисельність аналітиків в обох провідних агентствах разом узятих не перевищує 100 чоловік. Отже справитися з обробкою постійно оновлюваних даних про

4000 емітентів вони можуть лише використовуючи в основному автоматизовані процедури.

Спроби змоделювати алгоритм рейтингування облігацій робилися з 60-х років і базувалися на методі лінійної регресії. Типовий відсоток вгадування рейтингу в цих моделях складає приблизно 60%. Оскільки можливості нелінійного нейромережевого моделювання ширші, недивно, що перші ж спроби застосувати нейромережі показали істотно кращі результати - на рівні 88% для відтворення окремої градації рейтингу. Складніші нейромережеві моделі здатні з прийнятною точністю відтворювати широкий діапазон рейтингів облігацій по набору ключових фінансових індикаторів фірм-імітентов. Відмітимо, що не дивлячись на непогані, загалом, результати, подібні нейромережеві моделі вельми компактні. В якості вхідних змінних зазвичай використовується від 6 до 10 фінансових індикаторів, що є відношенням найбільш значущих статей балансів і звітів про прибутки і збитки корпорацій. Якість відтворення «тонких» градацій рейтингу агентства Standard & Poor's, досягнуте цією моделлю, ілюструється на малюнку 3.49.

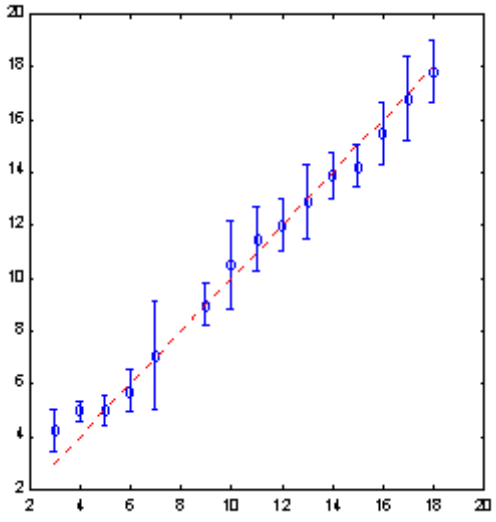


Рис. 3.49. Відтворення 15-ти градацій рейтингу Standard & Poor's.

Оцінка акцій. На відмінність від облігацій акції корпорацій не гарантують повернення відсотків і основної суми боргу. Проте, оцінка перспективності різних активів в пакетах акцій є одним з головних завдань

будь-якого інвестора. Одним з підходів до цього є рейтингування акцій, засноване на ширшому крузі фінансових показників компаній, доступних з їх фінансових звітів. Результативність подібного підходу ілюструють рейтинги провідного консультаційного агентства США по інвестиціях в акції - Value Line. Раз в тиждень це агентство розбиває акції близько 1700 компаній по 5 рейтинговим категоріям. Статистичні дослідження підтверджують значущість рейтингу Value Line. А саме, пакети, складені з акцій вищої рейтингової категорії, систематично дають великий прибуток в перебігу найближчого кварталу. Є підстави припускати, що квартальні звіти корпорацій впливають на курс акцій. Зокрема, несподівано високі прибутки (збитки) статистично значущо корелюють з підвищенням (пониженням) курсів акцій. Причому, ця кореляція існує достатньо довго - в перебігу принаймні двох місяців з дня публікації звіту. Отже, інвестор має можливість отримати певну вигоду з фінансової звітності корпорацій.

У більш загальній постановці йдеться про прогнозування фінансового «здоров'я» корпорації на підставі її фінансової звітності. Нетривіальним моментом тут є кількісне визначення фінансового благополуччя. Можна, як і у випадку з облігаціями, скористатися для навчання мережі рейтингами, наприклад, згаданого вище агентства Value Line для відтворення цієї, загалом суб'єктивної оцінки компанії. Можна спробувати використовувати як індикатор благополуччя об'єктивніший критерій - ринковий курс акцій в найближчому або віддаленішому майбутньому. Проте, ринковий курс може бути схильний до сильних флуктуацій чисто спекулятивного характеру. Нарешті, можна скористатися вказівками найсуворішого вчителя, досліджуючи крайню форму прояву фінансового «нездужання» - банкрутство. Аналіз банкрутства, таким чином, може служити джерелом об'єктивних оцінок стійкості фінансового положення фірм.

Прогноз ризиків банкрутства. Спочатку приведемо декілька цифр, що ілюструють «ціну питання». Світовий ринок тільки міжбанківських кредитів оцінюється в $58 трлн. Це майже в два рази перевищує світовий об'єм цінних

паперів. Природно, що оцінка ризику неповернення кредитів має для банків першорядне значення. Кількість банкрутств в США впродовж 2000 - 2005 років зростала щорічно приблизно на 14%. Таким чином, прогноз банкрутств, особливо в кризових економічних умовах, є насущним завданням економічного аналізу.

Якщо в проблемі рейтингування завданням нейромережі було відтворити думки експертів про надійність корпорації, то нейромережевий прогноз банкрутств заснований на статистичній обробці конкретних прикладів банкрутств. У такій постановці задачі нейромережі важливо самій стати експертом, що визначає фінансову стабільність корпорації, грунтуючись виключно на об'єктивній інформації - показниках фінансової звітності. Зазвичай від нейромережі потрібно оцінити вірогідність банкрутства через певний проміжок часу по доступній на даний момент фінансовій звітності. Як входи використовують фінансові індикатори - відносини балансових статей, що найбільш повно відображають певні сторони фінансового положення фірми. Найбільш загальний підхід полягає у використанні в якості входів логарифмів укрупнених статей балансів і звітів про прибутки (збитки). Нейромережа в цьому випадку сама вибере найбільш значущі лінійні комбінації входів, яким відповідатимуть найбільш значущі відносини різних статей в потрібних пропорціях. Використання індикаторів, з іншого боку, допомагає в інтерпретації результатів нейромоделювання якщо скористатися, наприклад, технікою проріджування зв'язків і витягання правил. Відмітимо, що використання описаних вище індикаторів лежить також в основі загальноприйнятої методики рейтінгованія банків CAMEL.

Узагальнюючи досвід порівняльного аналізу прогнозів банкрутств різними методиками, відзначимо:

- Нейромережеве моделювання забезпечує якнайкращу точність прогнозу банкрутств: близько 90%, в порівнянні з 80%-85% точністю для інших статистичних методик.

- При бажанні можна підвищити «підозрілість» нейромережі, забезпечивши точність виявлення банкротів аж до 99% - за рахунок зниження вимог до помилок.
- Банкрутства можна упевнено передбачати за декілька років до їх фактичного настання, причому точність прогнозу за два роки практично не відрізняється від точності прогнозу за рік. Таким чином, неявні сигнали неблагополуччя присутні у фінансовій звітності фірми задовго до її краху.

Корисність навчання мережі на прикладах збанкрутілих фірм полягає також в тому, що така мережа виробляє функцію - чисельний показник фінансового здоров'я фірми, міру її стійкості. Проте, стійкість не є єдиним можливим критерієм оцінки діяльності фірми. Акціонери, наприклад, зацікавлені не тільки в нескінченно довгому існуванні фірми, але і в отриманні достатньо вагомого прибутку. Важливо, крім того, не тільки стан фірми на справжній момент, але і характеристики існуючих тенденцій. Тут значущим може опинитися інший набір чинників, що дає іншу оціночну функцію. Так, висока прибутковість може забезпечити підвищення надійності в майбутньому. Тим часом, неясно яким чином можна навчати нейромережу на «майбутній успіх» за відсутності такого ж чіткого критерію успіху, яким є банкрутство для невдачі. Ці об'єктивні труднощі можна подолати, якщо пригадати, що фірма існує не сама по собі, а в співтоваристві подібних нею фірм-конкурентів. І саме в зіставленні з цим співтовариством можна говорити про сильні і слабкі сторони її діяльності. Ці міркування підводять нас до іншої постановки задачі: комплексної оцінки фінансового стану фірми шляхом систематичного порівняння її показників з показниками решти учасників даного ринку. Такий підхід не вимагає знання готових відповідей, оскільки заснований на навчанні без вчителя.

Порівняльний аналіз фінансового стану фірм. Порівняльний аналіз, на відмінність від рейтингування, припускає введення не однієї, а декількох оціночних координат. Це дозволяє краще використовувати наявну інформацію, точніше позиціонувати фірму серед інших. З іншого боку, для осяжності

результатів порівняльного аналізу, кількість параметрів порівняння повинна бути по можливості мінімальним. У вузькому сенсі «осяжність» вимагає введення не більше двох координат - щоб відносна позиція фірми могла бути представлена точкою на двовимірній карті, а різні фінансові показники могли бути візуалізовані у вигляді двовимірних поверхонь.

З математичної точки зору ця задача зводиться до оптимального стиснення інформації про фінансовий стан фірми, тобто відображенні інформації мінімальним числом параметрів при заданому рівні огрублення або мінімізації втрат інформації при заданому числі узагальнених координат. Для цілей візуалізації, вигідно обмежитися двохпараметричним уявленням. Це вже істотний крок вперед в порівнянні з однопараметричним рейтингом.

Для ілюстрації описуваного підходу будемо використовувати дані Центрального Банку України про річні баланси і звіти про прибутки (збитки) приблизно 100 українських банків за 2004 - 2005 роки. Кожен банк при цьому описується 30 фінансовими показниками - відношенням балансових статей до загальної суми активів банку. Подібна нормалізація приводить всі статті до єдиного масштабу, згладжуючи відмінності між крупними і дрібними банками. Згідно такому підходу надійність банку будемо характеризувати одним фінансовим показником - відношенням власного капіталу до привернутого. Більш загальний підхід - використовувати не дві окремі компоненти, а дві лінійні комбінації всіх 30 початкових параметрів, що найкращим чином представляють наявні дані (рис. 3.50).

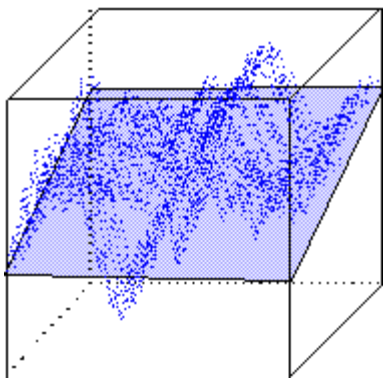


Рис. 3.50. Лінійна апроксимація багатовимірних даних.

Кожен банк представлен точкою в 30-мірному просторі і задача полягає в проведенні двовимірної площини в цьому просторі, що забезпечує мінімальне середньоквадратичне відхилення наявних точок від цієї площини. Подібне лінійне наближення створюється методом головних компонент. Якщо дійсне розташування точок не сильно відхиляється від площини, цей метод може дати непогане початкове наближення. Проте, виявляється, що в даном випадку це не так. Середньоквадратичне відхилення для випадку двох головних компонент виявилося рівним майже половині від загальної дисперсії. Таким чином, навіть оптимальний варіант лінійного стиснення не дає можливості візуалізувати фінансове положення банків.

У цій ситуації природно звернутися до нелінійного статистичного аналізу, тобто до нейромережевого моделювання. Нагадаємо, що методом, що дає оптимальне представлення інформації у вигляді координат двовимірної сітки, є побудова топографічних карт (карт Кохонена). Таким чином, можна з прийнятною точністю описати фінансовий стан українських банків використовуючи всього лише два узагальнені фінансові індикатори, а саме - дві координати на двовимірній карті Кохонена. Кожен банк за станом свого балансового звіту відображається конкретним осередком на карті. Осередки з однаковими координатами містять банки з схожим фінансовим станом. Чим далі на карті координати банків, тим більше відрізняється один від одного їх фінансовий портрет.

Розташування на карті банків з відкликаною ліцензією. Достоїнства карти Кохонена починають виявлятися після нанесення на неї якої-небудь графічної інформації. Малюнок 3.51 показує як виглядає карта Кохонена, на якій відмічені ячейки, що містять банки з відкликаними за наслідками 2004 року ліцензіями. Видно, що банки з відкликаними ліцензіями групуються в правому верхньому кутку карти – «зоні ризику». Ми побачимо, що ця зона має і інші ознаки неблагополуччя.

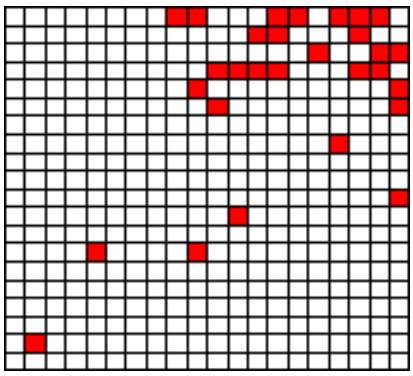


Рис. 3.51. Ячейки, що містять хоч би один банк з відкликаною в 2004 році ліцензією.

Відзначимо, що на відміну від аналізу банкрутств тут інформація про банкрутства не брала участь в навчанні мережі. Вона зображена на вже готовій карті, являсь лише індикатором області параметрів з підвищеним ризиком банкрутства. Ця особливість описуваної методики дозволяє виявити область ризику по відносно невеликому числу прикладів.

Карта розмірів банків. Розглянемо, наприклад як розташовуються на побудованій карті банки різних розмірів (рис. 3.52). Розміри банків беруться в логарифмічній шкалі, причому клітки, що відрізняються на одну колірну градацію, містять банки з п'ятикратним відношенням активів. Нагадаємо, що величина активів банків була спочатку виведена з набору параметрів, оскільки вона використовувалася для нормування решти статей. Не дивлячись на це, банки різних розмірів розташовуються не хаотично, а регулярним чином, що свідчить про значущість розміру банку при виборі їм своїй фінансовій стратегії.

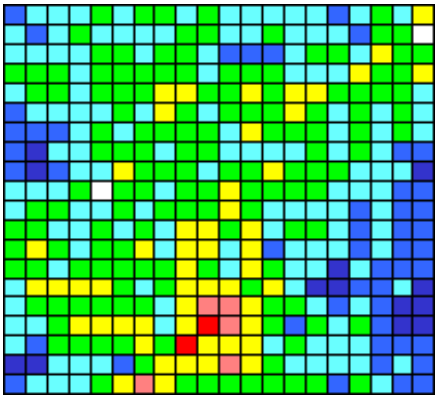


Рис. 3.52. Розмір активів українських банків.

Візуально на карті можна виділити наступні великі групи банків: великі банки (низ карти), малі банки (група зліва і група справа) і середні банки. Розфарбовування, що відображає відносний розмір статутного фонду показує, що між двома групами малих банків є істотні відмінності: банки в нижньому правому куті карти практично не ростуть (рис. 3.53).

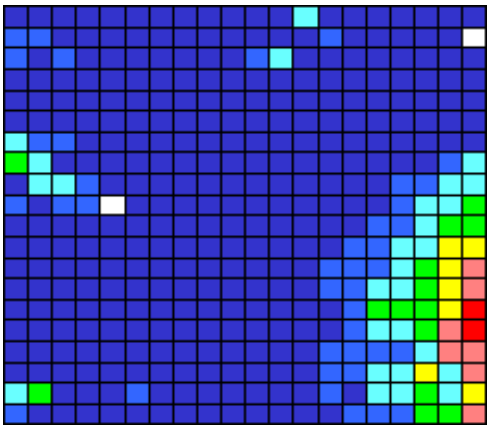


Рис. 3.53. Статутний фонд банків.

Порівняння з розташуванням банків-банкротів, показує, що вірогідність банкрутства як великих так і малих банків в 2004 - 2005 роках була невелика.

Приведені вище розфарбовування в сукупності утворюють атлас, що відображає фінансовий стан банків або інших фірм, що займаються схожими видами бізнесу. Цей атлас дає графічне відображення положення будь-якої конкретної фірми серед конкурентів і може використовуватися як зручний засіб фінансового аналізу. Зокрема, можна розглядати еволюцію фінансового положення окремої фірми в часі, виявляти існуючі тенденції і цикли. З погляду макроекономіки зручність такого роду карт полягає в тому, що площі на цій карті приблизно пропорційні долі фірм через більш-менш рівномірне заповнення ячеек. Таким чином можна зримо уявляти собі, наприклад, частку крупних банків або банків, що зазнають труднощі з поверненням кредитів.

Передбачення фінансових часових рядів. Прогноз фінансових часових рядів - необхідний елемент будь-якої інвестиційної діяльності. Сама ідея інвестицій - вкладення грошей зараз з метою отримання доходу в

майбутньому - грунтується на ідеї прогнозування майбутнього. Відповідно, прогноз фінансових часових рядів лежить в основі діяльності всієї індустрії інвестицій - всіх бірж і небіржових систем торгівлі цінними паперами. Приведемо декілька цифр, що ілюструють масштаб цієї індустрії прогнозів. Денний оборот ринку акцій тільки в США перевищує $40 млрд. Депозитарій DTC (Depositary Trust Company) в США, де зареєстровано цінних паперів на суму $21 трлн, реєструє в день операцій приблизно на $750 млрд. Ще активніше йде торгівля на світовому валютному ринку FOREX. Його денний оборот перевищує $1000 млрд. Це приблизно 1/50 всього сукупного капіталу людства. Відомо, що 99% всіх операцій - спекулятивні, тобто направлені не на обслуговування реального товарообігу, а поміщені з метою витягання прибутку по схемі «купив дешевше - продав дорожче». Всі вони засновані на прогнозах зміни курсу учасниками операції. Причому, що важливо, прогнози учасників кожної операції протилежні один одному. Отже об'єм спекулятивних операцій характеризує ступінь відмінностей в прогнозах учасників ринку, тобто реально - ступінь непередбачуваності фінансових часових рядів.

Методика передбачення часових рядів. Спершу змалюємо загальну схему нейромережевого передбачення часових рядів (рис. 3.54). Почнемо з етапу занурення. Як ми зараз переконаємося, не дивлячись на те, що передбачення, здавалося б, є екстраполяцією даних, нейромережі, насправді, вирішують задачу інтерполяції, що істотно підвищує надійність рішення. Передбачення часового ряду зводиться до типової задачі нейроаналіза - апроксимації функції багатьох змінних по заданому набору прикладів - за допомогою процедури занурення ряду в багатовимірний простір. Наприклад, d -мірний лаговий простір ряду X_t складається із d значень ряду в послідовні моменти часу: $X_{t-d} = (X_{t-1},...,X_{t-d})$.

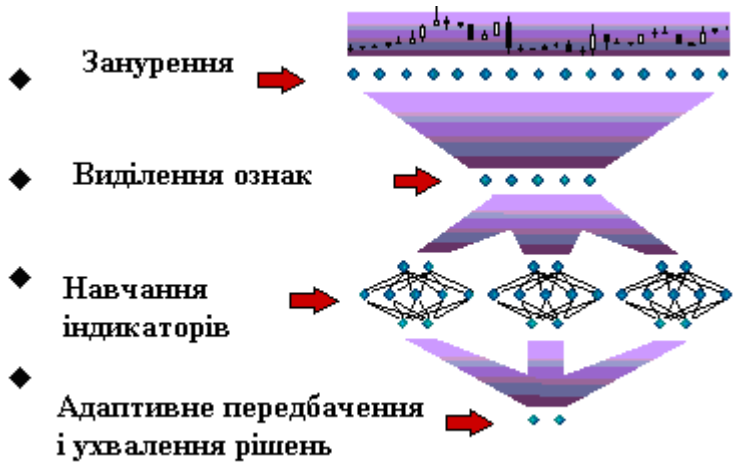


Рис. 3.54. Схема технологічного циклу передбачення ринкових часових рядів.

Для динамічних систем доведена наступна теорема Такенса. Якщо часовий ряд породжується динамічною системою, тобто значення X_t є довільна функція стану такої системи, існує така глибина занурення d (приблизно рівна ефективному числу мір свободи даної динамічної системи), яка забезпечує однозначний прогноз наступного значення часового ряду. Таким чином, вибравши чимале d можна гарантувати однозначну залежність майбутнього значення ряду від його d попередніх значень: $X_t = f(X_{t-d})$, тобто передбачення часового ряду зводиться до завдання інтерполяції функції багатьох змінних. Нейромережу далі можна використовувати для відновлення цієї невідомої функції по набору прикладів, заданих історією даного часового ряду. Навпаки, для випадкового ряду знання минулого нічого не дає для передбачення майбутнього. Тому, згідно теорії ефективного ринку, розкид значень ряду, що передбачається, на наступному кроці при зануренні в лаговий простір не зміниться.

Для навчання нейромережі недостатньо сформувати навчальні набори входів-виходів. Необхідно також визначити помилку прогнозів мережі. Середньоквадратична помилка, використовувана за умовчанням в більшості нейромережевих додатків, не має великого «фінансового сенсу» для ринкових рядів. Тому окремо слід розглянути специфічні для фінансових часових рядів функції помилки і показати їх зв'язок з можливою нормою прибутку.

Наприклад, для вибору ринкової позиції надійне визначення знаку зміни курсу важливіше, ніж пониження середньоквадратичного відхилення. Хоча ці показники і зв'язані між собою, мережі оптимізовані по одному з них даватимуть гірші прогнози іншого.

Основна специфіка передбачення часових рядів лежить в області передобробки даних. Процедура навчання окремих нейромереж стандартна. Як завжди, наявні приклади розбиваються на три вибірки: навчальну, валідаційну і тестову. Перша використовується для навчання, друга - для вибору оптимальної архітектури мережі або для вибору моменту зупинки навчання. Нарешті, третя, яка взагалі не використовувалася в навчанні, служить для контролю якості прогнозу навченої нейросеті. Проте, для сильно зашумленних фінансових рядів істотний виграш в надійності прогнозів здатне дати використання комітетів мереж. Поліпшення якості прогнозів можливе також за рахунок використання нейромереж із зворотними зв'язками. Такі мережі можуть володіти локальною пам'яттю, що зберігає інформацію про далеке минуле, ніж те, що в явному вигляді присутнє на входах.

В якості вхідних змінних логічно вибирати найбільш статистично незалежні величини, наприклад, зміни котирувань δC_t або логарифм відносного приросту $\log(C_t / C_{t-1}) \approx \Delta C_t / C_{t-1}$. Останній вибір хороший для тривалих часових рядів, коли вже помітно вплив інфляції. В цьому випадку прості різниці в різних частинах ряду матимуть різну амплітуду, оскільки фактично вимірюються в різних одиницях.

Одним з самих «слабких місць» у фінансових прогнозах є дефіцит прикладів для навчання нейросеті. Фінансові ринки, взагалі кажучи, не стаціонарні. З'являються нові фінансові інструменти, для яких ще не накопичена історія, змінюється характер торгівлі на колишніх ринках. У цих умовах довжина доступних для навчання нейросеті часових рядів вельми обмежена. Проте, можна підвищити число прикладів, використовуючи для цього ті або інші апріорні міркування про інваріанти динаміки часового ряду. Це ще одне фізико-математичне поняття, здатне значно поліпшити якість

фінансових прогнозів. Йдеться про генерацію штучних прикладів, що отримуються з вже отриманих шляхом застосуванням до них різного роду перетворень.

Пояснимо основну думку на прикладі. Психологічно виправдано наступне припущення: гравці звертають увагу, в основному, на форму кривої цін, а не на конкретні значення по осях. Тому якщо небагато розтягнути по осі котирувань весь часовий ряд, то отриманий в результаті такого перетворення ряд також можна використовувати для навчання разом з результатним. Ми, таким чином, подвоїли число прикладів за рахунок використання апріорної інформації, витікаючої з психологічних особливостей сприйняття часових рядів учасниками ринку. Більш того, ми не просто збільшили число прикладів, але і обмежили клас функцій, серед яких шукається рішення, що також підвищує якість передбачення.

Особливістю передбачення фінансових часових рядів є прагнення до отримання максимального прибутку, а не мінімізації середньоквадратичного відхилення, як це прийнято у разі апроксимації функцій. У простому випадку щоденної торгівлі прибуток залежить від вірно вгаданого знаку зміни котирування. Тому нейромережі потрібно орієнтувати саме на точність вгадування знаку, а не самого значення. Знайдемо як зв'язана норма прибутку з точністю визначення знаку в простій постановці щоденного входження в ринок. Позначимо на момент t: повний капітал гравця K_t, відносна зміна котирування $x_t = \Delta C_t / C_{t-1}$, а в якості виходу мережі візьмемо ступінь її упевненості в знаку цієї зміни $y_t \in [-1,1]$. Така мережа з вихідною нелінійністю вигляду $y = \tanh(\alpha)$ навчається передбачати знак зміни і видає прогноз знаку з амплітудою пропорційної його вірогідності. Тоді зростання капіталу на кроці t запишеться у вигляді:

$$K_t = K_{t-1}[1 + \delta |x_t| (x_t, y_t)],$$

де δ - частка капіталу «в грі». Виграш за весь час гри:

$$K_t = K_0 \exp(\sum_{k=1}^{t} \ln[1 + \delta x_k (y_k)])$$

нам і належить максимізувати, вибравши оптимальний розмір ставок σ. Хай в середньому гравець вгадує частку $p = 1/2 + \varepsilon$ знаків і, відповідно, помиляється з вірогідністю $q = 1/2 - \varepsilon$. Тоді логарифм норми прибутку

$$\ln(K_t / K_0) = t[\, p \ln(1 + |x|\delta) + q \ln(1 - |x|\delta)],$$

а отже і сам прибуток, буде максимальним при значенні $\delta = (p - q)(|x| / x^2)$ і складе в середньому:

$$\ln(K_t / K_0) \approx t(p - q)^2 \frac{|x|^2}{2x^2} = 2\alpha t \varepsilon^2 .$$

Тут ми ввели коефіцієнт $\alpha = |x|^2 / x^2 \le 1$. Наприклад, для Гауссова розподілу $\alpha \approx 0.8$. Ступінь передбаченості знаку безпосередньо пов'язаний з кросс-ентропією, яку можна оцінити a priory методом box-counting (рис. 3.55).

Рис. 3.55. Доля правильно вгаданих напрямків змін ряду.

У результаті отримуємо наступну оцінку норми прибутку при заданій величині передбаченості знаку I, вираженої в бітах $K_t = K_0 2^{\alpha I t}$. Тобто, для ряду з передбаченістю I в принципі можливо подвоїти капітал за $t = 1/\alpha I$ входжень в ринок. Таким чином, навіть невелика передбаченість знаку зміни котирувань здатна забезпечити вельми помітну норму прибутку. Підкреслимо, що

оптимальна норма прибутку вимагає достатньо акуратної гри, коли при кожному входженні в ринок гравець ризикує строго певною часткою капіталу:

$$\Delta K / K = \delta|x| = (p-q)(|x|^2 / x^2) = 2\alpha\varepsilon \approx 1.6\varepsilon,$$

де ΔK - типова при даній волатільності ринку $|x|$ величина виграшу або програшу. Як менші, так і більші значення ставок зменшують прибуток. Причому, занадто ризикована гра може привести до програшу при будь-якій передбачаючій здатності. Цей факт ілюструє рисунок 3.56.

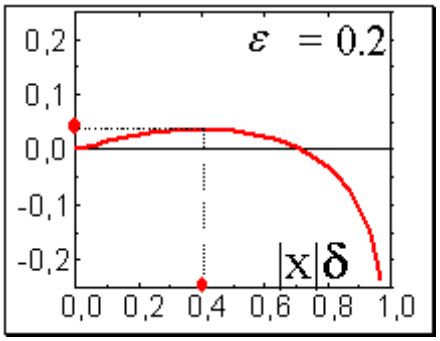


Рис. 3.56. Залежність середньої норми прибутку від вибору частки капіталу.

Використання комітетів мереж. Із-за випадковості у виборі початкових значень синаптичних вагів, прогнози мереж, навчених на одній і тій же вибірці, будуть різнитися. Цей недолік можна перетворити на перевагу, організувавши комітет нейро-експертів, що складається з різних нейромереж. Розкид в прогнозах експертів дасть уявлення про ступінь упевненості цих передбачень, що можна використовувати для правильного вибору стратегії гри. Легко показати, що середнє значень комітету повинно давати кращі прогнози, чим середній експерт з цього ж комітету. Хай помилка i-го експерта для значення входу дорівнює $\varepsilon_i(x)$. Середня помилка комітету завжди менше середньоквадратичної помилки окремих експертів в силу нерівності Коші:

$$\left(\frac{1}{L}\sum_{i=1}^{L}\varepsilon_i\right)^2 \le \frac{1}{L}\sum_{i=1}^{L}\varepsilon_i^2$$

Причому, зниження помилки може бути доволі помітним. Так, помилка комітету з L експертів в $\sqrt{L}$ раз менше, ніж середня індивідуальна помилка одного експерта.

$$E_L^2 \equiv \left(\frac{1}{L} \sum_{i=1}^{L} \varepsilon_i \right)^2 = \frac{1}{L^2} \left(\sum_{i=1}^{L} \varepsilon_i^2 + \sum_{i \neq j} \varepsilon_i \varepsilon_j \right) = \frac{1}{L^2} \sum_{i=1}^{L} \varepsilon_i^2 = \frac{1}{L} E_1^2 .$$

Тому, в прогнозах завжди краще спиратися на середні значення всього комітету. Ілюстрацією цього факту служить рисунок 3.57. Виграш комітету (кружки) вищий, ніж виграш середнього експерта. Рахунок вгаданих знаків для комітету 59:41.

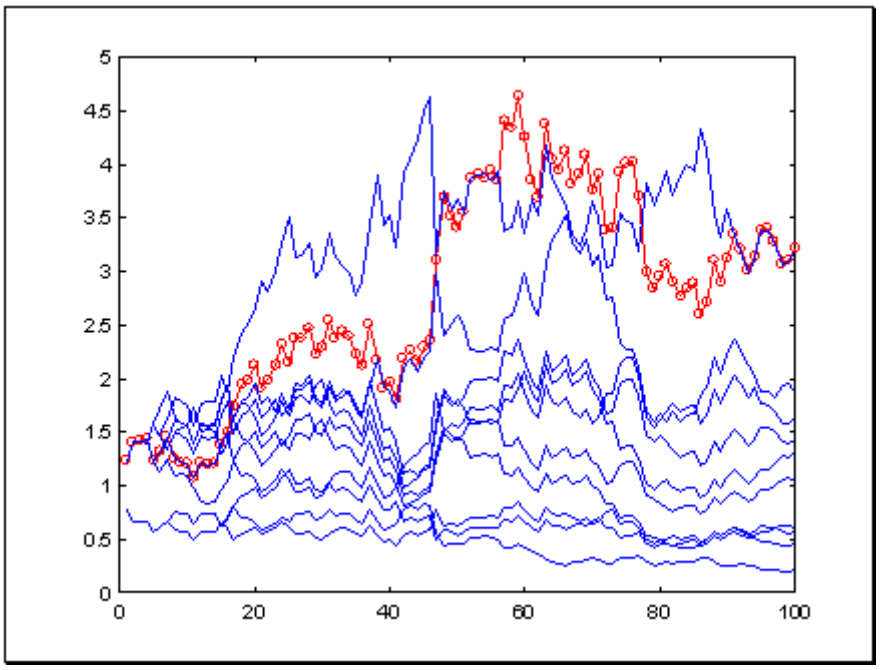


Рис. 3.57. Норма прибутку на останніх 100 значеннях ряду при передбаченні комітетом з 10 мереж.

Як видно з приведеного вище рисунка, в даному випадку виграш комітету навіть вище, ніж виграш кожного з експертів. Таким чином, метод комітетів може істотно підвищити якість прогнозування. Зверніть увагу на абсолютне значення норми прибутку: капітал комітету зріс в 3.25 разу при 100 входженнях в ринок.

Завершуючи огляд нейрокомп'ютерних технологій зупинимося на деяких перспективних напрямах застосування нейронних мереж.

Штучні нейронні мережі і експертні системи. Останніми роками над штучними нейронними мережами домінували логічні і символьно-операційні дисципліни. Наприклад, широко пропагувалися експертні системи, у яких є багато помітних успіхів, так само, як і невдач. Дехто говорить, що штучні нейронні мережі замінять собою сучасний штучний інтелект, але багато що свідчить про те, що вони існуватимуть, об'єднуючись в системах, де кожен підхід використовується для вирішення тих завдань, з якими він краще справляється. Ця точка зору підкріплюється тим, як люди функціонують в нашому світі. Розпізнавання образів відповідає за активність, що вимагає швидкої реакції. Оскільки дії здійснюються швидко і несвідомо, то цей спосіб функціонування важливий для виживання у ворожому оточенні. Уявите тільки, що було б, якби наші предки вимушені були обдумувати свою реакцію на хижака, що стрибнув? Коли наша система розпізнавання образів не в змозі дати адекватну інтерпретацію, питання передається у вищі відділи мозку. Вони можуть запитати додаткову інформацію і займуть більше часу, але якість отриманих в результаті рішень може бути вище. Можна уявити собі штучну систему, що наслідує такому розподілу праці.

Штучна нейронна мережа реагувала б в більшості випадків відповідним чином на зовнішнє середовище. Оскільки такі мережі здатні указувати довірчий рівень кожного рішення, то мережа «знає, що вона не знає» і передає даний випадок для дозволу експертній системі. Рішення, що приймаються на цьому вищому рівні, були б конкретними і логічними, але вони можуть потребувати збору додаткових фактів для отримання остаточного висновку. Комбінація двох систем була б могутнішою, ніж кожна з систем окремо, слідуючи при цьому високоефективній моделі, що дається біологічною еволюцією.

Надійність нейронних мереж. Перш ніж штучні нейронні мережі можна буде використовувати там, де поставлено на карту людське життя або цінне майно, повинні бути вирішені питання, що відносяться до їх надійності. Подібно до людей, структуру мозку яких вони копіюють, штучні нейронні мережі зберігають до певної міри непередбачуваність. Єдиний спосіб точно

знати вихід полягає у випробуванні всіх можливих вхідних сигналів. У великій мережі така повна перевірка практично нездійснена і повинні використовуватися статистичні методи для оцінки функціонування. В деяких випадках це неприпустимо. Наприклад, що є допустимим рівнем помилок для мережі, що управляє системою космічної оборони? Більшість людей скажуть, будь-яка помилка недопустима, оскільки веде до величезного числа жертв і руйнувань. Це відношення не міняється від тієї обставини, що людина в подібній ситуації також може припускатися помилок. Проблема виникає із-за допущення повної безпомилковості комп'ютерів. Оскільки штучні нейронні мережі іноді здійснюватимуть помилки навіть при правильному функціонуванні, то, як відчувається багатьма, це веде до ненадійності - якості, яку ми вважаємо неприпустимою для наших машин.

Схожа трудність полягає в нездатності традиційних штучних нейронних мереж «пояснити», як вони вирішують задачу. Внутрішнє уявлення, що виходить в результаті навчання, часто настільки складно, що його неможливо проаналізувати, за винятком найпростіших випадків. Це нагадує нашу нездатність пояснити, як ми пізнаємося людину, не дивлячись на відмінність у відстані, куті, освітленні і минулих роках. Експертна система може прослідкувати процес своїх міркувань в зворотному порядку, так що людина може перевірити її на розумність. Можливе вбудовування цієї здатності в штучні нейронні мережі може істотно вплинути на прийнятність цих систем.

Логічно прозорі нейронні мережі і виробництво явних знань з даних. Одним з основних недоліків нейронних мереж, з точки зору багатьох користувачів, є те, що нейронна мережа вирішує задачу, але не може розповісти як. Іншими словами з навченої нейронної мережі не можна витягувати алгоритм рішення задачі. Проте спеціальним чином побудована процедура дозволяє вирішити це завдання.

Задамося класом мереж, які вважатимемо логічно прозорими (тобто такими, які вирішують задачу зрозумілим для нас способом, для якого легко сформулювати словесний опис у вигляді явного алгоритму). Наприклад

зажадаємо, аби всі нейрони мали не більше трьох вхідних сигналів. Задамося нейронною мережею в якій всі вхідні сигнали подаються на всі нейрони вхідного шару, а всі нейрони кожного наступного шару приймають вихідні сигнали всіх нейронів попереднього шару. Навчимо мережу безпомилковому рішенню задачі. Після цього вироблятимемо контрастування у декілька етапів. Процедура контрастування заснована на оцінці значущості вагів зв'язків в мережі. Основною її метою є спрощення технічною реалізацію мережі і створення навику мережі, який був би зрозумілішим.

На першому етапі контрастуватимемо лише ваги зв'язків нейронів вхідного шару. Якщо після контрастування в деяких нейронів залишилося більше трьох вхідних сигналів, то збільшимо число вхідних нейронів. Потім аналогічну процедуру виробимо по черзі для всіх останніх шарів. Після завершення описаної процедури буде отримана логічно прозора мережа (рис. 3.58).

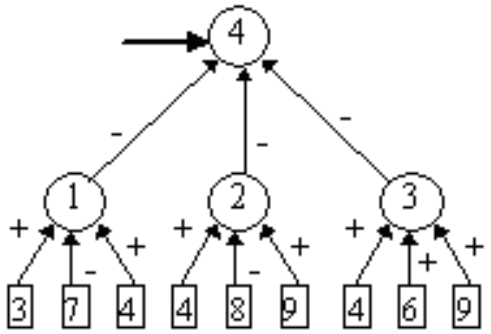


Рис. 3.58. Логічно прозора нейронна мережа.

Як приклад приведемо інтерпретацію алгоритму міркувань, отриманого по логічно прозорій мережі. Постановка задачі слідуюча: по відповідях на 12 питань необхідно передбачити перемогу правлячої або опозиційної партії. Відповіді на питання описують ситуацію на момент перед виборами. Негативний сигнал на виході мережі інтерпретується як передбачення перемоги правлячій партії. Інакше відповіддю вважається перемога опозиційної партії. Всі нейрони реалізовували порогову функцію, рівну 1, якщо сума алгебри

вхідних сигналів нейрона більше або рівна 0, і -1 при сумі меншою 0. Відповідь мережі базується на проявах двох синдромів: синдрому політичної нестабільності і синдрому поганої політики. Таким чином, для перемоги правлячої партії необхідна відсутність (-1) обох синдромів. На малюнку 3.59 приведена структура логічно прозорої нейронної мережі, яка вирішувала задачу про передбачення результатів виборів президента США.

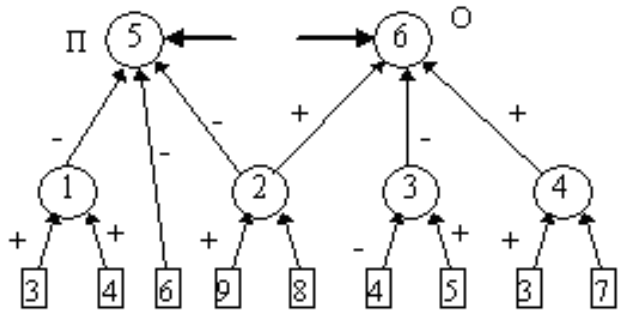


Рис.3.59. Логічно прозора мережа для задачі передбачення.

Технологія здобуття явних знань з даних за допомогою навчених нейронних мереж виглядає досить просто. Перший етап: навчаємо нейронну мережу вирішувати базову задачу. Зазвичай базовою є задача розпізнавання, передбачення і тому подібне. В більшості випадків її можна трактувати як задачу про заповнення пропусків в даних. Другий етап: за допомогою аналізу показників значущості, контрастування і донавчання приводимо нейронну мережу до логічно прозорого вигляду - так, щоб отриманий навик можна було «прочитати».

Тест.

1. Які обставини в сфері бізнесу викликали появу нейрокомп'ютерних технологій?

а) складність застосування алгоритмічного підхіду при вирішенні бізнес-задач, а отримані рішення не є однозначними;

б) можливість формалізації задач бізнесу та застосування математичних моделей для їх розв'язання;

в) великі витрати часу і фінансів для вирішення задач бізнесу, а також необхідність застосування потужних обчислювальних систем;

г) задачі бізнеса, в яких строгий алгоритмічний підхід неможливий, а отримати точне рішення принципово не можна.

2. Під поняттям «нейронна мережа» Ви розумієте …

а) адаптивні системи для інтелектуального аналізу даних, які є математичною структурою, що імітує деякі аспекти роботи людського мозку;

б) комп'ютерні системи, які демонструють можливість застосування математичних моделей до задач узагальнення, кластеризації та прогнозування;

в) комп'ютерні системи, які демонструють здібність до неформального навчання, узагальнення і кластеризації некласифікованої інформації, здатність самостійно будувати прогнози;

г) комп'ютерні системи, які в діалоговому режимі демонструють здібність до узагальнення, кластеризації інформації та будування прогнозів.

3. Головну відмінність нейронних мереж від інших методів Ви бачите в тому, що …

а) нейромережі в принципі не потребують заздалегідь відомої моделі, а будують її самі лише на основі власного досвіду;

б) нейромережі в принципі не потребують заздалегідь відомої моделі, а будують її самі лише на основі інформації, яку отримали;
в) нейромережі в принципі потребують заздалегідь відомі моделі, а не будують її лише на основі інформації, яку отримали.

4. Основне завдання аналітика при застосуванні технології нейронних мереж полягає в наступному:
а) створити архітектуру нейронної мережі, алгоритм її навчання та розробити математичну модель її дії;
б) створити найбільш ефективну архітектуру нейронної мережі, алгоритм її роботи, кількість нейронів і види зв'язків між ними;
в) створити найбільш ефективну архітектуру нейронної мережі, алгоритм її навчання, кількість нейронів і види зв'язків між ними.

5. Важливою властивістю нейромереж є коннекціонізм, що означає …
а) підхід, заснований на представленні як пам'яті даних, так і алгоритмів системою правил;
б) підхід, заснований на представленні як пам'яті даних, так і алгоритмів системою функцій;
в) підхід, заснований на представленні як пам'яті даних, так і алгоритмів системою зв'язків і їх вагами.

6. Важливими парадигмами, пов’язаними з нейрокомп’ютінгом, Ви вважаєте …
а) локальність і паралелізм обчислювань;
б) застосування мов програмування;
в) навчання, засноване на даних;
г) універсальність навчальних алгоритмів;
д) дискретний характер обробки інформації.

7. Якщо Вам запропонували для виконання інтелектуального аналізу даних використовувати нейро-емулятори, то Ви розумієте, що це …

а) прикладні системи нейромережевого аналізу даних, які використовують алгоритми на звичайних комп'ютерах;

б) прикладні системи нейромережевого аналізу даних, які використовують емуляцію нейромереж на звичайних комп'ютерах;

в) прикладні системи нейромережевого аналізу даних, які використовують емуляцію нейромереж на нейрокомп'ютерах.

8. Яка ідея лежить в основі побудови нейронних мереж?

а) нейрони можна моделювати досить простими автоматами (штучними нейронами), а вся складність мозку визначається зв'язками між нейронами;

б) нейрони можна моделювати досить простими автоматами (штучними нейронами), а вся складність мозку визначається алгоритмами їх роботи;

в) нейрони можна моделювати досить простими автоматами (штучними нейронами), а вся складність мозку визначається математичними моделями.

9. Застосовуючи для аналізу даних шаровані нейронні мережі, Ви вважаєте, що це…

а) нейронні мережі, в яких нейрони розбиті на окремі групи (шари) так, що обробка інформації здійснюється пошарово;

б) нейронні мережі, в яких нейрони розбиті на окремі групи (шари) так, що обробка інформації здійснюється разом;

в)нейронні мережі, в яких нейрони розбиті на окремі групи (шари) так, що обробка інформації здійснюється комбіновано.

10. Вхідні, вихідні та приховані нейрони є елементами …

а) одношарових нейронних мереж;

б) багатошарових нейронних мереж;

в) синхронних нейронних мереж;

г) асинхронних нейронних мереж.

11. Застосовуючи для аналізу даних повнозв’язні нейронні мережі, Ви вважаєте, що це…

а) мережі, в яких кожен нейрон передає свій вихідний сигнал решті нейронів, не включаючи самого себе;

б) мережі, в яких окремі нейрони передають свій вихідний сигнал решті нейронів, включаючи самих себе;

в) мережі, в яких кожен нейрон передає свій вихідний сигнал решті нейронів, включаючи самого себе.

12. Під нейронними мережами зі зворотніми зв’язками, Ви розумієте:

а) мережі, які характеризуються фіксованою структурою, одиничним навчанням, корегуванням вагів по помилках;

б) мережі, які характеризуються фіксованою структурою, ітераційним навчанням, корегуванням вагів по помилках;

в) мережі, які характеризуються довільною структурою, ітераційним навчанням, корегуванням вагів по помилках.

13. Під нейронними мережами без зворотних зв’язків, Ви розумієте:

а) мережі, які характеризуються простотою реалізації і гарантованим отриманням відповіді після проходження даних по шарах;

б) мережі, які характеризуються простотою реалізації і вірогідним отриманням відповіді після проходження даних по шарах;

в) мережі, які характеризуються складною реалізацією і гарантованим отриманням відповіді після проходження даних по шарах.

14. Якщо для інтелектуального аналізу даних Ви застосовуєте нейронні мережі прямого розповсюдження, то розумієте, що це …

а) мережі, в яких всі зв'язки направлені як від вхідних нейронів до вихідних так і навпаки;

б) мережі, в яких деякі зв'язки направлені строго від вхідних нейронів до вихідних;

в) мережі, в яких всі зв'язки направлені строго від вхідних нейронів до вихідних.

15. Характерною особливість рекурентних нейронних мереж є …

а) наявність блоків затримки і зворотних зв'язків, що дозволяє їм обробляти статистичні моделі;

б) наявність блоків динамічної затримки і зворотних зв'язків, що дозволяє їм обробляти динамічні моделі;

в) наявність блоків динамічної затримки і відсутність зворотних зв'язків, що дозволяє їм обробляти математичні моделі.

16. В чому полягає процес навчання нейронної мережі?

а) в підстроюванні її внутрішніх параметрів під конкретне завдання;

б) в розв’язанні прикладів конкретного завдання;

в) в накопиченні прикладів для конкретного завдання.

17. Процес навчання з вчителем передбачає, що …

а) для кожного навчального вхідного прикладу потрібне знання правильної відповіді;

б) для кожного навчального вхідного прикладу потрібне знання правильної відповіді або функції кількісної оцінки відповіді;

в) для кожного навчального вхідного прикладу потрібне знання правильної відповіді або функції оцінки якості відповіді.

18. Процес навчання без вчителя передбачає, що …

а) виходи нейронної мережі формуються спеціальним образом, а ваги змінюються по алгоритму, що враховує тільки вхідні і похідні від них сигнали;

б) виходи нейронної мережі формуються самостійно, а ваги змінюються по алгоритму, що враховує тільки вхідні і похідні від них сигнали;

в) виходи нейронної мережі формуються самостійно, а ваги змінюються по алгоритму, що враховує вхідні і вихідні сигнали.

19. Який процес Ви розумієте під проблемою перенавчання?

а) це надмірно точна відповідність нейронної мережі конкретному набору навчальних прикладів, при якому мережа починає робити помилки;

б) це надмірно точна відповідність нейронної мережі конкретному набору навчальних прикладів, при якому мережа втрачає здібність до функціонування;

в) це надмірно точна відповідність нейронної мережі конкретному набору навчальних прикладів, при якому мережа втрачає здібність до узагальнення.

20. Основним принципом роботи мережи Кохонена є …

а) введення в правило навчання нейрона інформації щодо їх кількості;

б) введення в правило навчання нейрона інформації щодо його розташування;

в) введення в правило навчання нейрона інформації щодо методу навчання.

Розділ 4.

АСОЦІАТИВНІ ПРАВИЛА ТА ДЕРЕВА РІШЕНЬ

4.1. Основні поняття теорії асоціативних правил.

Пошук асоціативних правил (Association Rules) - ключова тема в інтелектуальному аналізі даних. Асоціація має місце в тому випадку, якщо декілька подій зв'язані одна з одною. Наприклад, дослідження, проведене в супермаркеті, може показати, що 65% відвідувачів, які купили кукурудзяні чіпси беруть також і «кока-колу», а за наявності знижки за такий комплект купують його в 85% випадків. Маючи в своєму розпорядженні відомості про подібну асоціацію, менеджерам легко оцінити, наскільки дієва знижка, що надається.

Пошук виявляє приховані зв'язки в, на перший погляд, ніяк незв'язаних даних. Ці зв'язки - правила. Ті, які перевищують певний поріг, вважаються цікавими. Такі правила дають можливість виконувати дії ґрунтуючись на певних шаблонах. Вони так само допомагають в прийнятті і поясненні рішень. Як і більшість методів data mining, даний метод дозволяє перетворити потенційно величезну кількість інформації в невеликий і зрозумілий набір статистичних показників [9, , 37, 46, 55, 68].

Одним з найчастіше цитованих прикладів пошуку асоціативних правил служить проблема пошуку стійких зв'язків в корзині покупця (Market Basket Problem). Проблема пошуку стійких зв'язків в корзині покупця полягає в тому, аби визначити які товари отримуються покупцями разом, так, щоб фахівці з маркетингу могли відповідним чином розмістити ці товари в магазині для підвищення об'єму продажів, і так само прийняти інші рішення, сприяючі продажам. Деякі правила, що виявляються, можуть бути тривіальними, наприклад, «покупці, які купують хліб, так само купують і масло». Інші - цікаві і екстраординарні, наприклад «покупці, які купують дитячі товари, так само

купують і пиво». Наведемо один відомий приклад. Менеджери одного з супермаркетів найбільшої міжнародної мережі Wal - Mart, завдяки технології data mining, виявили, що в п'ятницю увечері пиво чомусь особливо добре продається разом з дитячими товарами. Спочатку вони були украй здивовані: здавалося б, який може бути зв'язок між такими різними товарами? Незабаром, проте, пояснення знайшлося: багато чоловіків, повертаючись вечорами з роботи, на прохання дружин, купували своїм дітям приналежності; а в п'ятницю при цьому вони помічали, що за важкий трудовий тиждень заслужили на свою винагороду – і додавали в корзину пиво. Менеджери Wal - Mart уміло скористалися такою знахідкою, поставивши поряд з дитячими товарами одні з найдорожчих марок пива і добилися значного зростання його продажів. Інший приклад - річна економія 700 тис. дол. за рахунок впровадження data mining в мережі універсамів у Великобританії. Саме здатність виявляти цікаві правила робить пошук асоціативних правил важливим процесом, що сприяє пошуку знань (рис. 4.1).

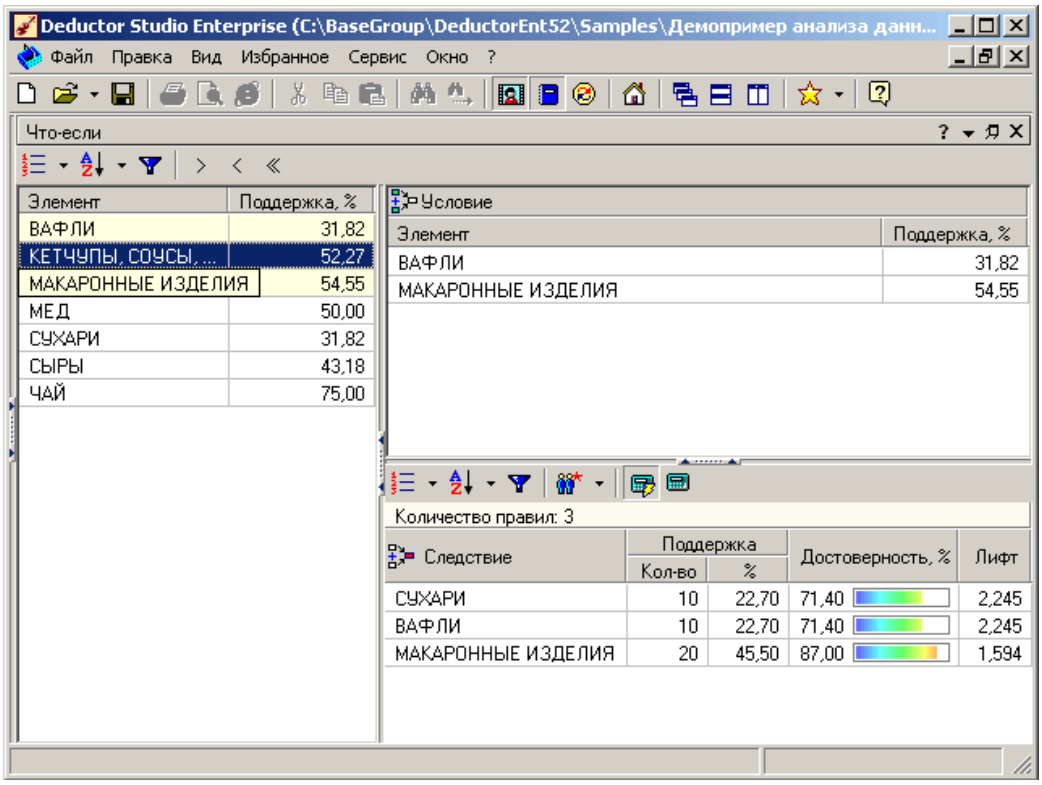


Рис. 4.1. Приклад застосування асоціативних правил.

Перший алгоритм пошуку асоціативних правил, що називався AIS, був розроблений в 1993 році співробітниками дослідницького центру IBM Almaden. З цієї піонерської роботи зріс інтерес до асоціативних правил; на середину 90-х років минулого століття припав пік дослідницьких робіт в цій області, і з тих пір щороку з'являлося по декілька нових алгоритмів. Вперше це завдання було запропоноване як пошук асоціативних правил для знаходження типових шаблонів покупок, що здійснюються в супермаркетах, тому інколи його ще називають аналізом ринкової корзини (market basket analysis). Проте сфера застосування цих алгоритмів не обмежується лише однією торгівлею. Їх також успішно застосовують і в інших областях: медицині, для аналізу відвідин веб-сторінок (Web Mining), для аналізу тексту (Text Mining), для аналізу даних по перепису населення, в аналізі і прогнозуванні збоїв телекомунікаційного устаткування і так далі. Завданням пошуку асоціативних правил не є виявлення всіх правил, оскільки частина з них відомі аналітикам, інші можуть не представляти статистичної цінності. Тому при пошуку вводяться пороги підтримки і достовірності асоціативних правил [2, 25, , 47, 56].

Нехай є база даних, що складається з купівельних транзакцій. Кожна транзакція - це набір товарів, куплених покупцем за один візит. Таку транзакцію ще називають ринковою корзиною.

Означення 1. Хай $I=(i_1,i_2,...,i_n)$ - множина (набір) товарів, званих елементами. Хай D - множина транзакцій, де кожна транзакція T - це набір елементів з I, $T\subseteq I$. Кожна транзакція є бінарним вектором, де $t[k]=1$, якщо i_k елемент присутній в транзакції, інакше $t[k]=0$. Вважатимемо, що транзакція T містить X, деякий набір елементів з I, якщо $X\subset T$. Асоціативним правилом називається імплікація $X\Rightarrow Y$, де $X\subset I$, $Y\subset I$ і $X\cap Y=0$. Правило $X\Rightarrow Y$ має підтримку s (support), якщо $s\%$ транзакцій з D, містять $X\cup Y$, $\sup p(X\Rightarrow Y)=\sup p(X\cup Y)$. Достовірність правила показує яка ймовірність того, що з X виходить Y. Правило $X\Rightarrow Y$ справедливо з достовірністю c

(confidence), якщо $c\%$ транзакцій з D, що містять X, також містять Y, $conf\ (X \Rightarrow Y) = \sup p(X \cup Y) / \sup p(X)$.

Розглянемо визначення на конкретному прикладі: «75% транзакцій, що містять хліб, також містять і молоко. 3% від загального числа всіх транзакцій містять обоє товари». 75% - це достовірність (confidence) правила, 3% це підтримка (support), або «Хліб»-«Молоко» з вірогідністю 75%. Іншими словами, метою аналізу є встановлення наступних залежностей: якщо в транзакції зустрівся деякий набір елементів X, то на підставі цього можна зробити висновок про те, що інший набір елементів Y також повинен з'явитися в цій транзакції. Встановлення таких залежностей дає нам можливість знаходити прості і інтуїтивно зрозумілі правила. Алгоритми пошуку асоціативних правил призначені для знаходження всіх правил X, Y, причому підтримка і достовірність цих правил мають бути вище за деякі наперед задані пороги, звані відповідно мінімальною підтримкою (minsupport) і мінімальною достовірністю (minconfidence). Якщо значення підтримки правила дуже велике, то в результаті роботи алгоритму будуть знайдені правила очевидні і добре відомі. Дуже низьке значення підтримки приведе до знаходження дуже великої кількості правил, які, можливо, будуть в більшій частині необґрунтованими, але і не відомими і не очевидними для аналітика. Таким чином, необхідно визначити такий інтервал, «золоту середину», який з одного боку забезпечить знаходження неочевидних правил, а з іншої - їх обґрунтованість. Якщо рівень достовірності дуже малий, то цінність правила викликає серйозні сумніви. Наприклад, правило з достовірністю в 3% лише умовно можна назвати правилом.

Пошук асоціативних правил зовсім не тривіальна задача, як може здатися на перший погляд. Одна з проблем - алгоритмічна складність при знаходженні часто зустрічаючих наборів елементів, оскільки із зростанням числа елементів в I експоненціально зростає число потенційних наборів елементів.

Існують різні типи асоціативних правил. У простій формі асоціативні правила повідомляють лише про наявність або відсутність асоціації. Логічна природа таких правил озвучена в їх назві - *булеві асоціативні правила* (Boolean Association Rule). На прикладі корзини споживача це відбувається так, «споживачі, які купують зняте молоко так само купують масло з низьким рівнем жиру», - типове булеве асоціативне правило.

Правила, які збирають декілька асоціативних правил разом, називаються *мультирівневі* або *узагальнені асоціативні правила* (Multilevel or Generalized Association Rules). При побудові таких правил елементи зазвичай групуються згідно ієрархії і пошук ведеться на найвищому концептуальному рівні. Наприклад, «споживачі, які купують молоко, так само купують хліб". В даному прикладі, «молоко і хліб» містять ієрархію різних типів і брендів, проте пошук на нижньому рівні не дозволить знайти цікаві правила.

Розглянемо цей процес детальніше. При пошуку асоціативних правил передбачалось, що всі аналізовані елементи однорідні. Повертаючись до аналізу ринкової корзини, це товари, що мають абсолютно однакові атрибути, за винятком назви. Проте не складе великих труднощів доповнити транзакцію інформацією про те, до якої товарної групи входить товар і побудувати ієрархію товарів. Наведемо приклад такого угрупування (таксономії) у вигляді ієрархічної моделі (рис. 4.2).

Нехай нам дана база транзакцій D і відомо в які групи (таксони) входять елементи. Тоді можна витягувати з даних правила, що пов'язують групи з групами, окремі елементи з групами і так далі. Наприклад, якщо покупець купив товар з групи «Безалкогольні напої», то він купить і товар з групи «Молочні продукти». Це і є узагальнені асоціативні правила.

Визначення 2. Узагальненим асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \subset I$, $Y \subset I$ і $X \cap Y = 0$ і де жоден з елементів, що входять в набір Y, не є предком жодного елементу, що входить в X. Підтримка і достовірність підраховуються так само, як і в разі асоціативних правил

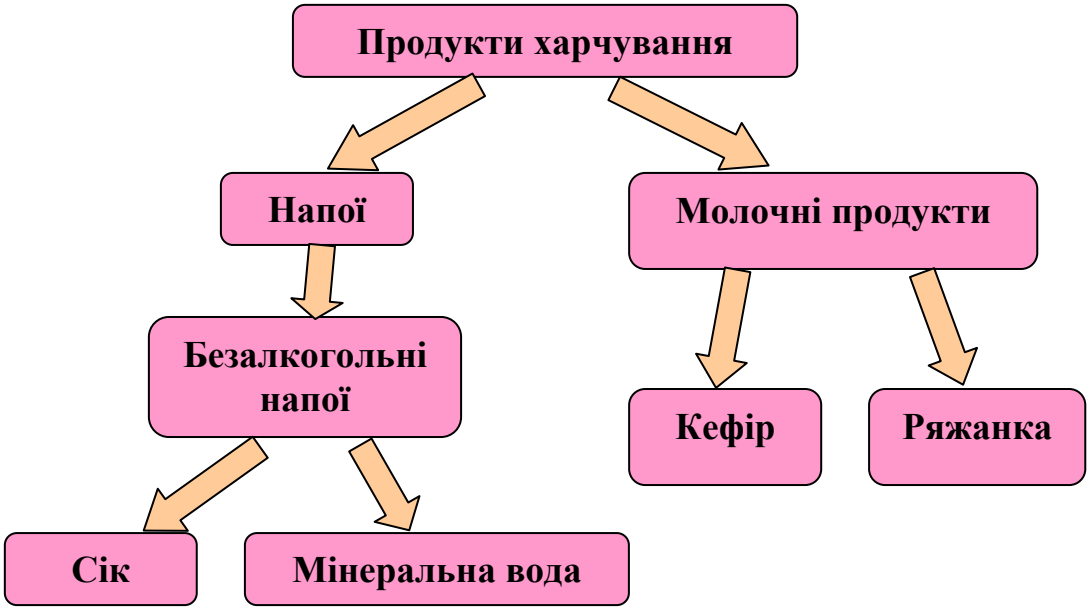


Рис. 4.2. Приклад ієрархічної моделі товарів.

Введення додаткової інформації про угрупування елементів у вигляді ієрархії дасть наступні переваги:

1. допомагає встановити асоціативні правила не лише між окремими елементами, але і між різними рівнями ієрархії (групами);

2. окремі елементи можуть мати недостатню підтримку, але в цілому група може задовольняти порогу minsupport.

Для знаходження таких правил можна використовувати будь-який з стандартних алгоритмів. Для цього кожну транзакцію потрібно доповнити всіма предками кожного елементу, що входить в транзакцію. Проте, застосування «в лоб» цих алгоритмів неминуче приведе до наступних проблем:

1. елементи на верхніх рівнях ієрархії прагнуть до значно великих значень підтримки в порівнянні з елементами на нижніх рівнях;

2. з додаванням в транзакції груп збільшилася кількість атрибутів і відповідно розмірність вхідного простору. Це ускладнює задачу, а також веде до генерації більшої кількості правил;

3. поява надлишкових правил, що перечать визначенню узагальненого асоціативного правила, наприклад, «Сік» - «Прохолодні напої». Вочевидь, що

практична цінність такого «відкриття» нульова при 100% достовірності. Отже, потрібні спеціальні оператори, що видаляють подібні надлишкові правила. Для знаходження узагальнених асоціативних правил бажане використання спеціалізованого алгоритму, який усуває вищеописані проблеми і до того ж працює в 2-5 разів швидше, ніж стандартний.

Складнішим типом правил є *кількісні асоціативні правила* (Quantitative Association Rules). Цей тип правил шукається із застосуванням кількісних (наприклад, ціна) або категоріальних (наприклад, стать) атрибутів. Наприклад, «покупці, чий вік знаходиться між 30 і 35 роками з доходом більше 75000 в рік купують машини вартістю більше 20000». Однією з ключових проблем пошуку таких правил є дискретизація числових атрибутів, яка полягає в заміні безперервного атрибуту його дискретним аналогом. Методика дискретизації впливає на цікавість виявлених правил, втрату правил унаслідок не задоволення порогам підтримки або достовірності, час роботи алгоритму. Дискретизацію числових атрибутів може проводити експерт прикладної області задаючи розбиття діапазону значень числового атрибуту на інтервали, ґрунтуючись на своїх знаннях з прикладної області.

Вищеперелічені типи правил не зачіпають той факт, що транзакції, за своєю природою, залежать від часу. Наприклад, пошук до того, як продукт був виставлений на продаж або після того, як він зник з ринку, несприятливо вплине на порогове значення підтримки (support). З врахуванням цього, введена концепція атрибутного часу життя в алгоритмах пошуку *часових асоціативних правил* (Temporal Association Rules).

Окрім описаних вище асоціативних правил існують *непрямі асоціативні правила, асоціативні правила із запереченням* та інші.

Не дивлячись на різні типи правил, алгоритм для пошуку асоціативних правил може бути в загальному вигляді розділений на два етапи:

1. пошук найбільш часто зустрічающихся наборів елементів (large (frequent) itemsets). Набір, що часто зустрічається, - це набір, в якого підтримка перевищує мінімальне значення;

2. генерація правил на основі часто зустрічающихся наборів.

З моменту появи, алгоритм Apriori найчастіше застосовується на першому кроці. Другий крок в основному характеризується достовірністю і цікавістю. Є дослідження, присвячені пошуку інших доріг генерації правил і альтернативним вимірам цікавості. Крім того, існує ряд досліджень присвячених генеруванню інших типів правил.

Алгоритм AIS. Першим алгоритмом пошуку асоціативних правил був алгоритм AIS, запропонований співробітниками дослідницького центру IBM Almaden Agrawal, Imielinski та Swami в 1993 році. З цієї роботи і почався інтерес до асоціативних правил. У алгоритмі AIS кандидати множини наборів генеруються і підраховуються «на льоту», в процесі сканування бази даних. Кожна транзакція перевірялася на наявність великих наборів, виявлених при попередньому проході. Відповідно, нові набори формувалися шляхом розширення наявних наборів.

Алгоритм SETM. Створення цього алгоритму було мотивоване бажанням використовувати мову SQL для обчислення часто зустрічающихся наборів товарів. Як і алгоритм AIS, SETM також формує кандидатів «на льоту», ґрунтуючись на перетвореннях бази даних. Аби використовувати стандартну операцію об'єднання мови SQL для формування кандидата, SETM відділяє формування кандидата від їх підрахунку.

Незручність алгоритмів AIS і SETM - зайве генерування і підрахунок дуже багатьох кандидатів, які в результаті не виявляються такими, що часто зустрічаються. Для поліпшення їх роботи був запропонований алгоритм Apriori.

Алгоритм Apriori. Робота даного алгоритму складається з декількох етапів, кожен з яких складається з наступних кроків:

- формування кандидатів;
- підрахунок кандидатів.

Формування кандидатів (candidate generation) - етап, на якому алгоритм, скануючи базу даних, створює множину i-елементних кандидатів (i - номер етапу). На цьому етапі підтримка кандидатів не розраховується. *Підрахунок*

кандидатів (candidate counting) - етап, на якому обчислюється підтримка кожного i-елементного кандидата. Тут же здійснюється відсікання кандидатів, підтримка яких менше мінімуму, встановленого користувачем (minsup). I-елементні набори, що залишилися, називаємо такими, що часто зустрічаються.

Для того, щоб було можливо застосувати алгоритм, необхідно провести передобробку даних: по-перше, привести всі дані до бінарного вигляду; по-друге, змінити структуру даних. Звичайний вигляд бази даних транзакцій представлен на рис. 4.3, а нормалізованої – на рис. 4.4.

Номер транзакції	**Найменування елементу**	**Кількість**
1001	A	2
1001	D	3
1001	E	1
1002	A	2
1002	F	1
1003	B	2
1003	A	2
1003	C	2
…	…	…

Рис. 4.3. Фрагмент основної бази даних.

TID	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**H**	**I**	**K**	**...**
1001	1	0	0	1	1	0	0	0	0	0	…
1002	1	0	0	0	0	1	0	0	0	0	…
1003	1	1	1	0	0	0	0	0	1	0	…

Рис. 4.4. Фрагмент нормалізованої бази даних.

Кількість стовпців в таблиці дорівнює кількості елементів, присутніх в множині транзакцій D. Кожен запис відповідає транзакції, де у відповідному стовпці стоїть 1, якщо елемент присутній в транзакції, і 0 інакше. Відмітимо, що вигляд таблиці може бути відмінним від приведеного на рис. 4.3. Головне, аби дані

були перетворені до нормалізованого вигляду, інакше алгоритм не може бути застосований. Більш того, як видно з таблиці, всі елементи впорядковані в алфавітному порядку (якщо це числа, вони мають бути впорядковані в числовому порядку).

Як було відмічено раніше, такі алгоритми працюють в два етапи. На першому кроці необхідно знайти набори елементів, що часто зустрічаються, а потім, на другому, витягувати з них правила. Кількість елементів в наборі називатимемо розміром набору, а набір, що складається з k елементів, – k-елементним набором.

Виявлення часто зустрічающихся наборів елементів – операція, що вимагає багато обчислювальних ресурсів і, відповідно, часу. Примітивний підхід до рішення даної задачі – простий перебір всіх можливих наборів елементів. Це зажадає 2^I операцій, де I – кількість елементів. Apriori використовує одну з властивостей підтримки, яке свідчить: підтримка будь-якого набору елементів не може перевищувати мінімальної підтримки будь-якої з його підмножин. Наприклад, підтримка 3-елементного набору {Хліб, Масло, Молоко} буде завжди менша або дорівнює підтримці 2-елементних наборів {Хліб, Масло} {Хліб, Молоко} {Масло, Молоко}. Річ у тому, що будь-яка транзакція, що містить {Хліб, Масло, Молоко}, також повинна містити {Хліб, Масло}, {Хліб, Молоко}, {Масло, Молоко}, причому зворотне не вірно. Ця властивість носить назву *антимонотонності* і служить для зниження розмірності простору пошуку. Не май ми в наявності такої властивості, знаходження багатоелементних наборів було б практично нездійсненним завданням у зв'язку з експоненціальним зростанням обчислень.

Властивості антимонотонності можна дати і інше формулювання: із зростанням розміру набору елементів підтримка зменшується або залишається такою же. Зі всього вищесказаного виходить, що будь-який k-елементний набір буде таким, що часто зустрічається тоді і лише тоді, коли все його (k-1)-елементні підмножини будуть такими, що часто зустрічаються.

Всі можливі набори елементів з I можна представити у вигляді грат, що починаються з пустої множини, потім на 1 рівні 1-елементні набори, на 2 рівні – 2-елементні і так далі На k рівні представлені k-елементні набори, пов'язані зі всіма своїми (k-1)-елементними підмножинами.

Розглянемо рис 4.5, що ілюструє набір елементів I – {A, B, C, D}. Вважатимемо, що набір з елементів {A, B} має підтримку нижче заданого порогу і, відповідно, не є таким, що часто зустрічається. Тоді, згідно властивості антимонотонності, всі його супермножини також не є такими, що часто зустрічаються і відкидається. Вся ця гілка, починаючи з {A, B}, виділена фоном. Використання цієї евристики дозволяє істотно скоротити простір пошуку.

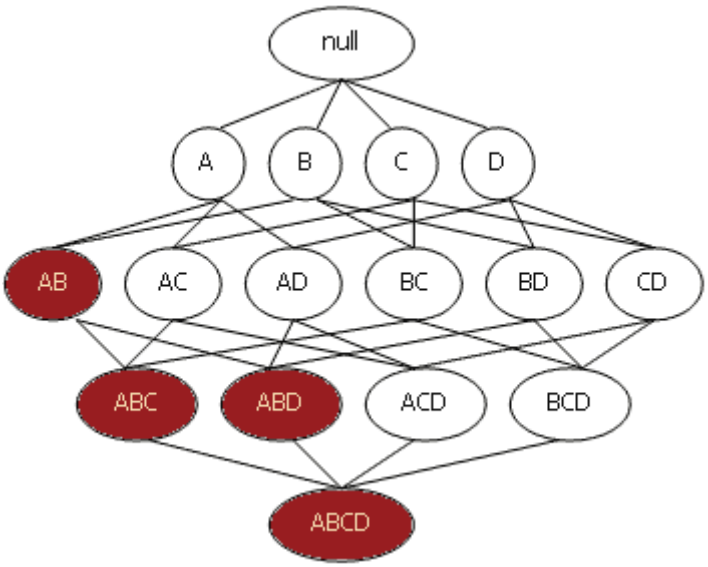


Рис. 4.5. Механізм скорочення простору пошуку.

На першому кроці алгоритму підраховуються 1-елементні набори, що часто зустрічаються. Для цього необхідно пройтися по всьому набору даних і підрахувати для них підтримку, тобто скільки разів зустрічається в базі. Наступні кроки складатимуться з двох частин: генерації потенційно часто зустрічающихся наборів елементів (їх називають кандидатами) і підрахунку підтримки для кандидатів. Описаний вище алгоритм можна записати у вигляді наступного псевдокоду:

1. F_1 = {1-елементні набори, що часто зустрічаються}
2. Для ($k = 2$; $F_{k-1} \lozenge \varnothing$; $k++$) {
3. C_k = Apriorigen(F_{k-1}) // генерація кандидатів
4. Для всіх транзакцій $t \in T$ {
5. C_t = subset(C_k, t) // видалення надлишкових правил
6. Для всіх кандидатів $c \in C_t$
7. c.count ++
8. }
9. $F_k = \{c \in C_k | c.count \geq \min sup\ port\}$ // відбір кандидатів
10. }
11. Результат $\cup F_k$

Опишемо функцію генерації кандидатів. Генерація кандидатів також складатиметься з двох кроків.

1. Об'єднання. Кожен кандидат C_k формуватиметься шляхом розширення часто зустрічающегося набору розміру $(k-1)$, шляхом додаванням елементу з іншого $(k-1)$-елементного набору. Приведемо алгоритм функції Apriorigen у вигляді невеликого SQL-подібного запиту.

   insert into C_k

   select p.item$_1$, p.item$_2$,…, p.item$_{k-1}$, q.item$_{k-1}$

   From F_{k-1} p, F_{k-1} q

   where p.item$_1$=q.item$_1$, p.item$_2$=q.item$_2$,…, p.item$_{k-1}$<q.item$_{k-1}$
2. Видалення надлишкових правил. На підставі властивості анти-монотонності, слід видалити всі набори $c \in C_k$ якщо хоч би одна з його $(k-1)$ підмножин не є такою, що часто зустрічається.

Після генерації кандидатів наступним завданням є підрахунок підтримки для кожного кандидата. Вочевидь, що кількість кандидатів може бути дуже великою і потрібний ефективний спосіб підрахунку. Найтривіальніший спосіб – порівняти кожну транзакцію з кожним кандидатом.

Але це далеко не краще рішення. Набагато швидше і ефективніше використовувати підхід, заснований на зберіганні кандидатів в хеш-дереві. Внутрішні вузли дерева містять хеш-таблиці з вказателями на нащадків, а листя – на кандидатів. Це дерево нами буде застосоване для швидкого підрахунку підтримки для кандидатів.

Хеш-дерево будується кожного разу, коли формуються кандидати. Первинне дерево складається лише з кореня, який є листом, і не містить жодних кандидатів-наборів. Кожного разу коли формується новий кандидат, він заноситься в корінь дерева і так до тих пір, поки кількість кандидатів в корені-листі не перевищить деякого порогу. Як тільки кількість кандидатів стає більше порогу, корінь перетвориться в хеш-таблицю, тобто стає внутрішнім вузлом, і для нього створюються нащадки-листя. І всі приклади розподіляються по вузлах-нащадках згідно хеш-значенням елементів, що входять в набір, і так далі Кожен новий кандидат хеширується на внутрішніх вузлах, поки він не досягне першого вузла-листа, де він і зберігатиметься, поки кількість наборів знову ж таки не перевищить порогу.

За допомогою хеш-дерева легко підрахувати підтримку для кожного кандидата. Для цього потрібно «пропустити» кожну транзакцію через дерево і збільшити лічильники для тих кандидатів, чиї елементи також містяться і в транзакції, тобто $C_k \cap T_i = C_k$. На кореневому рівні хеш-функція застосовується до кожного елементу з транзакції. Далі, на другому рівні, хеш-функція застосовується до других елементів і так далі На k-рівні хеширується k-елемент. І так до тих пір, поки не досягнемо листа. Якщо кандидат, що зберігається в листі, є підмножиною даної транзакції, тоді збільшуємо лічильник підтримки цього кандидата на одиницю. Після того, як кожна транзакція з вихідного набору даних «пропущена» через дерево, можна перевірити чи задовольняють значення підтримки кандидатів мінімальному порогу. Кандидати, для яких ця умова виконується, переносяться в розряд тих, що часто зустрічаються. Крім того, слід запам'ятати і підтримку набору, вона нам знадобиться при витяганні правил.

Після того, як знайдені всі набори елементів, що часто зустрічаються, можна приступити безпосередньо до генерації правил. Витягання правил – менш трудомістке завдання. По-перше, для підрахунку достовірності правила досить знати підтримку самого набору і множини, лежачої в умові правила. Наприклад, є часто зустрічающийся набір {A, B, C} і потрібно підрахувати достовірність для правила $AB \Rightarrow C$. Підтримка самого набору нам відома, але і його множина {A, B}, яка лежить в умові правила, також є такою, що часто зустрічається через властивість антимонотонності. Це означає, що його підтримка нам відома. Тоді ми легко зможемо підрахувати достовірність. Це позбавляє нас від небажаного перегляду бази транзакцій, що було б потрібно в тому разі якби ця підтримка була невідома.

Аби витягувати правило з набору F F, що часто зустрічається, слід знайти всі його непусті підмножини. І для кожної підмножини s ми зможемо сформулювати правило $s \Rightarrow (F - s)$, якщо достовірність правила $conf\ (s \Rightarrow (F - s) = \sup p(F) / \sup p(s)$ не менше порогу minconf. Відмітимо, що чисельник залишається постійним. Тоді достовірність має мінімальне значення, якщо знаменник має максимальне значення, а це відбувається у тому випадку, коли в умові правила є набір, що складається з одного елементу. Вся супермножина даної множини має меншу або рівнішу підтримку і, відповідно, більше значення достовірності. Ця властивість може бути використана при витяганні правил. Якщо ми почнемо витягувати правила, розглядаючи спочатку лише один елемент в умові правила, і це правило має необхідну підтримку, тоді всі правила, де в умові стоїть супермножина цього елементу, також мають значення достовірності вище заданого порогу. Наприклад, якщо правило $A \Rightarrow BCDE$ задовольняє мінімальному порогу достовірності minconf, тоді $AB \Rightarrow CDE$ також задовольняє. Для того, щоб витягувати всі правила використовується рекурсивна процедура. Важливе зауваження: будь-яке правило, складене з набору, що часто зустрічається, повинне містити всі елементи набору. Наприклад, якщо набір складається з елементів {A, B, C}, то правило $A \Rightarrow B$ не повинно розглядатися.

Залежно від розміру щонайдовшого часто зустрічающегося набору, алгоритм Apriori сканує базу даних певну кількість разів. Різновиди алгоритму Apriori, що є його оптимізацією, запропоновані для скорочення кількості сканувань бази даних, кількості наборів-кандидатів або того і іншого. Були запропоновані наступні різновиди алгоритму Apriori: AprioriTid і AprioriHybrid.

Алгоритм AprioriTid. Цей алгоритм генерує набори елементів використовуючи лише великі набори, знайдені на попередньому кроці, без повторного розгляду транзакцій. AprioriTid покращує Apriori за рахунок того, що використовує базу даних лише при першому проході. При підрахунках на подальших кроках використовуються лише дані, створені при першому проході і такі, що мають набагато менший розмір, чим вихідна база даних. Це приводить до колосального зростання продуктивності - в три рази швидше, ніж AIS і в чотири рази швидше, ніж SETM.

Алгоритм AprioriHybrid. Аналіз часу роботи алгоритмів Apriori і AprioriTid показує, що в ранішніх проходах Apriori досягає більшого успіху, чим AprioriTid; проте AprioriTid працює краще Apriori в пізніших проходах. Крім того, вони використовують одну і ту ж процедуру формування наборів-кандидатів. Заснований на цьому спостереженні, алгоритм AprioriHybrid запропонований, аби об'єднати кращі властивості алгоритмів Apriori і AprioriTid. AprioriHybrid використовує алгоритм Apriori в початкових проходах і переходить до алгоритму AprioriTid, коли очікується, що закодований набір первинної множини в кінці проходу відповідатиме можливостям пам'яті. Проте, перемикання від Apriori до AprioriTid вимагає залучення додаткових ресурсів.

Алгоритм DHP. Проблема Apriori в тому, що він генерує надто багато двоелементних наборів, які не є частими. J. Park, M. Chen and P. Yu був запропонований механізм *прямого хешування і обрізання даних* (direct hashing and pruning, DHP), які дозволяють зменшити кількість наборів кандидатів за рахунок відкидання k наборів з хеш-таблиці, якщо їх значення не досягає величини мінімальної підтримки. В основі роботи алгоритму - імовірнісний підрахунок наборів-кандидатів, здійснюваний для скорочення числа

підраховуваних кандидатів на кожному етапі виконання алгоритму Apriori. Скорочення забезпечується за рахунок того, що кожен з k-елементних наборів-кандидатів окрім кроку скорочення проходить крок хешування. У алгоритмі на $k-1$ етапі під час вибору кандидата створюється хеш-таблиця. Кожен запис хеш-таблиці є лічильником всієї підтримки k-елементних наборів, які відповідають цьому запису в хеш-таблиці. Алгоритм використовує цю інформацію на етапі k для скорочення множини k-елементних наборів-кандидатів. Після скорочення підмножини, як це відбувається в Apriori, алгоритм може видалити набір-кандидат, якщо його значення в хеш-таблиці менше порогового значення, встановленого для забезпечення. Як показали порівняльні експерименти, ця чудова фільтруюча особливість дозволяє DHP завершити вже всі обчислення в тому момент, коли Apriori все ще знаходиться на другому кроці.

Подальші зусилля по поліпшенню алгоритму Apriori пов'язані з розпаралелюванням алгоритму. Було запропоновано 3 паралельні алгоритми, що дозволяють прискорити пошук часто використовуваних наборів елементів. Алгоритм *Count Distribution* (CD) мінімізує обмін даними (communication) за рахунок проведення дублюючих обчислень. Алгоритм *Data Distribution* (DD) використовує основну пам'ять системи для передачі локальних даних на інші вузли системи. Алгоритм *Candidate Distribution* - це збалансований по навантаженню алгоритм, що знижує синхронізацію між процесорами і сегментує базу даних на основі різних шаблонів транзакцій. Ці паралельні алгоритми були протестовані і алгоритм CD мав найкращу продуктивність в порівнянні з алгоритмом Apriori. Його накладні витрати не перевищували 7.5% в порівнянні з Apriori.

Масштабованість - інша важлива область дослідження асоціативних алгоритмів, оскільки бази даних з кожним днем стають все більше і більше. Алгоритми мають бути здатні масштабуватися, аби обробити великі набори даних. Ґрунтуючись на цій ідеї була зроблена спроба зробити DD і CD алгоритми масштабованими. Були, відповідно, створені алгоритми *Intelligent*

Data Distribution (IDD) і *Hybrid Distribution* (HD). IDD направлений на зниження накладних витрат, а також на взаємодію і виключення зайвих обчислень за рахунок використання агрегатної пам'яті (aggregate memory) для розділення кандидатів і ефективного переміщення даних. HD був покращен в порівнянні з IDD за рахунок динамічного розділення масиву кандидатів для кращої підтримки збалансованого навантаження. Експерименти показали, що час відгуку IDD в 4.4 разу менше, ніж DD на 32-процесорній системі, а HD виявився кращим CD на 9.5% при роботі на 128-процесорах.

Інше дослідження масштабування засновано на введенні *полегшених структур даних* Segment Support Map (SSM), які зменшують кількість наборів-кандадитів, необхідних в розрахунках. SSM містить значення підтримки для одинарних наборів. Шляхом підсумовування значень підтримки для одинарних наборів оцінюються верхні кордони для k-елементних наборів. Стосовно Apriori, зусилля по генеруванню одноелементних наборів можуть бути заощаджені простим пошуком тих одноелементних наборів в SSM, в яких підтримка вище мінімального порогу. Більш того, одноелементні набори, в яких підтримка менше порогового значення, взагалі виключаються, що дозволяє понизити кількість наборів більш високого рівня.

Інша область досліджень, що направлена на поліпшення Apriori і базується на використанні оригінальних структур даних, зв'язана із застосуванням *дерев часто зустрічающихся елементів* (frequent pattern tree, or FP-tree). FP-дерево зберігає інформацію про часто використовувані шаблони (patterns). Метод, заснований на FP-деревах, називається *FP-growth* (метод вирощування найбільш популярних шаблонів). У ньому, замість генерації наборів-кандидатів, пропонується шукати часто використовувані шаблони. FP-growth метод на порядок кращий ніж Apriori і також краще масштабується.

Перехід від ітераційних методів, подібно Apriori і DHP, до інноваційного використання структур даних, типа SSM і FP-дерев, привів до того, що пошук часто зустрічающихся наборів елементів став більш масштабованим і ефективним. Необхідність в швидших і більш масштабованих

алгоритмах як і раніше існує, оскільки бази даних стають все більше і більше з кожним днем. Дослідження, присвячені розподіленим алгоритмам для пошуку наборів, що часто зустрічаються, потребують тим більше уваги, чим більше баз даних інтегрується разом. Така інтеграція переводить задачу пошуку на новий рівень, що вимагає гнучкіших алгоритмів, які беруть до уваги різне представлення однакових даних. Наприклад, zip-коди можуть бути представлені рядками або числами, тому дані не можуть бути нормалізовані до проведення пошуку. Можна чекати збільшення кількості досліджень, пов'язаних з паралельними алгоритмами, оскільки мережеві обчислення набирають все більшу популярність.

Дослідження, присвячені генерації правил, в основному фокусуються на нових алгоритмах (таких, що дають більше типів асоціативних правил) і цікавості правил. Нові алгоритми в основному використовують нові стратегії, такі, як *паралельні обчислення* і *Evolutionary Algorithm* (EA). Нові типи правил додають розмірність і якість в традиційні булеві типи правил. Приклади - кількісні асоціативні правила і часові асоціативні правила. Нові критерії цікавості мають тенденцію бути об'єктивнішими, ніж підтримка і достовірність. Більшість асоціативних правил згенерували шляхом підрахунку кількості транзакцій, в яких правило зустрічається в базі даних - достовірності. Інтуїтивно ясно, що набори часто зустрічающихся елементів можна розділити на частини і проводити підрахунки в паралель.

Останніми роками, алгоритм EA був широко схвалений в багатьох наукових областях. EA запозичує механізми біологічної еволюції і застосовує їх для вирішення проблем. Особливо добре він використовується для пошукових і оптимізаційних проблем, так що проблема пошуку асоціативних правил якраз підходить для вирішення цим алгоритмом. В комп’ютерних системах інтелектуального аналізу даних EA був використаний для генерації асоціативних правил. Популяція наборів, що часто зустрічаються, бралася як початкова популяція. Використання EA включає пересічення і мутацію цих наборів. Популяція еволюціонує так, що в ній залишаються лише набори, що

щонайкраще задовольняють функції придатності (fitness function). Коли в популяції залишається бажана кількість часто зустрічающихся наборів, алгоритм зупиняється. Це оригінальна дорога генерування та пошуку асоціативних правил застосовується для пересічних інтервалів в різних наборах. Наприклад, один набір може мати інтервал [10, 20], а інший [15, 25]. Тут є разючий контраст в порівнянні з іншими технологіями, які розділяють атрибути на інтервали, що не перетинаються. Правила, які потрапляють в пересічення двох інтервалів можуть бути не знайдені із-за втрати інформації. Алгоритм EA дозволяє знайти нові правила.

При розробці сучасних систем data mining були проведені також дослідження відносно типів асоціативних правил. На першому етапі еволюції таких систем домінували булеві асоціативні правила. Пізніше, фокус змістився на кількісні асоціативні правила. Кількісні асоціативні правила - це правила над кількісними і категоріальними атрибутами, подібно зросту і сімейному стану. Пошук таких правил включає розділення діапазону значень атрибутів на частини, при цьому інформація при такому розділенні може втрачатися. Був запропонований алгоритм пошуку кількісних асоціативних правил, заснований на Apriori. У ньому вводиться поняття часткової завершеності (partial completeness) для вимірювання втрат інформації при розділенні і інтерес, «більш ніж очікуваний» («greater than expected» interest) як міра цікавості. Часткова завершеність прямо пропорційна втратам інформації. Набуваючи мінімального значення підтримки і часткової завершеності від користувача, система може визначити необхідну кількість розділень. Кількісне правило цікаве лише, якщо воно має «більш ніж очікувану» підтримку і достовірність, вказану користувачем.

Розмірність часу одна з тих характеристик, що існує для всіх транзакцій. Тому, вона має бути включена в процес пошуку наборів, що часто зустрічаються, особливо, коли не всі елементи існують протягом всього періоду, який охоплюють зібрані дані. Існує концепція часу шляхом обмеження пошуку часто зустрічающихся наборів згідно часу життя елементів. Також

введена концепція тимчасової підтримки на додаток до звичайної підтримки і достовірності. Час життя елементу визначається як перший і останній момент часу появи елементу в базі даних. Тимчасова підтримка - це ширина мінімального інтервалу. Таким чином, правило розглядається в тому випадку, якщо воно має досить високі значення підтримки і тимчасової підтримки. Побічним продуктом даного підходу є те, що старі або застарілі набори елементів, можуть бути відкинуті.

Будь-яка асоціація, знайдена вказаними вище алгоритмами, може бути правилом. Якість цих правил вимірюється достовірністю. Проте, лише ті правила, достовірність яких вище певного рівня, цікаві і заслуговують на увагу. Більшість алгоритмів визначають цікавість в термінах величин підтримки і достовірності, вказаних користувачем. Проблема в тому, що ці алгоритми покладаються на те, що користувачі зможуть задати відповідні значення. Новий алгоритм, названий APACS2, не вимагає від користувача яких-небудь припущень, але використовує об'єктивно цікаві значення, названі регульованою різницею (adjusted difference). Більш того, APACS2 може виявляти як позитивні, так і негативні асоціативні правила. Також представлена нова концепція зв'язаності (relatedness) як альтернативного підходу для визначення цікавості. APACS2 використовує регульовану різницю як об'єктивну міру цікавості. Регульована різниця визначається в термінах стандартизованої різниці (standardized difference) і оцінки максимальної правдоподібності (maximum likelihood estimate). Якщо величина регульованої різниці вища, ніж 1.96, тобто 95% нормального розподілу, асоціація розглядається як істотно різна і, отже, цікава. Якщо регульована різниця позитивна, це означає, що правило ймовірно, і навпаки. Направлена природа регульованої різниці дає нову розмірність пошуку асоціативних правил.

Цікавість може бути суб'єктивною, якщо застосовуються значення підтримки і достовірності, або об'єктивною, якщо використовується регульована різниця. Протилежна до концепції - зв'язність (relatedness), - була введена для того, щоб досліджувати співвідношення між двома елементами, на

основі частоти їх спільної появи в транзакціях і контексту. Зв'язність призначена для використання замість цікавості для кількісних правил. В зв'язності є три компоненти: середня передбачаюча здатність присутності одного елементу при присутності іншого, інтенсивність появи пар елементів у присутності інших елементів; заміщаємість іншого елементу для елементу в парі елементів. Ці три заходи складають силу зв'язності в термінах частоти правила по відношенню до інших елементів.

Дослідження генерації правил починалося з простих булевих типів з використанням суб'єктивних значень підтримки і достовірності. Нині воно включає різних типів правил з об'єктивними характеристиками, подібно до регульованих різниць і зв'язності. Нові дослідження додали кількісні і якісні аспекту генерації правил. Якість покращена за рахунок використання об'єктивних властивостей якості правил. Кількість збільшена за рахунок вживання оригінальних методологій, що дозволяють здійснювати пошук правил в пересічних інтервалах і знаходити негативні асоціації. З появою все нових і нових типів даних, наприклад, мультимедійних даних, більше досліджень стало проводитися для визначення нових типів асоціативних правил. Це дозволить проаналізувати нові шаблони поведінки і зробити нові передбачення. Здатність перетворювати мультимедійні дані в raw-формат (формат необроблених даних) може знайти практичне застосування в медицині, національній безпеці та інших областях.

Технології пошуку асоціативних правил вивчається и застосовуються понад двадцять років. Безліч фундаментальних досліджень вже проведена. Велику увагу було сфокусовано на продуктивності і масштабованості алгоритмів, але недостатньо уваги було приділено якості (цікавості) правил, що генерувалися. У наступні десятиліття, увага сфокусується на практичному впровадженні досліджень в різних сферах нашого життя, наприклад, генетичних дослідженнях, медицині, національній безпеці і тому подібне. У міру інтеграції баз даних і із збільшенням в них наборів даних, пошуку алгоритмів, що дозволяють сканувати більше і швидше, приділятиметься

менше увага. Навпаки, розподіленим алгоритмам, що дозволяють розділяти навантаження в обчислювальних мережах, уваги буде приділятися все більше. Чим більше даних створюється поза традиційними базами даних, data mining і пошук правил переростуть сканування таблиць баз даних і почнуть працювати з даними в raw-форматі, наприклад, з відеокліпами. Питання продуктивності і масштабованості стануть ще важливіші. Нові типи даних можуть бути необхідні для просування нових типів аналізу даних. Так само стають більш потрібнішими об'єктивні заходи цікавості, аби експерти предметних областей могли уникнути маніпуляцій з критеріями пошуку правил при здобутті бажаних результатів. Пошук асоціативних правил залишатиметься благодатною темою для досліджень. На основі досліджень і розвитку в останнє десятиліття, в ньому сформувалися і виділилися області, які повинні колосально змінитися в найближче десятиліття.

4.2. Програмні засоби пошуку асоціативних правил.

За допомогою алгоритмів виявлення асоціативних правил можна вирішувати чималий спектр практичних завдань. Саме тому ринок програмних продуктів, що реалізовують ці технології, досить представницький та різнорідний. Практично кожна відома компанія в тому або іншому вигляді використовує технології пошуку асоціативних правил в своїх програмних продуктах. Зупинимося на деяких програмних рішеннях, що отримали найбільшу популярність [2, 25, 70, 77].

Пакет Deductor. Розглянемо приклад рішення задачі пошуку асоціативних правил. Вважатимемо, що існує транзакційна база даних. Необхідно знайти набори товарів, що зустрічаються найбільш часто, і набір асоціативних правил з певними границями значень підтримки і довіри.

Процес побудови асоціативних правил виконаємо в аналітичному пакеті Deductor. Транзакційна база даних, яка містить в кожній записі номер чека і

товар, придбаний по цьому чеку, має формат MS Excel. Спершу імпортуємо дані з файлу MS Excel в середовище Deductor. Для номера транзакції (зазвичай в базі даних - це поле «номер чека») вказуємо тип «ідентифікатор транзакції (ID)», а для найменувань товару - тип «елемент». Результат імпорту бази даних з файлу MS Excel в середовище Deductor відображені на рисунку 4.6.

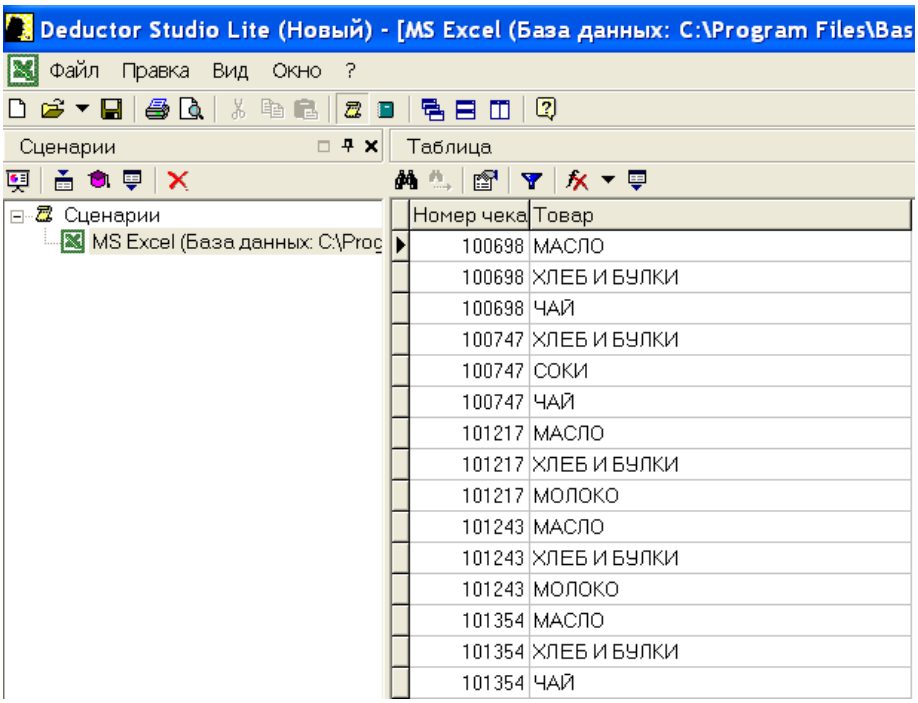


Рис. 4.6. Транзакційна база даних, яка імпортована з файлу MS Excel.

За допомогою майстра обробки вибираємо метод «Асоціативні правила». На другому кроці майстра перевіряємо призначення вихідних стовпців даних, вони повинні мати тип «ID» і «елемент». На третьому кроці, проілюстрованому на рис. 4.7, необхідно настроїти параметри пошуку правил, тобто встановити мінімальні і максимальні характеристики підтримки і достовірності. Це найбільш «відповідальний» момент формування набору правил. Вибір можна зробити на основі яких-небудь міркувань, наявного досвіду аналізу подібних даних, інтуїції або ж визначити в ході експериментів.

Встановимо наступні границі для параметрів пошуку: мінімальний і максимальний рівень підтримки дорівнюють 20% і 60% відповідно, мінімальний і максимальний рівень значення достовірності дорівнюють 40% і

90% відповідно. Ці значення були виявлені в ході проведення декількох експериментів, і виявилось, що саме при таких значеннях формується необхідний набір правил. При застосуванні деяких інших значень, наприклад, рівня підтримки від 30% до 50%, набір правил не формується, оскільки жодне правило по параметрах підтримки не входить в цей інтервал.

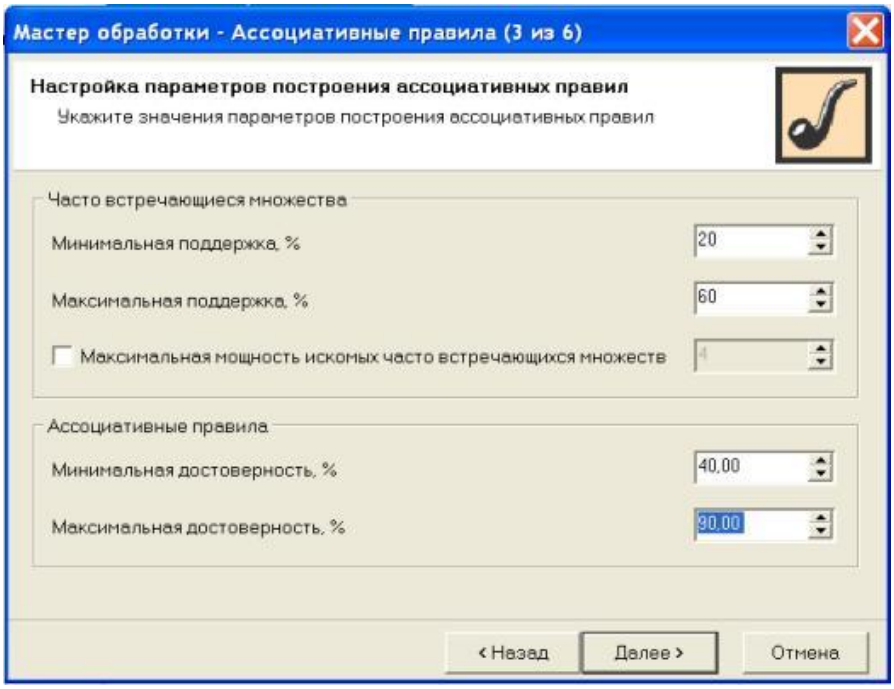


Рис. 4.7. Налаштування параметрів побудови асоціативних правил.

На наступному кроці майстра запускається процес пошуку асоціативних правил. В результаті бачимо інформацію про кількість множин і знайдених правил у вигляді гістограми розподілу часто зустрічающихся множин по їх потужності. Даний процес проілюстрований на рис. 4.8.

Ми бачимо, що кількість сформованих множин дорівнює тринадцяти - це популярні набори, кількість сформованих правил - п'ятнадцять. На наступному кроці для перегляду отриманих результатів пропонується вибрати візуалізацію із списку. Виберемо такі: «Популярні набори», «Правила», «Дерево правил», «Що-якщо».

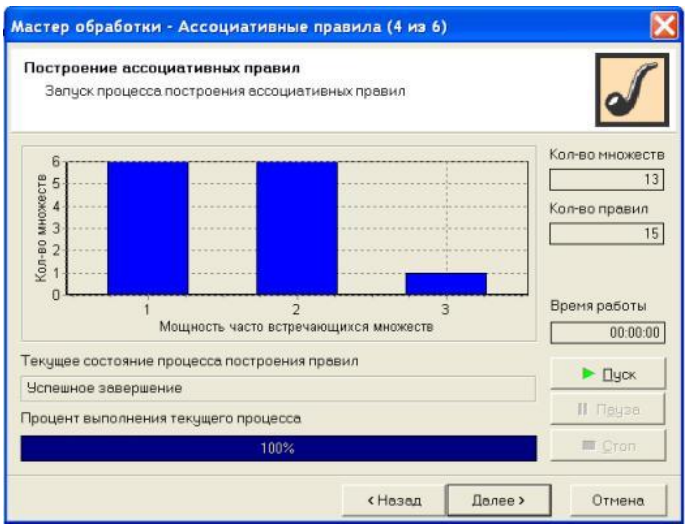


Рис. 4.8. Процес побудови асоціативних правил.

Візуалізація «Популярні набори». Популярні набори або набори, що часто зустрічаються, - це набори, що складаються з одного або декількох товарів, які в транзакціях найчастіше зустрічаються одночасно. Характеристикою, наскільки часто набір зустрічається в аналізованому наборі даних, є підтримка. Популярні набори нашого набору даних, знайдені при заданих параметрах, приведені на рис 4.9. Є можливість відсортувати дану таблицю згідно різним її характеристикам. Для визначення найбільш популярних товарів і їх наборів зручно відсортувати її по рівню підтримки. Таким чином, ми бачимо, що найбільшою популярністю користуються такі товари: хліб і булки, масло, соки.

N	Множество	Поддержка	
		%	Кол-во
6	ХЛЕБ И БУЛКИ	54,55	24
3	МАСЛО	52,27	23
5	СОКИ	50,00	22
10	МАСЛО **И** ХЛЕБ И БУЛКИ	45,45	20
4	МОЛОКО	43,18	19
2	КЕФИР	31,82	14
1	ЙОГУРТЫ	31,82	14
12	СОКИ **И** ХЛЕБ И БУЛКИ	22,73	10
11	МОЛОКО **И** ХЛЕБ И БУЛКИ	22,73	10
8	МАСЛО **И** МОЛОКО	22,73	10

7	ЙОГУРТЫ **И** КЕФИР	22,73	10
13	МАСЛО **И** МОЛОКО **И** ХЛЕБ И БУЛКИ	20,45	9
9	МАСЛО **И** СОКИ	20,45	9

Рис. 4.9. Візуалізація «Популярні набори».

Візуалізація «Правила». Правила в даній візуалізації розміщені у вигляді списку. Кожне правило, представлене як «умова-наслідок», характеризується значенням підтримки в абсолютному і процентному вираженні, а також достовірністю. Таким чином, аналітик бачить поведінку покупців, описану у вигляді набору правил. Набор правил для вирішуваної нами задачі приведений на рис. 4.10. Наприклад, перше правило говорить про те, що якщо споживач купив йогурт, то з достовірністю або вірогідністю 71% він купить також кефір. Ця інформація корисна з різних точок зору. Вона, наприклад, допомагає вирішити завдання розташування товарів в магазині.

N	**Условие**	**Следствие**	**Поддержка**		**Достоверность, %**
			%	**Кол-во**	
1	ЙОГУРТЫ	КЕФИР	22,73	10	71,43
2	КЕФИР	ЙОГУРТЫ	22,73	10	71,43
3	МАСЛО	МОЛОКО	22,73	10	43,48
4	МОЛОКО	МАСЛО	22,73	10	52,63
5	СОКИ	МАСЛО	20,45	9	40,91
6	МАСЛО	ХЛЕБ И БУЛКИ	45,45	20	86,96
7	ХЛЕБ И БУЛКИ	МАСЛО	45,45	20	83,33
8	МОЛОКО	ХЛЕБ И БУЛКИ	22,73	10	52,63
9	ХЛЕБ И БУЛКИ	МОЛОКО	22,73	10	41,67
10	СОКИ	ХЛЕБ И БУЛКИ	22,73	10	45,45
11	ХЛЕБ И БУЛКИ	СОКИ	22,73	10	41,67
12	МАСЛО **И** МОЛОКО	ХЛЕБ И БУЛКИ	20,45	9	90,00
13	МАСЛО **И** ХЛЕБ И БУЛКИ	МОЛОКО	20,45	9	45,00
14	МОЛОКО **И** ХЛЕБ И БУЛКИ	МАСЛО	20,45	9	90,00
15	МОЛОКО	МАСЛО **И** ХЛЕБ И БУЛКИ	20,45	9	47,37

Рис. 4.10. Візуалізація «Правила».

При великій кількості знайдених правил і широкому асортименті товарів аналізувати отримані правила досить складний. Для зручності аналізу таких наборів правил пропонується візуалізація «Дерево правил» і «Що-якщо».

Візуалізація «Дерево правил» є дворівневим деревом, яке може бути побудоване по двох критеріях: по умові і по наслідку. Якщо дерево побудоване по умові, то зверху списку відображується умова правила, а список, що додається до даної умови, складається з його наслідків. При виборі певної умови, в правій частині візуалізації відображуються наслідки умови, рівень підтримки і достовірності. В разі побудови дерева по наслідку, зверху списку відображується наслідок правила, а список складається з його умов. При виборі певного наслідку, в правій частині візуалізації ми бачимо умови цього правила з вказівкою рівня підтримки і достовірності.

Візуалізація «Що-якщо» зручний, якщо нам необхідно відповісти на питання, які наслідки можуть вийти з даної умови. Наприклад, вибравши умову «МОЛОКО», в лівій частині екрану отримуємо три наслідки «МАСЛО», «ХЛІБ І БУЛКИ, «МАСЛО І ХЛІБ І БУЛКИ», для яких вказані рівень підтримки і достовірності. Ця візуалізація представлена на рис. 4.11.

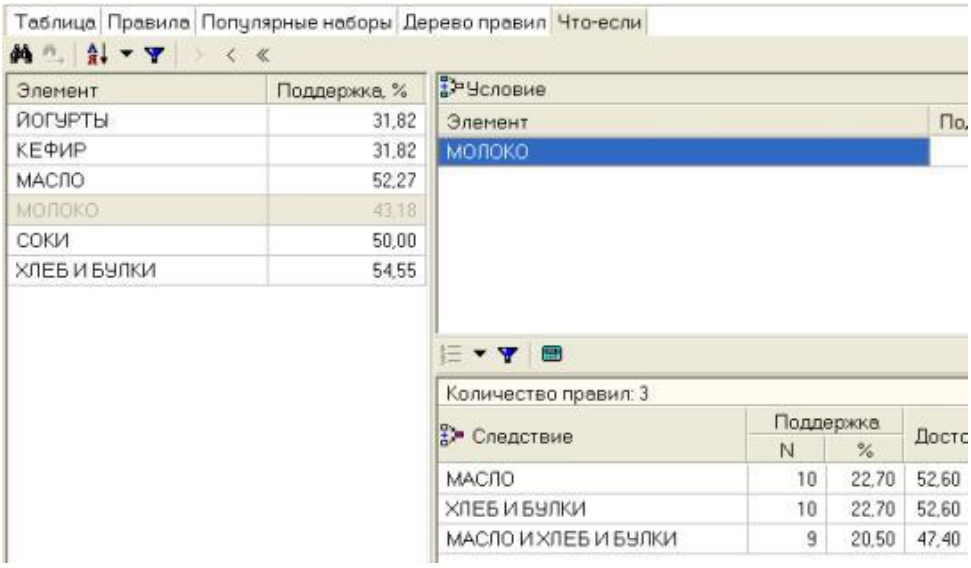


Рис. 4.11. Візуалізація «Що-якщо».

Система PolyAnalyst. Система PolyAnalyst призначена для автоматичного аналізу числових і текстових даних з метою виявлення в них раніше невідомих, нетривіальних, практично корисних і доступних розумінню закономірностей, необхідних для ухвалення оптимальних рішень в бізнесі і в інших областях людської діяльності. В даний час вона є однією з найпотужніших систем Data Mining в світі, реалізованих для Intel платформ і операційних систем Microsoft Windows. Аналогічні системи Data Mining таких провідних виробників, як IBM (Intelligent Miner, Data Miner), Silicon Graphics (SGI Miner), Integral Solutions (Clementine), SAS Institute (SAS) працюють на середніх і великих машинах і коштують десятки і навіть сотні тисяч доларів. Завдяки унікальній технології «Еволюційного програмування» та іншим інноваційним математичним алгоритмам, PolyAnalyst поєднує в собі високу продуктивність «великих систем» з низькою вартістю, властивою програмам для Windows. PolyAnalyst набув широкого поширення в світі. Більше 500 інсталяцій в 20 країнах світу, серед користувачів системи значний список складають найбільші світові корпорації: Boeing, 3M, Chase Manhattan Bank, Dupont, Siemens та інші. Розглянемо реалізацію технологій пошуку асоціативних правил в цьому програмному комплексі.

Модуль Market Basket Analysis (BA) - метод аналізу «корзини покупця». Назва цього методу походить від задачі визначення які товари ймовірно купуються спільно. Проте реальна сфера його застосування значно ширша. Наприклад, продуктами можна вважати сторінки в Інтернеті, або ті або інші характеристики клієнта або відповіді респондентів в соціологічних і маркетингових дослідженнях та інше. Алгоритм BA отримує на вхід бінарну матрицю, в якій рядок - це одна корзина (наприклад, касовий чек), а стовпці заповнені логічними 0 і 1, що позначають наявність або відсутність даної ознаки (товару). На виході формуються кластери ознак, що спільно зустрічаються, з оцінкою їх вірогідності і достовірності. Окрім цього формуються асоціативні направлені правила типа: якщо ознака «A», то з деякою вірогідністю ще і ознака «B» і ще ознака «C». Алгоритм BA в

PolyAnalyst працює виключно швидко і здатний обробляти величезні масиви даних (рис. 4.12).

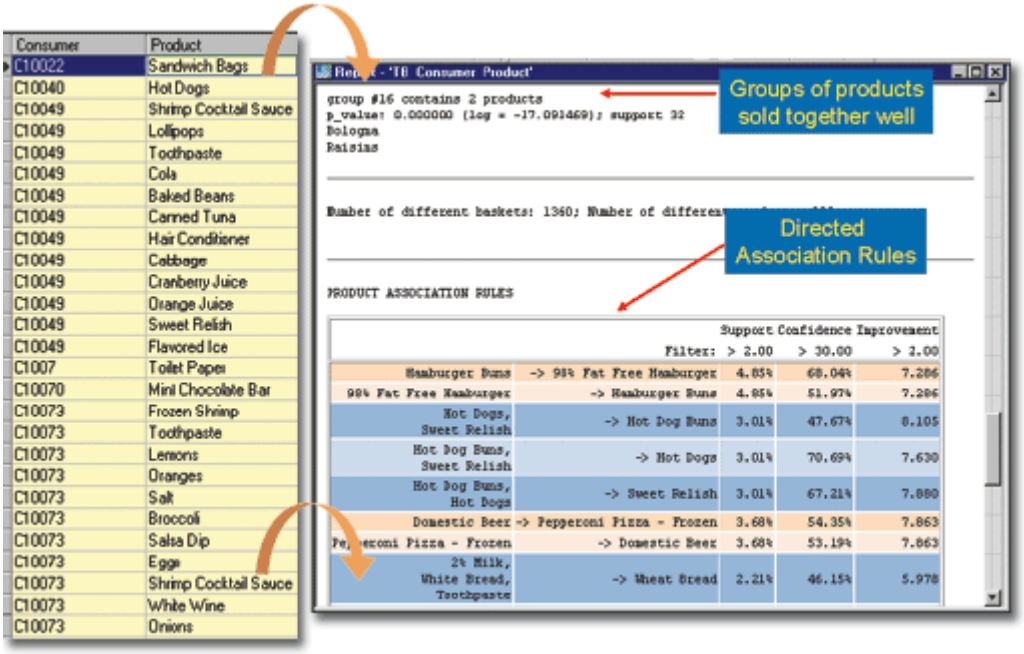


Рис. 4.12. Приклад роботи Market Basket Analysis.

Модуль Transactional Basket Analysis (ТБ) - транзакційний аналіз «корзини». Transactional Basket Analysis - це модифікація алгоритму BA, застосована для аналізу дуже великих даних, що не рідкість для цього типа задач. Він передбачає, що кожен запис в базі даних відповідає одній транзакції, а не одній корзині (набору куплених за одну операцію товарів). На основі цього алгоритму компанія «Мегапьютер» розробила окремий продукт - X-SellAnalyst, призначений для on-line рекомендації продуктів в Інтернет магазинах.

Однією з унікальних особливостей PolyAnalyst є інтеграція інструментів Data Mining - засобів аналізу числової інформації з методами аналізу текстів на природній мові - алгоритмів Text Mining.

Модуль Text Analysis (ТА) - текстовий аналіз. Text Analysis є засобом формалізації неструктурованих текстових полів в базах даних. При цьому текстове поле представляється як набір булевих ознак, заснованих на наявності і частоті даного слова, стійкої словосполуки або поняття в даному тексті. При

цьому з'являється можливість розповсюдити на текстові поля всю потужність алгоритмів, реалізованих в системі PolyAnalyst. Крім того, цей метод може бути використаний для кращого розуміння текстової компоненти даних за рахунок автоматичного виділення найбільш ключових понять (рис. 4.13).

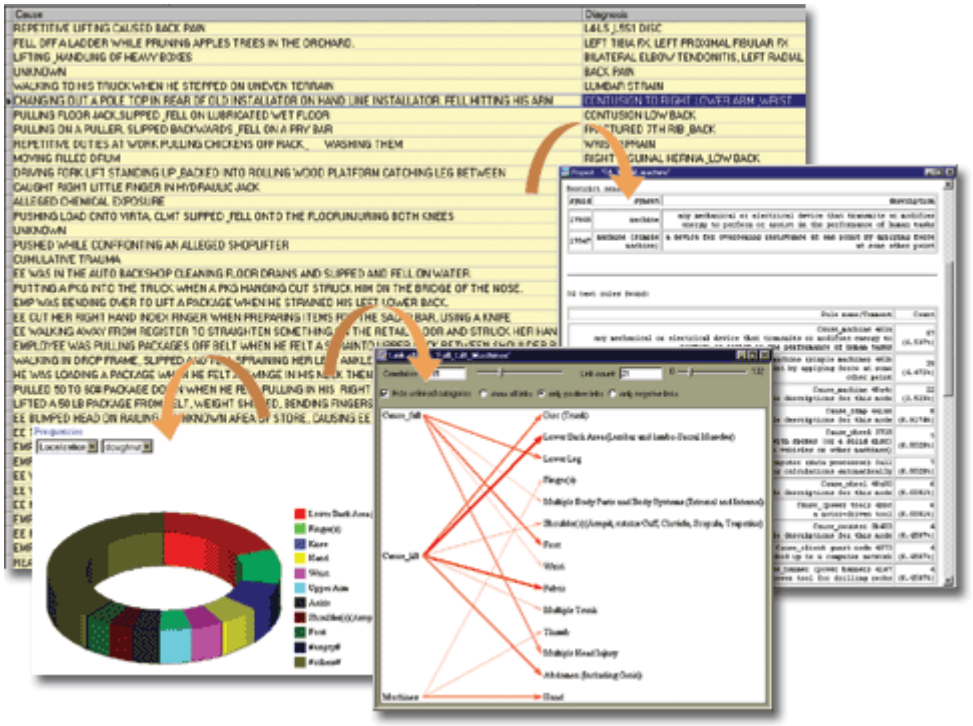


Рис. 4.13. Текстовий аналіз на основі асоціацій.

Модуль Text Categorizer (TC) - каталогізатор текстів. Цей модуль дозволяє автоматично створити ієрархічний деревовидний каталог наявних текстів і помітити кожен вузол цієї деревовидної структури найбільш індикативною ознакою для текстів, що відносяться до нього. Це потрібно для розуміння тематичної структури аналізованої сукупності текстових полів і ефективної навігації по ній.

Модуль Link Terms (LT) - зв'язок понять. Цей модуль дозволяє виявляти зв'язки між поняттями, що зустрічаються в текстових полях бази даних і представляти їх у вигляді графа. Цей граф також може бути використаний для виділення записів, що реалізовують вибраний зв'язок (рис. 4.14).

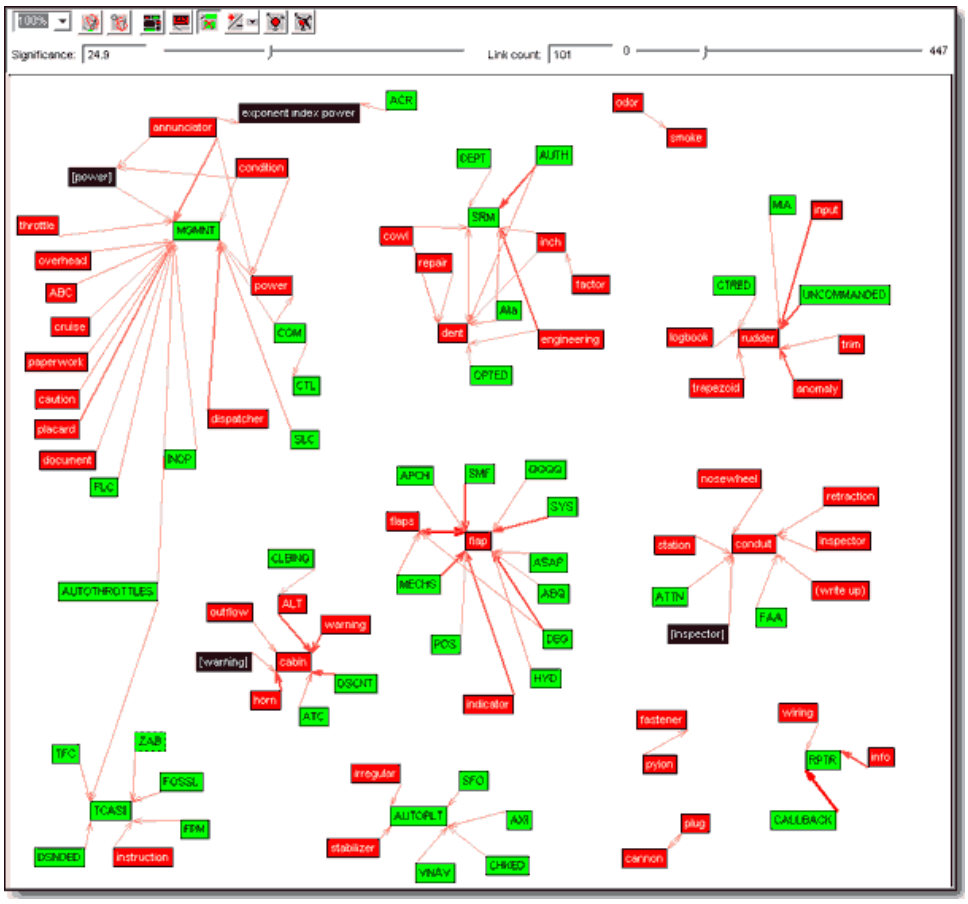


Рис. 4.14. Результати роботи модуля Link Terms.

Пакет STATISTICA. Виконання алгоритму Apriori в пакеті STATISTICA дозволяє швидко обробляти великі бази даних для зв'язків, заснованих на зумовлених «відправних пунктах» значень для дослідження. Корисність цього методу при вирішенні нестандартних задач видобутку даних краще всього розглянути на прикладі. Будемо вважати, що збираються дані з апарату, реєструючого готівку у великому книжковому магазині. Інформація по кожному покупцеві заноситься в базу даних, вона складається з назв куплених книг, додаткових назв магазина і інших куплених товарах. Отже, кожен запис в базі даних представляє одного покупця (транзакцію), і може складатися з однієї купленої книги або множини (можливо сотень) різних пунктів, розташованих в довільному порядку, залежно від послідовності їх поступлення і реєстрації пристроєм. Мета аналітиків – знайти зв'язки між купленими товарами, тобто виявити зв'язки між назвами товарів і частотою їх вжитку. Наприклад, Ви

хочете взнати, яку ще книгу захоче придбати покупець, що вже купив одну. Ця інформація може бути швидко використана для пропозиції покупцеві якихось додаткових найменувань товарів (рис. 4.15).

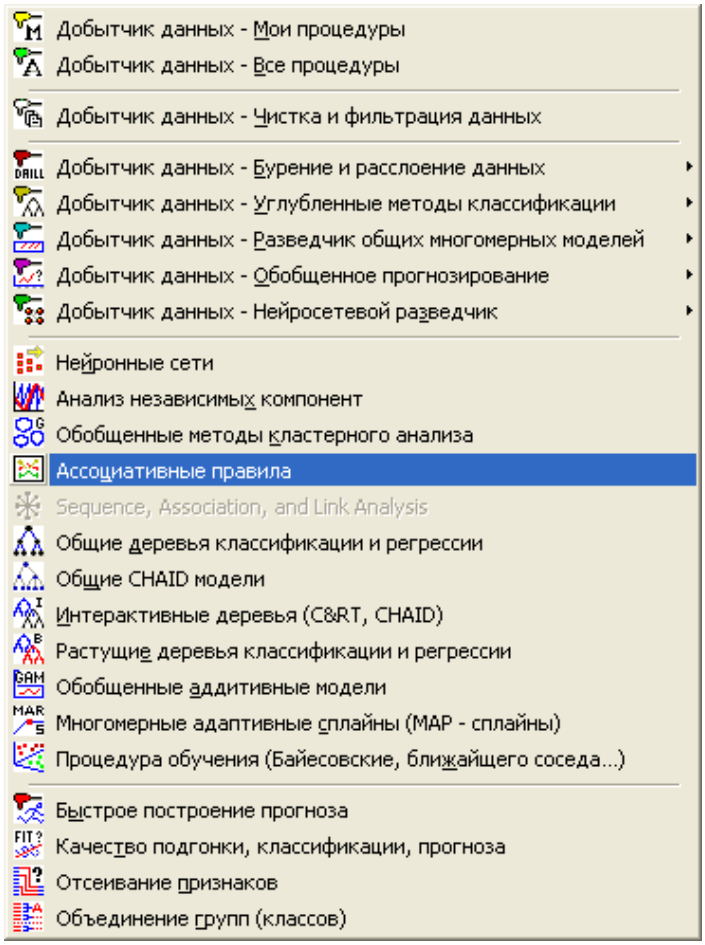


Рис. 4.15. Меню Data Mining пакету STATISTICA.

Правила зв'язків в STATISTICA підтримують всі основні типи даних і формати, в яких зазвичай записуються категорії, об'єкти або протоколи (наприклад, інформація про покупку).

Змінні багатовимірного відгуку. Змінні багатовимірного відгуку зазвичай містять багатовимірні змінні (точніше, список змінних) які можуть містити, для кожного результату спостереження окремо, кодові або текстові значення, що описують окремий «вимір» або транзакцію. Прикладом використання змінних багатовимірного відгуку є: продавець зафіксував

покупку споживача в окремому записі, де кожен запис може містити одне або більш придбань, причому в довільному порядку. Це найприродніший формат даних для зберігання інформації про покупки споживача.

Багатовимірні дихотомії. У цьому форматі даних кожна змінна представлена однією подією або категорією, і дані дихотомії в кожній змінній відображають так чи інакше відповідний об'єкт або категорію, що відноситься до певного спостереження. Наприклад, передбачимо що продавець створив таблицю даних, в якій кожен стовпець містить товари, які можна купити. Кожна транзакція (ряд таблиці даних) записуватиме купив чи ні відповідний покупець цей товар, тобто задіяна чи ні відповідна транзакція.

Первинна обробка даних: підтримка. В першу чергу STATISTICA сканує всі змінні, аби визначити унікальні кодові або текстові значення, знайдені серед змінних для аналізу. При цій первинній обробці відповідна частота, з якою унікальні кодові або текстові значення зустрічаються в кожній транзакції, також будуть обчислені. Можливість того, що транзакція містить певне кодове або текстове значення називається підтримкою. Підтримка також обчислюється при подальших послідовних обробках даних, як імовірність зустрічі подвійних, потрійних і так далі кодових або текстових значень (об'єктах), тобто визначається окремо для «Причини» і «Наслідок» кожного зв'язку.

Вторинна обробка даних: довіра, кореляція. Після первинної обробки даних, всі об'єкти, в яких значення підтримки менше, ніж деякий визначений заздалегідь мінімум підтримки, будуть збережені в пам'яті для подальших обробок даних. Особливістю є те, що STATISTICA обчислюватиме умовну ймовірність для всіх пар кодових і текстових значень, в яких значення підтримки більше, ніж деякий певний мінімум підтримки. Ця умовна ймовірність - результат, який містить кодове або текстове значення X також містить кодове або текстове значення Y, - називається довіра.

На додаток STATISTICA обчислить підтримку для кожної пари кодових або текстових значень і кореляцію, засновану на підтримці. Значення кореляції

для пари кодових або текстових значень $\{X, Y\}$ обчислюється як підтримка цієї пари, що ділиться на квадратний корінь з величини підтримки X і Y. Після другої обробки даних програма збереже в пам'яті ті пари кодових або текстових значень, які:

- мають значення довіри, більше чим деякий визначений користувачем мінімум довіри;
- мають підтримку, більшу чим деякий визначений користувачем мінімум підтримки;
- мають значення кореляції, більше чим деяка мінімальна кореляція.

Подальші обробки даних: максимальний розмір об'єкту в «Причина», «Наслідок». STATISTICA продовжуватиме сканувати дані, обчислюючи підтримку, довіру і кореляцію для подвійних кодових або текстових значень, потрійних і так далі При кожному повторенні програма витягуватиме правила зв'язку вигляду: *якщо «Причина» то «Наслідок»*, де «Причина» і «Наслідок» представлені кодовими або текстовими значеннями (об'єктами), або комбінацією кодових або текстових значень (об'єктів). Процес продовжуватиме до тих пір, поки ще можуть бути знайдені зв'язки, що задовольняють мінімуму значення підтримки, довіри і умови кореляції. Процес може продовжувати вибудовувати дуже складні правила зв'язку, наприклад, якщо X_1 і X_2 … і X_{20} то Y_1 і Y_2 ... і Y_{20}. Аби уникнути небажаного ускладнення, користувач додатково може точно встановити максимальну кількість кодових або текстових значень (об'єктів) в правилах зв'язку «Причина» і «Наслідок». Тоді це значення сприйматиметься як максимальний розмір об'єкту в зв'язку «Причина» і «Наслідок».

Основними статистиками, що обчислюються для зв'язків є підтримка (відносна частота умови «Причина» або «Наслідок»), довіра (умовна ймовірність «Причини», визначеної «Наслідком»), і кореляція (підтримка для «Причина» і «Наслідок», ділена на квадратний корінь з результату підтримки для «Причина» і підтримки для «Наслідок»). Ці показники можна звести в електронну таблицю (рис. 4.16). Ця таблиця результатів показує, як зв'язки

можуть бути застосовані до задачі аналізу тексту. Підтримка, значення довіри і кореляції виражені у відсотках.

Data: Summary of association rules (Scene 1)*

Summary of association rules (Scene 1.sta)
Min. support = 5.0%, Min. confidence = 5.0%, Min. correlation = 5.0%
Max. size of body = 10, Max. size of head = 10

	Body	==>	Head	Support(%)	Confidence(%)	Correlation(%)
154	and, that	==>	like	6.94444	83.3333	91.28709
126	like	==>	and, that	6.94444	100.0000	91.28709
163	and, PAROLLES	==>	will	5.55556	80.0000	73.02967
148	will	==>	and, PAROLLES	5.55556	66.6667	73.02967
155	and, you	==>	your	5.55556	80.0000	67.61234
122	your	==>	and, virginity	5.55556	57.1429	67.61234
164	and, virginity	==>	your	5.55556	80.0000	67.61234
121	your	==>	and, you	5.55556	57.1429	67.61234
73	that	==>	like	6.94444	41.6667	64.54972
75	that	==>	and, like	6.94444	41.6667	64.54972
161	and, like	==>	that	6.94444	100.0000	64.54972

Рис. 4.16. Пошук асоціативних правил в пакеті STATISTICA.

Графічне представлення зв'язків. Приведемо результати аналізу даних з прикладу Fastfood.sta. Опитувані в дослідженні вказували 3 їх улюблені фаст-фуд блюда. Правила зв'язки, отримані виходячи з цих даних, показані на 2М графіці зв'язку (рис. 4.17).

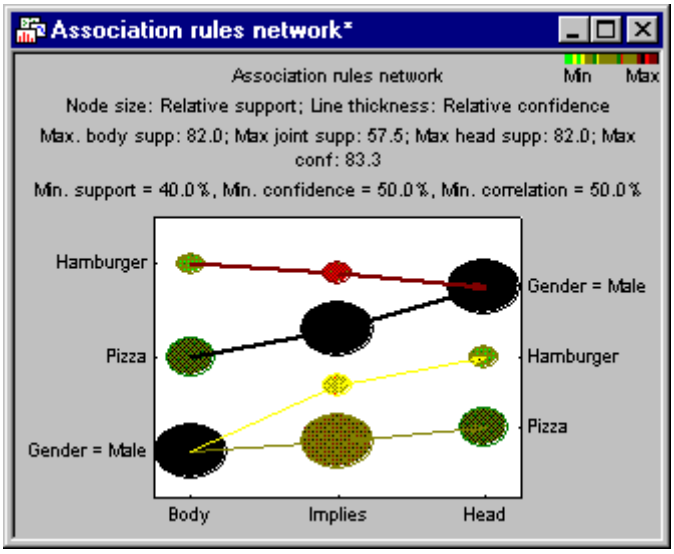


Рис. 4.17. Графічне відображення правила зв’язків.

Пункти, що визначають причини показані на графіці зліва, а наслідок - справа. Лінії, які сполучають причину із наслідком, відображують правила зв'язку. Значення підтримки для «Причина» і «Наслідок» для кожного зв'язку відбиваються в розмірі і кольорах кіл. Товщина кожної лінії відображає довірче значення (умовну ймовірність, яку для «Причина» визначає наслідок) для відповідного зв'язку; розміри і кольори кіл в центрі, над написом Implies відображають об'єднану підтримку (для тих, що зустрілися) відповідних компонентів «Причина» і «Наслідок».

У 3М графіці зв'язку підтримка для компонентів «Причина» і «Наслідок» кожного зв'язку відбиті в розмірі і кольорах кіл в 2М площині. Товщина кожної лінії відображає значення довіри (можливість об'єднання) для відповідного зв'язку; розміри і колір «плаваючих» кіл, побудованих напроти осі Z вказують на з'єднання підтримки (для частоти повторень) відповідних компонентів «Причина» і «Наслідок» правил зв'язку. Позиція накресленого кола по відношенню до осі Z відображає відповідне значення довіри. Отже, абсолютно точно визначений і виражений графічно висновок дозволяє сформулювати 2 правила: опитувані, які назвали Піца, як саме їх улюблене, також назвали Гамбургер, і навпаки Рис. 4.18).

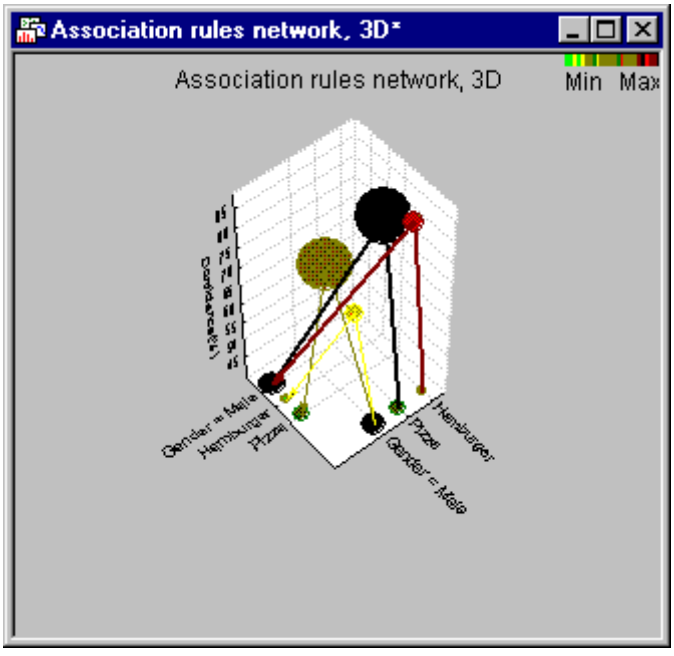


Рис.4.18. 3D відображення асоціативних правил в пакеті STATISTICA.

Система 1С: Предприятие 8.0. Одна з головних тенденцій на ринку обліково-управлінських систем – це постійне підвищення попиту на вживання засобів аналітичної обробки даних, що забезпечують ухвалення обґрунтованих управлінських рішень. Саме тому одним із стратегічних напрямів розвитку системи ПО «1С: Підприємство» є постійне розширення можливостей засобів економічної і аналітичної звітності. Підприємствам все частіше потрібні якісно інші засоби, що дозволяють автоматично шукати неочевидні правила і виявляти невідомі закономірності. Саме так можна генерувати якісно нові знання накопиченою компанією інформації і приймати деколи зовсім нетривіальні рішення для підвищення ефективності бізнесу, застосовуючи методи інтелектуального аналізу даних.

Пошук асоціацій Цей метод призначений для виявлення стійких комбінацій елементів в певних подіях або об'єктах. Результати аналізу представляються підсистемою у вигляді груп асоційованих елементів. Тут же окрім виявлених стійких комбінацій елементів наводиться розгорнута аналітика по асоційованих елементах (рис. 4.19).

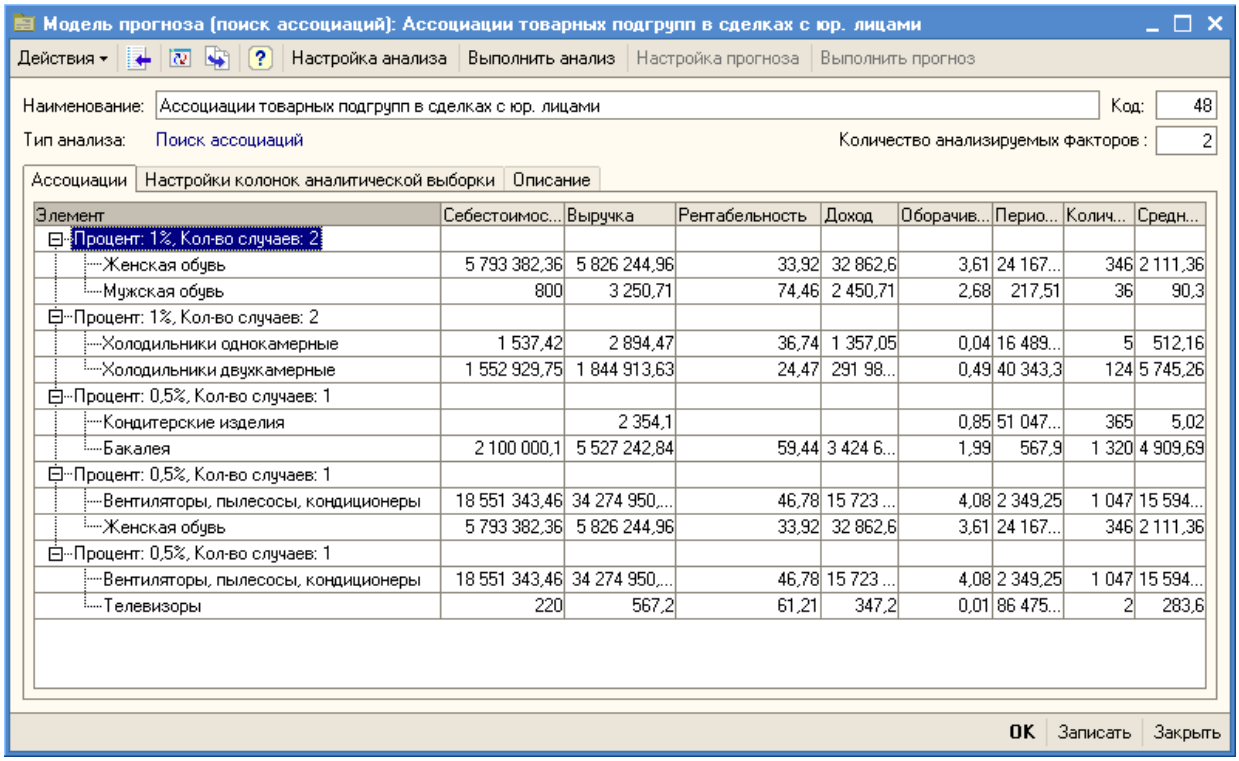


Рис. 4.19. Представлення результатів аналізу методом пошуку асоціацій".

Спочатку метод був розроблений для пошуку типових поєднань товарів в покупках, тому інколи його ще називають аналізом купівельної корзини. Стосовно цього сценарію в якості асоційованих елементів, як правило, виступають товарні групи або окремі товари. Групуючим об'єктом, що об'єднує елементи вибірок, може бути будь-який об'єкт інформаційної системи, що ідентифікує операцію, наприклад: замовлення покупця, акт про надання послуг або касовий чек.

Інформація про закономірності в товарних перевагах покупців дозволяє підвищити ефективність управління стосунками з клієнтами (у частині рекламних кампаній і маркетингових акцій), ціноутворення (формування комплексних пропозицій і системи знижок), управління запасами і мерчендайзинга (розподіл товарів в торгівельних залах). Іншим прикладом використання цього методу є визначення комбінацій рекламних каналів, яким віддаються переваги клієнтами, для виключення їх дублювання при проведенні цільових рекламних кампаній. Це дозволяє істотно понизити витрати на подібні заходи.

Реалізований в платформі алгоритм пошуку асоціацій має досить гнучкі засоби управління адекватністю моделей аналізу або прогнозу. Параметр «Мінімальний відсоток випадків» визначає «поріг спрацьовування» алгоритму на ту або іншу комбінацію елементів в події або об'єкті, що дозволяє не брати до уваги слабо поширені асоціації. Параметр «Мінімальна достовірність» визначає необхідну стійкість шукаємих асоціацій, а параметр «Мінімальна значущість» дозволяє виявити з них найбільш пріоритетні. Істотно полегшує сприйняття результатів аналізу і прогнозу параметр «Тип відсікання правил», який може набувати значень «Відсікати надлишкові» і «Відсікати покриті іншими правилами».

Для практичної інтерпретації результатів, отриманих даним алгоритмом критично важливе розбиття вихідної множини асоційованих елементів на дійсно однорідні з точки зору здійснюваного аналізу групи.

При створенні даної інформаційної системи були розроблені типові проекти застосування інтелектуального аналізу даних. Одним з таких проектів є «Управління ланцюжками постачань»

- *Сценарій* "Оптимізація вибору постачальників по товарній групі". Вибір домінуючих постачальників «першого ряду» для ключових товарних груп надзвичайно важливий для стабілізації системи логістики зокрема і загальної системи управління ланцюжками постачань в цілому, зменшення середньої тривалості ланцюжків постачань. З іншого боку, тісніша інтеграція з основними постачальниками дозволяє, як правило, істотно понизити собівартість товарів. У зв'язку з цим представляє інтерес аналіз стійких комбінацій постачальників в різних товарних групах порівняно з аналітикою по асоційованих в рамках груп постачальниках. Це дозволяє виявити «пересічення» постачальників в різних товарних групах і оптимізувати взаємини з ними.
- *Алгоритм* – «Пошук асоціацій».
- *Прогнозні атрибути* - стійкі комбінації постачальників.
- *Основні чинники* – товарні групи.
- *Розшифровка* - аналітика по постачальниках (об'єм закупівель, виручка, умови постачання, оплати, песимістичний, оптимістичний, середній терміни виконання замовлення).
- *Приклад закономірності*. Стійка асоціація крупного і непередбачуваного постачальника А і передбаченого середнього постачальника Б у великій кількості товарних груп. Можливо, при формуванні замовлень по конкурентних товарних групах як основного розглядати середнього постачальника, якщо обсяг замовлення великому не перевищує деякого (що дає істотний виграш на масштабах) порогу.

Клієнт Data Mining для Excel. Клієнт Data Mining для Excel дозволить провести повний цикл інтелектуального аналізу даних за допомогою клієнта Excel з використанням даних електронних таблиць або зовнішнього джерела, доступного базі даних Analysis Services, зокрема, пошук асоціативних правил (рис. 4.20).

Рис. 4.20. Панель інструментів клієнта Data Mining для Excel 2007.

Організація зліва направо та угрупування кнопок на панелі інструментів відображає типовий порядок виконання задач в проекті аналізу даних.

Підготовка даних. Задача вибору правильних атрибутів з джерела даних і представлення їх в правильному форматі займає високий відсоток часу в процесі побудови аналітичних моделей. Ця секція надає інструменти для основних задач підготовки даних до початку їх поглибленого аналізу.

Дослідження даних - служить для побудови графіка розподілу дискретних і безперервних змінних, а також для додавання угрупувань у вихідні дані.

Очищення даних - служить для видалення викидів і для зміни значень міток дискретних даних (наприклад, якщо вихідні дані містять значення «M» і «W» в колонці «Стать», а ви вважаєте за краще бачити ці значення як «чоловічий» і «жіночий» для ясності презентації результатів).

Секціонування даних - служить для розбиття вихідних даних на навчальну і тестову множину за допомогою випадкових вибірок вихідних даних.

Побудова моделей. Ця секція призначена для створення і обробки моделей аналізу даних. Вона надає майстри, які допомагають вам побудувати більшість поширених типів моделей Data Mining без необхідності знатися на відповідних алгоритмах і пов'язаних з ними параметрів, які виконуються на сервері. Також, в цій секції є можливості, що дозволяють користувачеві вибрати конкретний алгоритм і задавати додаткові параметри. Нижче приведен приклад застосування майстра асоціативних правил, меню для знаходження асоціацій в транзакційних даних (рис. 4.21).

Рис. 4.21. Майстер асоціативних правил.

Існують також інші програмні засоби для пошуку асоціативних правил, серед яких слід виділити комерційне та вільно поширюване програмне забезпечення. Зокрема, серед комерційного програмного забезпечення найбільш популярними є:

- Azmy SuperQuery - пошукова система асоціативних правил;
- Clementine, набір від SPSS, що включає аналіз ринкової корзини;
- IBM Intelligent Miner for Data;
- The LPA Data Mining Toolkit - підтримує пошук асоціативних правил в реляційних базах даних;
- Magnum Opus є швидким інструментом пошуку асоціативних правил в даних, підтримується операційними системами Windows, Linux і Solaris;
- Nuggets - це набір, що включає пошук асоціативних правил і інші алгоритми;
- Purple Insight MineSet є набором візуального Data Mining, що включає візуалізатор асоціативних правил.

Серед вільно поширюваного програмного забезпечення слід виділити:

- Apriori - інструмент для знаходження асоціативних правил за допомогою алгоритму Apriori;
- ARtool – інструмент, що містить набір алгоритмів для пошуку асоціативних правил в бінарних базах даних (binary databases);
- DM-II system - інструмент включає алгоритм CBA для виконання класифікації на основі асоціативних правил і деяких інших характеристик;
- FIMI, Frequent Itemset Mining Implementations - є репозиторієм, що включає програмне забезпечення і бази даних.

4.3. Практичний аспект застосування технології асоціативних правил.

Практичні додатки систем інтелектуального аналізу даних на основі технологій асоціативних правил воістину безмежні: виробництво, торгівля, фінанси, медицина, соціологія, наукові дослідження, освіта та ін. Такий інструментарій використовується для задач технічної і медичної діагностики, проектування, управління процесами, контролю якості, прогнозування, оцінки кредитоспроможності, аналізу стану ринків, маркетингових досліджень, роботи з клієнтами, соціологічних опитів, моделювання і вивчення складних систем на основі історії їх еволюції [9, 37, 47, 53, 68, 70, 77].

У великих масивах корпоративних даних часто зберігаються відповіді на багато питань, які цікавлять керівництво і співробітників організацій. Розглянемо деякі з них.

Маркетингові задачі. Одним з прикладів аналізу механізмів стимулювання продажів на основі великих масивів накопичених даних про поведінку споживачів виступають узагальнені асоціативні правила. Ставиться завдання знайти приховані закономірності і типові шаблони поведінки покупців. Для цього вводиться поняття «Купівельна транзакція»: набір товарів, куплених покупцем за один візит до супермаркету. Відмітною властивістю є те, що визначувані асоціативні правила включають елементи, які є предками

елементів, що входять в множину транзакцій. В результаті можливо виявляти асоціації не лише між окремими елементами транзакцій, але і між різними рівнями ієрархії елементів. Якщо продукти харчування розбити, наприклад, на дві групи товарів «Молочні продукти» і «Напої», то ілюстрацією першого випадку буде асоціативне правило «Якщо покупець купив сік, то він, швидше за все, купить кефір», а ілюстрацією другого — «Якщо покупець купив молочні продукти, то він, швидше за все, купить мінеральну воду». Подібні асоціації не так конкретні, як у випадку: «Якщо покупець купив хліб, то з вірогідністю 75% він купить і молоко». Але використання алгоритму пошуку узагальнених асоціативних правил дозволяє значно розширити круг вирішуваних завдань. Цікавою задачею, наприклад, виступає знаходження залежностей між товарами, що продаються деякою фірмою, і її покупцями в наступній постановці: потрібно знайти тих покупців на конкретні товари даної фірми (наприклад, певні товари, які «завалялися» на складі), які до цих пір купували подібні товари інших виробників.

Розглянемо практику комп'ютерного розв’язання типової проблеми, що вирішується за допомогою знаходження асоціативних залежностей. Йдеться про механізми стимулювання продажів, що базуються на знаннях про найбільш типову поведінку покупців при оформленні замовлень. Практична версія системи використовувала 2 бази даних. У кожній є інформація про історію продажів оптової фірми за досить тривалий термін. У першому випадку основною діяльністю фірми є оптовий продаж кондитерських виробів, і аналізована історія продажів складає 1 рік (близько 1,5 млн. записів); у другому випадку це підприємство, що займається оптовим продажем медикаментів, і аналізована історія продажів складає 1,5 років (близько 0,7 млн. записів).

Виявлення дійсно цікавих правил - це одна з головних підзадач при обчисленні асоціативних залежностей. Для того, щоб отримати дійсно цікаві залежності, потрібно розібратися з декількома емпіричними правилами, що претендують на звання аксіом:

1. Зменшення мінімальної підтримки приводить до того, що збільшується кількість потенційно цікавих правил, проте це вимагає істотних обчислювальних ресурсів. Одним з обмежень зменшення порогу minsupport є те, що дуже маленька підтримка правила робить його статистично необґрунтованим.

2. Зменшення порогу достовірності також приводить до збільшення кількості правил. Значення мінімальної достовірності також не має бути дуже маленьким, оскільки цінність правила з достовірністю 5% настільки мала, що це правило і правилом вважати не можна.

3. Правило з дуже великою підтримкою з точки зору статистики є великою цінністю, але з практичної точки зору це, швидше за все, означає те, що або правило всім відомо, або товари, присутні в ньому, є лідерами продажів, звідки слідує їх низька практична цінність.

4. Правило з дуже великою достовірністю практичної цінності в контексті вирішуваного завдання не має, оскільки товари, що входять в наслідок, покупець швидше за все вже купив.

Якщо значення верхньої межі підтримки має дуже велике значення, то в правилах основну частину складатимуть товари - лідери продажів. При такому розкладі не представляється можливим зменшити поріг minsupport до того значення, при якому можуть з'являтися цікаві правила. Причиною тому є просто величезне число правил, і, як наслідок, брак системних ресурсів. Причому отримувані правила відсотків на 95 містять товари - лідери продажів. Приклад таких правил представлений на рисунку 4.22 (зверніть увагу на значення величин, minsupport і maxsupport).

Умова	Наслідок	Підтримка (%)	Достовірність (%)
ПАСТИЛА ВАНІЛЬНА і З-Р ВАНІЛЬНИЙ і ВФ ГУЛІВЕР	ВФ АРТЕК-СУПЕР	1,11	52,27
ПЕЧ.ВІВСЯНЕ і З-Р КРЕМ-БРЮЛЕ і ВФ ШОК-ГОРІХОВИЙ і	ВФ АРТЕК-СУПЕР і З-Р ВАНІЛЬНИЙ	1,15	59,50

З-Р ВЕРШКОВИЙ З-Р ФРУКТОВО-ЯГІДНИЙ	З-Р КРЕМ-БРЮЛЕ	1,11	68,32
ПАСТИЛА ВАНІЛЬНА і З-Р КРЕМ-БРЮЛЕ і ПЕЧ. З РОДЗИНКАМИ	ПЕЧ.ВІВСЯНЕ і З-Р ВАНІЛЬНИЙ	1,12	55,12
...	...	...	...

Рис. 4.22. Приклад асоціативних правил (БД: Кондитерські вироби; minsupport: 1.1; maxsupport: 100; minconfidence: 50; maxconfidence: 95).

Варіюючи верхньою і нижньою межами підтримки, можна позбавитися від очевидних і нецікавих закономірностей. Як наслідок, правила, що генеруються алгоритмом, набирають наближеного до реальності вигляду (рис. 4.23).

Умова	**Наслідок**	**Підтримка (%)**	**Достовірність (%)**
ДОЛЬКА МУЛЬТИФРУКТ НЕКТАР	ДОЛЬКА ТОМАТ СІК	0,93	77,33
КФ ВІЗІТ	КФ ВЕЧІРНІ	0,91	45,24
З-Р РОМОВИЙ В ШОК.	З-Р КАВОВИЙ В ШОК.	1,39	63,97
...	...	...	...

Рис 4.23. Приклад асоціативних правил (БД: Кондитерські вироби; minsupport: 0,9; maxsupport: 10; minconfidence: 40; maxconfidence: 95)

В деяких випадках значення порогу minsupport має бути дійсне маленьким. Прикладом можуть бути правила, отримані на основі даних бази даних «Медикаменти». Річ у тому, що в деяких областях торгівлі товарів настільки багато, що кожен з них не володіє ринком навіть на декілька відсотків. Такою областю є торгівля фармацевтичною продукцією. Наприклад, навіть в маленькій аптеці кількість найменувань товарів складає біля 2-х тисяч. В цьому випадку з'являється ще гостріша необхідність в зменшенні мінімальної підтримки, яку, у свою чергу, просто не удається зменшити, не зменшуючи верхній поріг (рис. 4.24). На рисунку 4.24 представлені приклади правил з великим значенням порогу maxsupport.

Умова	Наслідок	Підтримка (%)	Достовірність (%)
Андипал # 10 і Еналаприл тб. 0,01 № 20	Аскорбінова к-та 5% 1мл #10	1,01	64,31
Ортофен 2,5% 3,0 № 10 і Екстракт валер'яни 0.02 # 10	Аскорбінова к-та 5% 1мл #10 і Цианокобаламін (віт.В12)500мкг 1мл #10	1,02	42,63
Діоксидин 1% 5.0 # 10 і Ортофен 2,5% 3,0 № 10	Аскорбінова к-та 5% 1мл #10 і Екстракт валер'яни 0.02 # 10	1,09	48,22
Аскорбінова к-та 5% 1мл #10 і Цитрамон П тб.# 6	Рибоксин 0,2 тб.#50 і Валідол 0,05 капс.#20	1,05	40,96
...	...	...	...

Рис 4.24 Приклад асоціативних правил (БД: Медикаменти; minsupport: 0,9; maxsupport: 100; minconfidence: 40; maxconfidence: 95).

А ось що виходить, якщо зменшити значення максимально доступної підтримки і порогу minsupport (рис. 4. 25).

Умова	Наслідок	Підтримка (%)	Достовірність (%)
Ібупрофен 0,2 тб.#50 і Валідол тб.0.06 # 10	Анальгін 0.5 # 10	0,10	51,89
Валідол тб.0.06 # 10 і Парацетамол 0.5 # 10 і Ацетилсалицилова к-та 0,5 #10	Бромгексин 0,008 тб.#10	0,11	72,84
Аеровіт # 30 і Гептавіт # 20	Декамевіт тб.#20	0,10	52,38
Еуфіллін 0,15 тб.#30 і Бромгексин 0,008 тб.#10	Парацетамол 0.5 # 10	0,13	59,83
...	...	...	...

Рис 4.25. Приклад асоціативних правил (БД: Медикаменти; minsupport: 0,1; maxsupport: 5; minconfidence: 40; maxconfidence: 95).

Описані вище механізми є однією із складових частин інформаційної системи виписки замовлень. Обчислення правил при цьому проходить у фоновому режимі, тобто користувач цієї системи (менеджер) цього не бачить. Проте при виписці замовлень або ж по запиту йому будуть представлені

обчислені правила, що допоможе менеджерові бути підготовленішим до питань клієнтів. Обчислені правила можна використовувати як для оперативної підказки, так і для аналізу. Але в обох випадках асоціативні правила потрібно грамотно представити користувачеві. Для цього можна використовувати різні відображення, головна задача яких - бути максимально зрозумілими і легко засвоюваними.

В інформаційній системі представлено чотири варіанти відображення асоціативних правил: у вигляді звичайної таблиці, у вигляді форматованого тексту, у вигляді дерева, у вигляді перехресної таблиці. В кожного відображення є як свої переваги, так і недоліки, і вибір конкретного залежить як від тих цілей, які користувач хотів би досягти, так і від особливостей особистого сприйняття.

Однією з найпростіших для сприйняття візуалізацій асоціативних правил є таблиця (рис. 4.22 – 4.25). Її перевагами є:

- легко зрозуміти, де умова, а де наслідок (у різних стовпцях);
- можливість експорту в більшість форматів;
- можливість сортування по умові, по слідству, по підтримці, по достовірності;
- можливість фільтрації по умові, по слідству, по підтримці, по достовірності.

Недоліки: дуже велика кількість правил приводить до генерації громіздкого документа і, як наслідок, до складності його розуміння.

Найприроднішою конструкцією для сприйняття правил є конструкція вигляду: ЯКЩО «Умова» ТО «Наслідок». Такий варіант відображення правил – найбільш доступна форма представлення (рис. 4.26).

...
Правило N 6
ЯКЩО Метронідазол 0,25 тб.#10
ТО Ібупрофен 0,2 тб.#50
Підтримка = 1,20%
Достовірність = 44,62%

Правило N 7
ЯКЩО Пірідоксину г/х (витий.В6) 2мг тб.#50
ТО Магнію сульфат 25% 5мл #10
Підтримка = 0,60%
Достовірність = 42,30%
...

Рис. 4.26. Асоціативні правила у вигляді форматованого тексту.

Перевагами цього варіанту відображення є: природний запис правил, можливість експорту в .rtf файл, можливість сортування по умові, по слідству, по підтримці та по достовірності, можливість фільтрації по умові, по слідству, по підтримці та по достовірності. Недоліки: дуже велика кількість правил приводить до генерації громіздкого документа, і як наслідок складність в розумінні.

Один з варіантів відображення асоціативних правил – це дерево правил. На перший погляд використання такого відображення невиправдане з точки зору розуміння сенсу правил, але при частому використанні ця візуалізація стає просто незамінною. Правило читається таким чином:

1. Вираз, який буде умовою правила збирається починаючи з поточного (виділеного) вузла.

2. Далі, рухаючись до кореневого вузла, до виразу додається «І» плюс «значення батьківського вузла».

3. Процедура (2) повторюється рекурсивно до тих пір, поки не досягне батьківського вузла.

4. Наслідки знайденої умови (якщо такі є) знаходяться праворуч від дерева в таблиці.

Наприклад, правила для виділеного вузла виглядатимуть так (рис. 4.27):

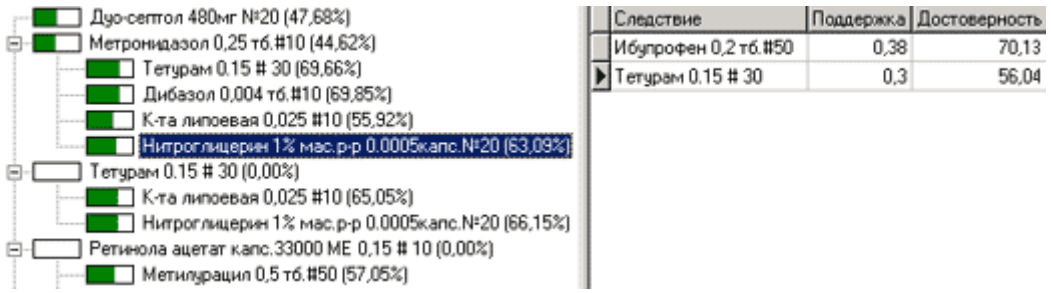


Рис. 4.27. Асоціативні правила у вигляді дерева (БД: Медикаменти; minsupport: 0,1; maxsupport: 3; minconfidence: 40; maxconfidence: 95).

Перевагами цього засобу є: компактний вигляд, групування правил по умові, можливість відображення значень підтримки і достовірності правила в графічному вигляді. Недоліки: незвичайне відображення правил, і, як наслідок, складність розуміння правил на перших порах.

Ще одним варіантом візуалізації асоціативних правил є перехресна таблиця. У цій таблиці по осі X перераховані всі правила, а по осі Y - всі товари, що увійшли до правил (рис. 4.28). Правило читається таким чином:

1. Вибирається який-небудь стовпець. Це буде одне правило.
2. У умову входять товари, позначені знаком «+».
3. У наслідок входять товари, позначені знаком «=».
4. Підтримка і достовірність правила вказані в 2-му і 3-му рядках відповідно.

№	4	5	6	7	8	9	10	11	12
Поддержка (%)	1,20	1,20	0,98	0,83	0,55	0,72	0,55	0,75	0,59
Достоверность (%)	44,62	40,50	40,44	52,35	69,66	51,36	62,22	42,43	56,59
Викасол тб.#10				+					
Викалин # 10				=					
Дуо-септол 120мг №20									
Дуо-септол 480мг №20									
Ибупрофен 0,2 тб.#50	=	+			=		+	=	
К-та липоевая 0,025 #10								+	
Магния сульфат 25% 5мл #10									
Метронидазол 0,25 тб.#10	+	=			+		=		
Нитроглицерин 1% мас.р-р 0.0005капс.№20									
Папаверина г/хл 0,04 тб.#10									
Пиридоксина г/х (вит.В6) 2мг тб.#50						+			
Проходол детский суспензия 100 мл									=
Проходол таб.№ 12									+
Ретинола ацетат капс.33000 МЕ 0,15 # 10						=			
Тетурам 0.15 # 30					+		+		
Тисасен драже 10 мг # 30			+						
Тисацид тб. 500 мг # 30			=						

Рис. 4.28. Асоціативні правила у вигляді перехресної таблиці (БД: Медикаменти; minsupport: 0,1; maxsupport: 3; minconfidence: 40).

Перевагами перехресних таблиць є: можливість побачити всі товари, присутні у всіх правилах, можливість сортування по умові, по слідству, по підтримці та по достовірності, можливість фільтрації по умові, по слідству, по підтримці та по достовірності. Недоліки: дуже велика кількість товарів, присутніх в правилах, викликає деяку незручність при аналізі закономірностей.

Слід зазначити, що застосованість алгоритму цим не обмежується. За допомогою даного алгоритму можна знаходити закономірності між зв'язаними подіями, тобто якщо вихідні дані представлені у вигляді подій, то на цих даних можна побудувати асоціативні закономірності. Одним з таких прикладів може служити знаходження залежностей вигляду: ЯКЩО «РЕЗУЛЬТАТИ АНАЛІЗІВ» ТО «ЗАХВОРЮВАННЯ», де «РЕЗУЛЬТАТИ АНАЛІЗІВ» – це результати аналізів пацієнта, а «ЗАХВОРЮВАННЯ» – це список захворювань, якими, можливо, хворіє даний пацієнт. Список же потенційно можливих областей дуже великий.

Задачі митниці. Технологія асоціативних правил може успішно застосовуватися для виявлення прихованих тенденцій в зовнішньоторговельній діяльності. Одна з основних проблем, що стоїть перед митними органами, полягає у виявленні навмисного спотворення вантажних митних декларацій. Через обмежені ресурси повна перевірка всіх переміщуваних через кордон вантажів неможлива. Проте митниця збирає детальні бази даних по вантажних митних деклараціях. Аналіз цих даних може бути використаний для виявлення тенденцій в зовнішній торгівлі України по групах товарів, найбільш схильних до фальсифікації при проходженні митниці — «товарів ризику». Маючи дані про такі товари, митні пости могли б ретельніше перевіряти проходження відповідних вантажів і зменшити втрати від фальсифікації митних документів. Однією з особливостей задачі стала відсутність «тренувального» набору даних - даних, для яких було б апріорі відомо, які з них є спробою фальсифікації вантажної митної декларації, а які є сумлінно задекларованими товарами. Це істотно обмежувало круг алгоритмів, які можна було використовувати: наприклад, популярні методи типа дерева рішень, нейроні мережі і тому

подібне вимагають попереднього навчання на тренувальному наборі даних. У нашому розпорядженні залишалися лише алгоритми асоціативних правил.

Як правило, при «прикритті» одного товару іншим в рамках одного вантажу (і однієї митній декларації) дійсно перевозяться обоє товарів, проте доля «дорогого» занижується. Цей факт і може бути використаний для виявлення подібних пар. При відборі потенційних пар «товар риску» - «товар прикриття» використовувалися наступні критерії:

1. $P(A/B) = \frac{P(AB)}{P(B)}$ або $P(B/A) \approx 1$ - умовна ймовірність ввозу товару A, коли по тій же декларації ввозиться товар B, достатньо велика (аналогічний критерій застосовується а алгоритмі асоціативних правил системи Oracle Server 9i ODM);
2. $S(A) > S(B)$ - мито на «товар риску» більше ніж на «товар прикриття»;
3. $netto(B) > 0$ - порівняння зі статистикою ЄС показує завищення об’єму імпорту для «товарів прикриття»;
4. $netto(A) < 0$ - порівняння зі статистикою ЄС показує заниження об’єму імпорту для «товарів риску».

Перший критерій основний і означає, що один з товарів найімовірніше супроводить іншому. Вибір умовної вірогідності, замість, наприклад, коефіцієнта кореляцій, обумовлюється їх більшою чутливістю. Коефіцієнт кореляції близький до одиниці лише в разі, якщо обоє товарів весь час ввозяться одночасно. Критерій накладає набагато слабкішу умову: лише один з товарів постійно супроводить іншому, оскільки один з товарів може ввозитися у великих об'ємах без жодного супроводу. Використаний критерій відомий в літературі як алгоритм асоційованих правил і, зокрема, реалізований в Oracle Data Mining 9i. Втім, висока кореляція одного з товарів з іншим ще не означає, що товар обов'язково прикривається іншим: безліч людей щодня купують одночасно хліб і молоко без жодного злого наміру. І при імпорті товарів існують випадки природної кореляції між товарами. Аби очистити відібрані пари від таких випадків, були накладені додаткові умови: прикриття має бути

економічно вигідно, а порівняльний аналіз статистичних даних повинен підтверджувати факт прикриття.

Аналіз наданих митницею даних виявив значну кількість пар, що задовольняють вибраним критеріям. Безумовно, не всі вони є парами «товар риску» - «товар прикриття». Ефективність реалізованого алгоритму може бути підтверджена лише в ході додаткових перевірок на митних постах. Проте слід зазначити, що число подібних пар істотно менше, ніж загальне число товарних груп, і їх список сповна може бути використаний як рекомендація по ретельнішому догляду певних вантажів. Як приклад приведемо одну пару товарів: шини для легкових автомобілів і протекторні заготовки для їх відновлення. Впродовж всього 2008 року ймовірність ввезення шин разом із заготовками дуже висока — в середньому 95% за рік. Випадків ввезення лише заготовок практично не було. При цьому коефіцієнт кореляції не настільки великий, оскільки чималий об'єм імпорту шин не супроводиться заготовками. Сам по собі факт кореляції між цими групами товарів досить природний, проте ставка митного збору на заготовки була в 5 разів нижче, ніж для шин. Більш того, порівняльний аналіз даних України і ЄС показав, що імпорт заготовок згідно з українськими даними майже в 200 разів вище, ніж за даними ЄС, а імпорт шин нижче в 3,5 рази, якщо порівнювати об'єми імпорту по вазі. При цьому сумарна вага імпорту по цих двох групах збігається за даними України і ЄС з точністю до 20%. Схожа картина спостерігається і у вартісному вираженні. Вартість ввезених в Україну заготовок в 30 разів вища, ніж вивезених з країн ЄС, тоді як шин, якщо судити за вартістю, що декларується, ввезено в 2,7 разу менше вивезеної кількості. Таким чином., судячи по приведеним даним, з великою вірогідністю протекторні заготовки використовувалися рядом імпортерів як прикриття для шин, що ввозилися. Втрати держави на митних зборах склали імовірно близько 7 млн. дол.

Відзначимо, що аналіз був проведений на повному об'ємі вантажних митних декларацій за 2008 рік, що складає більше 2 млн. декларацій із загальним числом товарів порядка 5 млн. Ясно, що аналіз такої кількості даних

не може бути виконаний ні вручну, ні за допомогою ряду інших технологій підтримки рішень. І хоча, безумовно, неможливо повністю замінити аналітика автоматизованою системою, вживання методів пошуку знань дозволяє відсіяти величезну кількість даних, що не представляють інтересу і скоротити об'єм аналізованої інформації до рівня адекватного людському сприйняттю.

Задачі медицини. Сучасний рівень розвитку медицини характеризується тим, що при здійсненні практично будь-якого різновиду лікувального процесу збирається велика кількість супутньої інформації: результати аналізів та обстежень, протоколи оперативних втручань та оглядів. Особливостями цієї інформації є:

- Велика кількість атрибутів (у тому числі числових).
- Зашумленість. Наявність помилкових значень, викликаних ручним введенням даних або погрішностями апаратури.
- Часткова заповненість. Наявність незаповнених полів в дослідженнях або відсутність цілих досліджень у деяких пацієнтів.
- Погана структурованість. Вихідна інформація зберігається в різних типах СУБД, файлах, на паперових носіях і, як правило, складається з наборів моніторингів та аналізів різної структури.

Таким чином, актуальним завданням є розробка ефективних методів і алгоритмів виявлення залежностей у вигляді, доступному інтерпретації людині, між значеннями атрибутів даних, що володіють перерахованими властивостями, та реалізація вказаних методів і алгоритмів в програмних системах, що забезпечують всі фази обробки даних, включаючи: автоматизацію збору даних, первинну обробку і нормалізацію, інтерактивну взаємодію з експертом, візуалізацію результатів. Слід зазначити, що, весь набір інформації, що супроводжує лікувальний процес, в більшості випадків можна розділити на 2 групи: вхідна інформація (результати обстежень та моніторингів, отриманих до і під час дії на пацієнта, характеристики дії на пацієнта), вихідна інформація (результати обстежень у відновний період). Кожна з груп зазвичай містить значне число атрибутів.

В якості основного математичного апарату виявлення залежностей були використані числові асоціативні правила. Через наявність в даних прикладної області двох груп «вхідних» і «вихідних» атрибутів була вибрана методика використання асоціативних правил з обмеженнями на місце розташування атрибуту в правилі. Методика передбачає можливість явно задати для будь-якого атрибуту обмеження на його положення в лівій або правій частині асоціативного правила. Дане обмеження використовується в алгоритмі Apriori для оптимізації фази генерації наборів значень атрибутів, що часто зустрічаються. Окрім цього в системі реалізована можливість явно задавати на множині атрибутів відношення приналежності до груп залежних атрибутів. Групи можуть вибиратися експертами-медиками, ґрунтуючись на теоретичних знаннях про взаємозв'язки між деякими аналізами та дослідженнями. Планується реалізувати з використанням методів кореляційного аналізу можливість автоматизованого виявлення залежних атрибутів в групах атрибутів лівої і правої частин і виключення з результуючого набору правил, що містять їх комбінації.

У медицині, як ні в якій іншої прикладної області, велика роль теоретичних знань, накопичених за час її розвитку. В нашому випадку прикладом таких медичних знань разом з апріорі відомими взаємозв'язками між деякими аналізами є діапазони значень норм різних показників аналізів. Для інтеграції цих знань в процес виявлення асоціативних правил в системі існує спеціальний інструментарій, що дозволяє медичному експертові задавати лінгвістичні змінні декількох різновидів на множині значень атрибутів аналізованих даних. Результуючі асоціативні правила описуватимуть взаємозв'язки між заданими лінгвістичними змінними. Системою підтримуються лінгвістичні змінні у вигляді чітких і нечітких інтервалів. Останні знаходять в медицині широке вживання через нечіткість багатьох її понять.

Однією з ключових можливостей системи є автоматична дискретизація атрибутів за допомогою вдосконаленого алгоритму дискретизації RUDE, що

дозволяє знаходити семантично обґрунтоване розбиття на інтервали значень атрибутів, лінгвістичні змінні по яких не задані.

В якості базового алгоритму використовується алгоритм, що реалізовує метод Apriori пошуку асоціативних правил з використанням дерев. Алгоритм був модифікований для ефективної роботи з числовими атрибутами у фазі генерації наборів, що часто зустрічалися. Для валідації результатів роботи застосованих алгоритмів і методів використовується ідея синтаксичної близькості правил, заснованої на відстані між правилами:

$$D(r_1, r_2) = \delta_1 \mid (X_1 Y_1) \Theta (X_2 Y_2) \mid + \delta_2 \mid X_1 \Theta X_2 \mid + \delta_3 \mid Y_1 \Theta Y_2 \mid$$

Для порівняння двох наборів правил r_1 та r_2 по одному з критеріїв цікавості пропонується порівнювати їх пересічення $r_1 \cap r_2$ з кожним з множини, знаходити сумарне значення критерію цікавості для підмножин правил r_1 та r_2, що входять в пересічення, окремо експертним способом оцінювати цікавість і корисність правил, що не увійшли до пересічення.

Система реалізована у вигляді віконного застосування на платформі .NET. Зберігання моделей і взаємодія між ядром і призначеним для користувача інтерфейсом здійснюється з використанням стандарту PMML 3.0, розширеного для нечіткої дискретизації і обмежень на місце розташування атрибутів. Це дозволяє в перспективі інтегрувати дану систему з іншими системами інтелектуального аналізу даних.

Інтернет-торгівля. Для ефективного управління бізнесом у сфері електронної комерції великого поширення набувають методи бізнес-аналітики. У сферу їх застосування входять задачі по прогнозуванню об'ємів продажів, управлінню кількістю товарних запасів, визначенню оптимальних торгівельних націнок, виявленню типових паттернів купівельної поведінки, оптимізації навігації по сайту, поліпшенню рубрикації і тому подібне

Розглянемо вживання технологи асоціативних правил для виявлення закономірностей в даних книжкового інтернет-магазину. Всі використовувані для аналізу дані охоплюють період в півтора роки функціонування магазина і зберігаються в централізованій базі даних під управлінням СУБД Microsoft SQL

Server 2005. В якості технологічної платформи для аналізу даних був використаний Microsoft Analysis Services 2005.

Методом асоціативних правил були проаналізовані товари і товарні групи, що спільно входять в одну транзакцію (одне замовлення). Асоціативний аналіз спільного входження книжкових найменувань в одне замовлення виявив:

1. Правило входження книг одного і того ж автора в одне замовлення. Таким чином, для збільшення об'єму продажів, при замовленні відвідувачем сайту якої-небудь книги слід пропонувати йому книги того ж автора, що є в наявності (рис. 4.29).

Рис. 4.29. Відвідувачам сайту книг слід пропонувати книги того ж автора.

2. Правило спільних покупок дитячих книг в одному замовленні. Цей факт привів до виділення дитячої літератури в окрему групу, що збільшило продажі книг цієї групи на 7,5% (рис. 4.30).

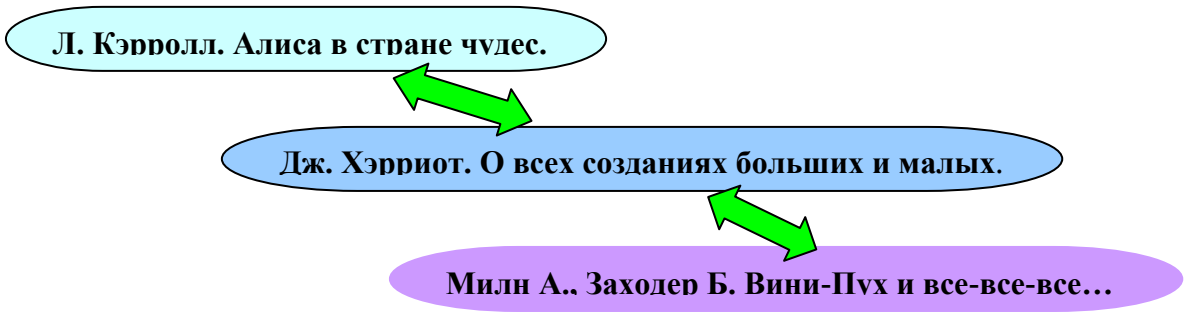


Рис.4.30. Правило спільних покупок дитячих книг в одному замовленні.

3. Правило спільних покупок езотеричної літератури. Цей факт привів до рекомендації про виділення книг про «правильне харчування», «особисте вдосконаленні» і так далі в окрему групу.

Асоціативний аналіз правил спільного входження товарних груп в одне замовлення не виявив значимих і достовірних правил по входженню товарів різних груп в одне замовлення, що говорить про адекватне існуюче розбитті книг на групи.

4. Послідовності категорій книг, що замовляються клієнтами.

Аналіз послідовностей книг, що замовляються клієнтами, показав, що якщо в замовлення клієнта входять книги з груп: зарубіжна поезія, комп'ютери і інтернет, біографії, мемуари та публіцистика, медицина, економіка, бізнес, та менеджмент, то в подальших замовленнях того ж клієнта зустрічаються книги з тієї ж групи.

Якщо клієнт замовив детективи, то в подальших замовленнях він замовляє або знову детективи, або зарубіжну прозу, або дитячу літературу. При замовленні клієнтом дитячої літератури в наступних замовленнях він, швидше за все, знову вибере книги з цієї групи, або купить зарубіжну прозу. Висновок: групи «зарубіжна проза», «дитяча література» і «детективи» повинні розташовуватися з точки зору навігації на сайті як можна ближче один до одного, що означає можливість швидкого переходу між цими тематичними групами.

Перспективні напрямки. Існує широкий спектр задач, в яких потрібно виявляти асоціації із збереженням об’єктно-ознакового опису даних. Це задачі біоінформатики по виявленню груп генів, що володіють загальними властивостями, пошук груп відвідувачів з схожими інтересами для рекомендаційних систем, виявлення інтернет-співтовариств, наукових співтовариств, задачі аналізу соціальних мереж, побудова автоматичних каталогів і рубрикаторів в інформаційних системах, пошук документів-дублікатів.

Задача пошуку асоціацій останніми роками набула велику популярність у зв'язку із зростаючою потребою в аналізі генетичних даних. Основне завдання такого аналізу — виявлення груп генів, що проявляють схожу поведінку лише за певних умов. Надалі фахівець генетик на підставі проведеного аналізу висуває гіпотезу про те, чи є дана група генів причиною досліджуваної хвороби. Окрім даних генної експресії, як додатки в біології можна привести проекти, які аналізували активність лікарських препаратів. Розглядаються і інші масиви генетичних даних, що в основному відносяться до ракових захворювань, таким як лімфома. Наприклад, для таких задач широке застосування має система Coron.

Система аналізу даних Coron призначена для пошуку частої множини ознак і асоціативних правил. Програма володіє непоганим графічним інтерфейсом, власним форматом даних, можливістю роботи з базами даних. Для пошуку частої множини ознак використовуються найбільш ефективні алгоритми співтовариства FIM. Пошук асоціативних правил також використовує ефективні алгоритми, що спираються на досягнення ФАП і виявилися корисними для компактного відображення правил і побудови їх базисів. Ще одним достоїнством продукту є вільний доступ і кросплатформенність.

Останніми роками поновився інтерес до міждисциплінарних досліджень в області аналізу соціальних мереж, в яких задіяні математична соціологія і інформатика, що спираються на апарат теорії графів. Зусилля в цьому напрямі в значній мірі підтримуються завдяки новим обчислювальним можливостям і доступності електронних даних для деяких соціальних систем: співтовариств вчених, людей, ведучих особисті електронні щоденники (weblogers), покупців інтернет-магазинів, мереж друзів, сайтів знайомств і так далі. Зокрема, в центрі багатьох поточних досліджень знаходяться мережі знань, тобто мережі взаємодій, в яких агенти виробляють знання або обмінюються ними. До числа досліджень входить виявлення співтовариств, що розглядається як знаходження агентів, які володіють множиною загальних ознак. Аналіз соціальних мереж

спеціалізується на методах виявлення, опису і правдоподібної організації різних видів соціальних співтовариств. Для аналізу соціальних аспектів співтовариств основний інтерес представляють лідери, периферійні члени, міжгрупова і внутрішньогрупова взаємодія.

Технологія асоціативних правил успішно застосовувалися для аналізу епістемічних співтовариств (тобто агентів, що мають справу з однаковими темами, наприклад, наукові співтовариства або користувачі блогів) або філіальних мереж (актори належать одній і тій же організації). Успіх цього підходу обумовлений наявністю таксономій, що виявляється корисним при ієрархічному описі груп акторів в термінах схожості інтересів.

Методи пошуку асоціацій можуть бути використані для так званої коллаборативної фільтрації (collaborative filtering) при виявленні груп покупців зі схожими перевагами у вигляді деякої підмножини товарів (задача цільового маркетингу). Схожа ситуація має місце в рекомендаційних системах, де асоціативні правила надають інформацію про схожі інтереси груп відвідувачів. Необхідно відзначити, що рекомендаційні системи і цільовий маркетинг - важливі застосування в області електронної комерції. У таких проектах основною метою є виявлення груп покупців, ведучих себе схожим чином, аби передбачити їх інтереси і запропонувати адекватні рекомендації. Іншим прикладом є ринок інтернет-реклами, для якого актуальний пошук асоціацій, що представляють окремі ринки, тобто множину покупців і придбаних ними рекламних словосполук.

Існують і менш поширені застосування теорії асоціативних правил, що спираються, наприклад, на дані про голосування. В цьому випадку необхідно виявляти підмножини рядків виборців, що дотримуються схожих політичних поглядів і що проявляють схожу електоральну поведінку на підмножині даних ознак.

4.4. Дерева рішень - загальні принципи технології.

Стрімкий розвиток інформаційних технологій, зокрема, прогрес в методах збору, зберігання і обробки даних дозволив багатьом організаціям збирати величезні масиви даних, які необхідно аналізувати. Об'єми цих даних настільки великі, що можливостей експертів вже не вистачає, що породило попит на методи автоматичного дослідження (аналізу) даних, який з кожним роком постійно збільшується. Дерева рішень (decision trees) – один з таких методів автоматичного аналізу даних. Ця технологія є однією з найбільш популярних методів вирішення задач класифікації і прогнозування. Інколи цей метод інтелектуального аналізу даних також називають деревами рішающих правил або деревами класифікації і регресії. Як видно з останньої назви, за допомогою даного методу вирішуються задачі класифікації і прогнозування. Якщо залежна, тобто цільова змінна набуває дискретних значень, при допомозі методу дерева рішень вирішується задача класифікації. Якщо ж залежна змінна набуває безперервних значень, то дерево рішень встановлює залежність цієї змінної від незалежних змінних, тобто вирішує задачу чисельного прогнозування.

Перші ідеї створення дерев рішень запропоновані в роботах Ховленда (Hoveland) і Ханта (Hunt) в кінці 50-х років XX століття. Проте, основополагаючою роботою, що дала імпульс для розвитку цього напряму, з'явилася книга Ханта (Hunt E.B.), Меріна (Marin J.) і Стоуна (Stone P.J) «Experiments in Induction», що побачила світло в 1966 р. [9, 37, 47]

Означення. Дерева рішень – це технологія представлення правил в ієрархічній, послідовній структурі, де кожному об'єкту відповідає єдиний вузол, що дає рішення. Під правилом розуміється логічна конструкція, представлена у вигляді «Якщо ... то ...» (if-then).

Для ухвалення рішення, до якого класу віднести деякий об'єкт або ситуацію, потрібно відповісти на питання, що стоять у вузлах цього дерева, починаючи з його кореня. Питання мають вигляд «значення параметра А

більше x?». Якщо відповідь позитивна, здійснюється перехід до правого вузла наступного рівня, якщо негативна - то до лівого вузла; потім знову слідує питання, пов'язане з відповідним вузлом (рис. 4.31).

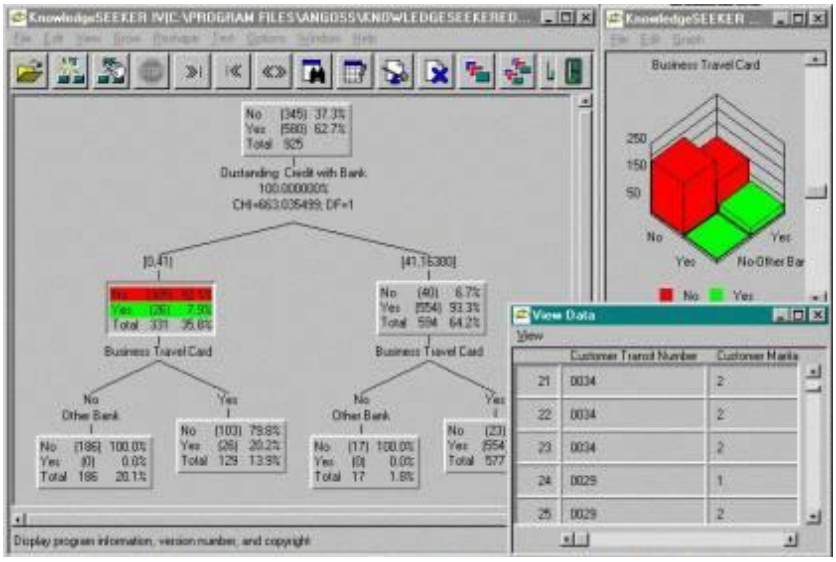


Рис 4.31. Система KnowledgeSeeker обробляє банківську інформацію.

Більшість систем інтелектуального аналізу даних використовують цей метод для пошуку закономірностей. Найвідомішими є See5/C5.0 (RuleQuest, Австралія), Clementine (Integral Solutions, Великобританія), SIPINA (University of Lyon, Франція), IDIS (Information Discovery, США), KnowledgeSeeker (ANGOSS, Канада).

Сфера застосування дерев рішень в даний час широка, але всі задачі, що вирішуються цією технологією можуть бути об'єднані в наступні три класи:

1. Опис даних: Дерева рішень дозволяють зберігати інформацію про дані в компактній формі. Замість самих даних ми можемо зберігати дерево рішень, яке містить точний опис об'єктів.

2. Класифікація: Дерева рішень відмінно справляються із задачами класифікації, тобто віднесення об'єктів до одного із заздалегідь відомих класів.

3. Регресія: Якщо цільова змінна має безперервні значення, дерева рішень дозволяють встановити залежність цільової змінної від незалежних (вхідних) змінних.

Дерева рішень є прекрасним інструментом в системах підтримки прийняття рішень та інтелектуального аналізу даних. В склад багатьох пакетів, призначених для інтелектуального аналізу даних, вже включені методи побудови дерев рішень. У областях, де висока ціна помилки, вони служать відмінною підмогою аналітика або керівника. Дерева рішень успішно застосовуються для вирішення практичних задач в наступних областях:

- *Банківська справа*. Оцінка кредитоспроможності клієнтів банку при видачі кредитів.
- *Промисловість*. Контроль за якістю продукції (виявлення дефектів), випробування без руйнувань (наприклад перевірка якості зварки) і так далі**.**
- *Медицина*. Діагностика різних захворювань.
- *Молекулярна біологія*. Аналіз будови амінокислот.

Це далеко не повний список областей де можна використовувати дерева рішень. Не досліджено ще багато потенційних сфер застосування [9, 37, 47].

Розглянемо сутність цієї технології на прикладі дерева рішень, завдання якого - відповісти на питання: «Чи грати в гольф?» Аби вирішити задачу, тобто прийняти рішення, чи грати в гольф, слід віднести поточну ситуацію до одного з відомих класів (в даному випадку – «грати» або «не грати»). Для цього потрібно відповісти на низку запитань, які знаходяться у вузлах цього дерева, починаючи з його кореня (рис. 4.32).

Перший вузол нашого дерева «Сонячно?» є вузлом перевірки, тобто умовою. При позитивній відповіді на питання здійснюється перехід до лівої частини дерева, званою лівою гілкою, при негативному - до правої частини дерева. Таким чином, внутрішній вузол дерева є вузлом перевірки певної умови. Далі йде наступне питання і так далі, поки не буде досягнутий кінцевий вузол дерева, що є вузлом рішення. Для нашого дерева існує два типи кінцевого вузла: «грати» і «не грати» в гольф. В результаті проходження від кореня

дерева (інколи званого кореневою вершиною) до його вершини вирішується задача класифікації, тобто вибирається один з класів – «грати» і «не грати» в гольф.

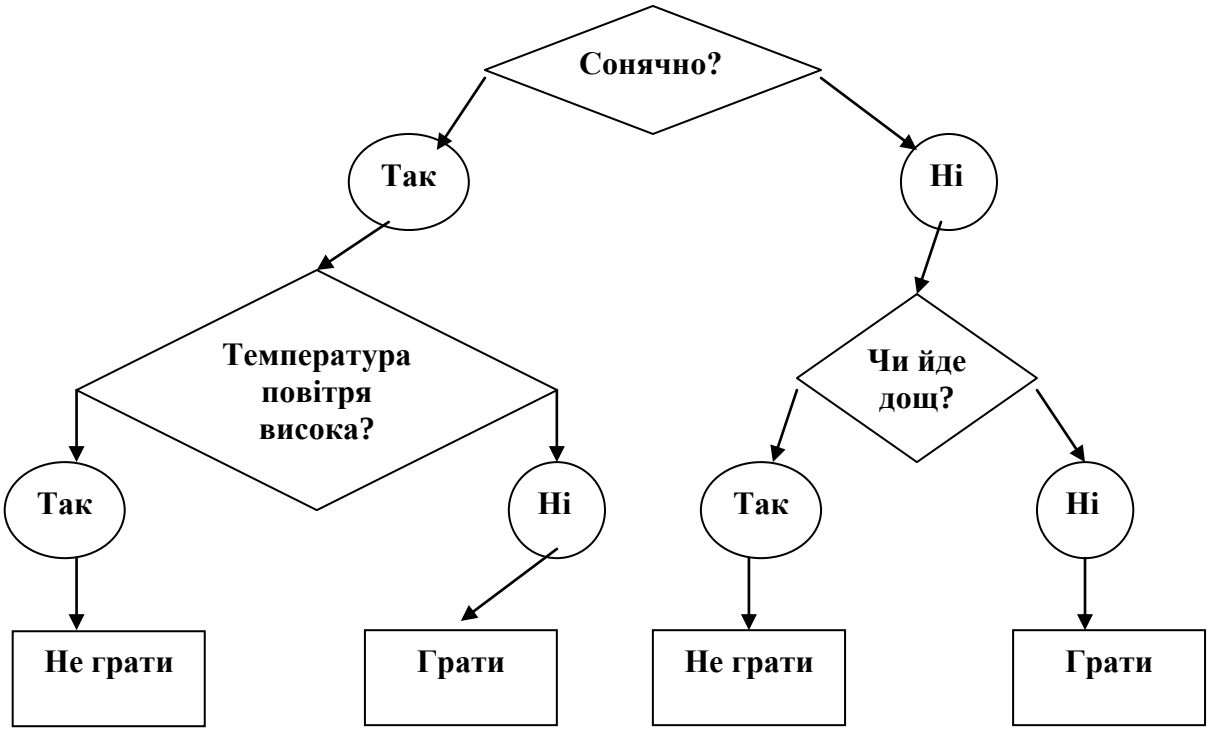


Рис. 4.32. Дерево рішень задачі «Чи грати в гольф?».

Метою побудови дерева рішення в нашому випадку є визначення значення категоріальної залежної змінної. Отже, для нашої задачі основними елементами дерева рішень є: корінь дерева – «Сонячно?», внутрішній вузол дерева або вузол перевірки – «Температура повітря висока?», «Чи йде дощ?», лист, кінцевий вузол дерева, вузол рішення або вершина: «Грати», «Не грати», гілка дерева (випадки відповіді) – «Так», «Ні».

У розглянутому прикладі вирішується задача бінарної класифікації, тобто створюється дихотомічна класифікаційна модель. Приклад демонструє роботу так званих бінарних дерев. У вузлах бінарних дерев галуження може вестися лише в двох напрямах, тобто існує можливість лише двох відповідей на поставлене питання («так і ні»). Такі дерева є найпростішими, окремим

випадком дерев рішень. В інших випадках, відповідей і, відповідно, гілок дерева, що виходять з його внутрішнього вузла, може бути більше двох.

Розглянемо складніший приклад. База даних, на основі якої повинне здійснюватися прогнозування, містить наступні ретроспективні дані про клієнтів банку, що є її атрибутами: вік, наявність нерухомості, освіта, середньомісячний дохід, чи повернув клієнт вчасно кредит. Завдання полягає в тому, аби на підставі перерахованих вище даних (окрім останнього атрибуту) визначити, чи варто видавати кредит новому клієнтові. Така задача, як вже відомо, вирішується в два етапи: побудова класифікаційної моделі і її використання.

На етапі побудови моделі, власне, і будується дерево класифікації або створюється набір деяких правил. На етапі використання моделі побудоване дерево, або дорога від його кореня до однієї з вершин, той, що є набором правил для конкретного клієнта, використовується для відповіді на поставлене питання чи «Видавати кредит?» Правилом є логічна конструкція, представлена у вигляді «якщо : то :».

На рис. 4.33. наведений приклад дерева класифікації, за допомогою якого вирішується задача «Видавати кредит клієнтові?». Вона є типовою задачею класифікації, і за допомогою дерев рішень отримують досить хороші варіанти її рішення.

Як бачимо, внутрішні вузли дерева (вік, наявність нерухомості, дохід і освіта) є атрибутами описаної вище бази даних. Ці атрибути називають такими, що прогнозують, або атрибутами розщеплювання (splitting attribute). Кінцеві вузли дерева, або листи, іменуються мітками класу, що є значеннями залежної категоріальної змінної «видавати» або «не видавати» кредит. Кожна гілка дерева, що йде від внутрішнього вузла, відмічена предикатом розщеплювання. Останній може відноситися лише до одного атрибуту розщеплювання даного вузла. Характерна особливість предикатів розщеплювання: кожен запис використовує унікальну дорогу від кореня дерева лише до одного вузла-рішення. Об'єднана інформація про атрибути розщеплювання і предикати

розщеплювання у вузлі називається критерієм розщеплювання (splitting criterion).

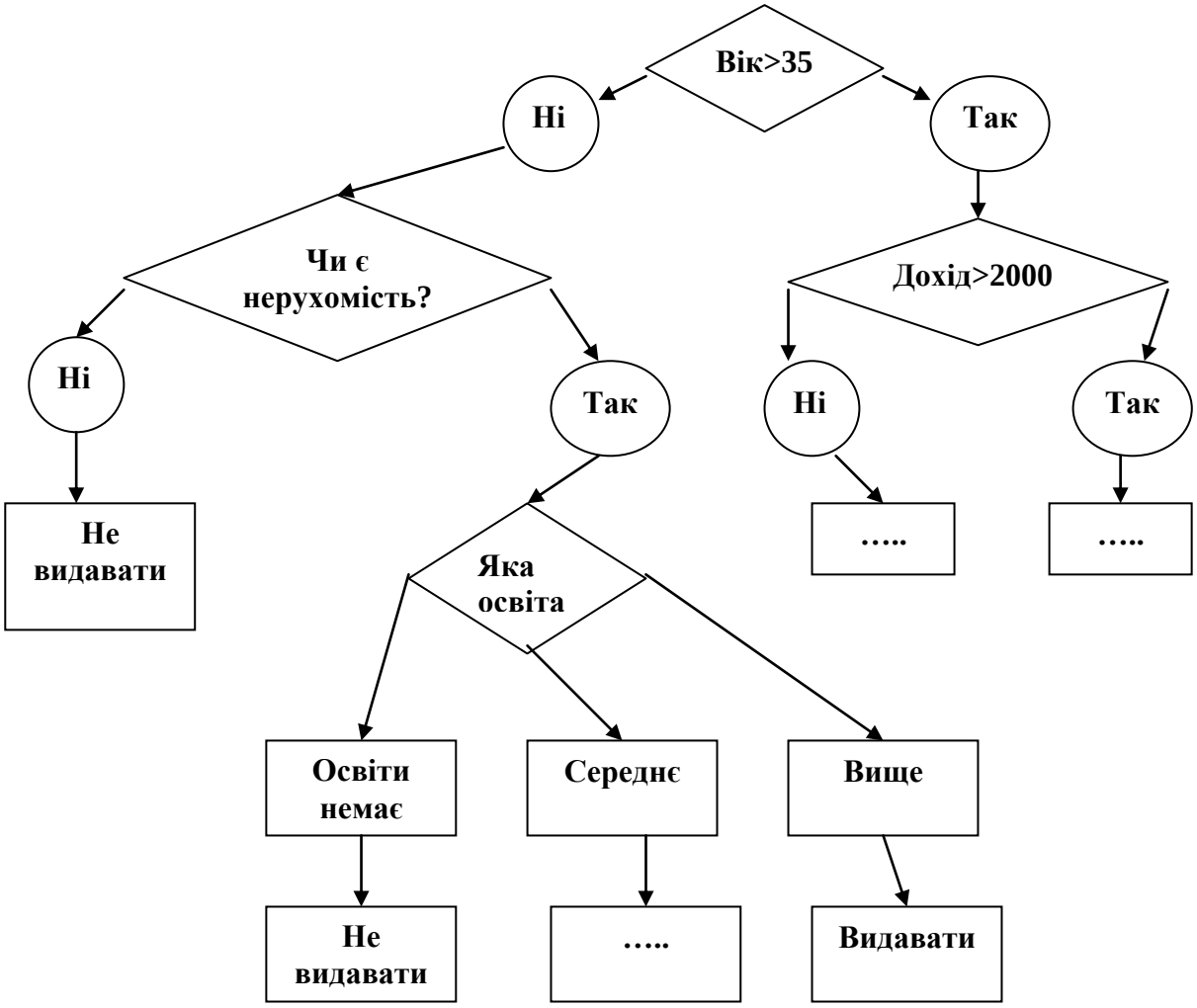


Рис. 4.33. Дерево рішень «Чи видавати кредит?».

Слід зазначити, що на рис. 4.33 змальоване лише одне з можливих дерев рішень для даної бази даних. Наприклад, критерій розщеплювання «Яка освіта?», міг би мати два предикати розщеплювання і виглядати інакше: освіта «вища» і «не вища». Тоді дерево рішень мало б інший вигляд. Таким чином, для даної задачі (як і для будь-якої іншої) може бути побудована множина дерев рішень різної якості, з різною прогнозуючою точністю.

Якість побудованого дерева рішення вельми залежить від правильного вибору критерію розщеплювання. Над розробкою і удосконаленням критеріїв працюють багато дослідників. Метод дерев рішень часто називають «наївним»

підходом. Але завдяки цілому ряду переваг, даний метод є одним з найбільш популярних для вирішення задач класифікації.

Розглянемо процес конструювання дерева рішень. Хай нам задана деяка навчальна множина T, що містить об'єкти (приклади), кожен з яких характеризується m атрибутами, причому один з них вказує на приналежність об'єкту до певного класу. Ідею побудови дерев рішень з множини T, вперше висловлену Хантом, приведемо по Р. Куїнлену (R. Quinlan).

Хай через $\{C_1, C_2, ..., C_k\}$ позначені класи (значення мітки класу), тоді існують 3 ситуації:

1. множина T містить один або більше прикладів, що відносяться до одного класу C_k. Тоді дерево рішень для T – це лист, що визначає клас C_k;

2. множина T не містить жодного прикладу, тобто порожня множина. Тоді це знову лист, і клас, що асоціюється з листом, вибирається з іншої множини відмінної від T, скажімо, з множини, що асоціюється з родителем;

3. множина T містить приклади, що відносяться до різних класів. В цьому випадку слід розбити множину T на деякі підмножини. Для цього вибирається одна з ознак, що має два і більше відмінних один від одного значень $O_1, O_2, ..., O_n$. Множина T розбивається на підмножини $T_1, T_2, ..., T_n$, де кожна підмножина T_i містить всі приклади, що мають значення O_i для вибраної ознаки. Це процедура рекурсивно продовжуватиметься до тих пір, поки кінцева множина не складатиметься з прикладів, що відносяться до одного і тому ж класу.

Вищеописана процедура лежить в основі багатьох сучасних алгоритмів побудови дерев рішень, цей метод відомий ще під назвою *розділення і захвату* (divide and conquer). Вочевидь, що при використанні даної методики, побудова дерева рішень буде відбувається зверху вниз. Оскільки всі об'єкти були заздалегідь віднесені до відомих нам класів, такий процес побудови дерева рішень називається *навчанням з вчителем* (supervised learning). Процес навчання також називають *індуктивним навчанням* або індукцією дерев (tree induction).

Алгоритми конструювання дерев рішень складаються з етапів «побудови» або «створення» дерева (tree building) і «скорочення» дерева (tree pruning). В ході створення дерева вирішуються питання вибору критерію розщеплювання і зупинки навчання (якщо це передбачено алгоритмом). В ході етапу скорочення дерева вирішується питання відсікання деяких його гілок. Розглянемо ці питання детальніше.

Критерій розщеплювання. Процес створення дерева відбувається зверху вниз, тобто є низхідним. В ході процесу алгоритм повинен знайти такий критерій розщеплювання, інколи також званий критерієм розбиття, аби розбити множину на підмножини, які б асоціювалися з даним вузлом перевірки. Кожен вузол перевірки має бути помічений певним атрибутом. Існує правило вибору атрибуту: він повинен розбивати вихідну множину даних так, щоб об'єкти підмножин, що отримуються в результаті цього розбиття, були представниками одного класу або ж були максимально наближені до такого розбиття. Остання фраза означає, що кількість об'єктів з інших класів, так званих «домішок», в кожному класі повинно прагнути до мінімуму.

Існують різні критерії розщеплювання. Найбільш відомі - міра ентропії і індекс Gini.

У деяких методах для вибору атрибуту розщеплювання використовується так звана міра інформативності підпросторів атрибутів, яка ґрунтується на підході ентропії і відома під назвою «міра інформаційного виграшу» (information gain measure) або міра ентропії.

$$Gain(X) = Info(T) - Info_X(T),$$

де $Info(T)$ - ентропія множини T, а $Info_X(T) = \sum_{i=1}^{n} \frac{|T_i|}{|T|} * Info(T_i)$.

Множини $T_1, T_2, ..., T_n$ отримані при розбитті вихідної множини T по перевірці X. Вибирається атрибут, що дає максимальне значення по критерію. Вперше цей захід був запропонований Р. Куїнленом в розробленому їм алгоритмі ID3.

Інший критерій розщеплювання, запропонований Брейманом (Breiman), реалізовано в алгоритмі CART і називається індексом Gini (на честь

італійського економіста Corrado Gini). За допомогою цього індексу атрибут вибирається на підставі відстаней між розподілами класів. Якщо дана множина T, що включає приклади з n класів, індекс Gini, тобто gini(T), визначається по формулі:

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2,$$

де T - поточний вузол, p_j - імовірність класу j у вузлі T, n - кількість класів.

Велике дерево не означає, що воно «підходяще». Чим більше окремих випадків описано в дереві рішень, тим менша кількість об'єктів потрапляє в кожен окремий випадок. Такі дерева називають «гіллястими» або «кущистими», вони складаються з невиправдано великого числа вузлів і гілок, вихідна множина розбивається на велике число підмножин, що складаються з дуже малого числа об'єктів. В результаті «переповнювання» таких дерев їх здібність до узагальнення зменшується, і побудовані моделі не можуть давати вірні відповіді.

В процесі побудови дерева, аби його розміри не стали надмірно великими, використовують спеціальні процедури, які дозволяють створювати оптимальні дерева, так звані дерева «відповідних розмірів». Який розмір дерева може вважатися оптимальним? Дерево має бути достатнє складним, аби враховувати інформацію з досліджуваного набору даних, але одночасно воно має бути достатнє простим. Іншими словами, дерево повинне використовувати інформацію, поліпшуючу якість моделі, і ігнорувати ту інформацію, яка її не покращує. Тут існує дві можливі стратегії. Перша полягає в нарощуванні дерева до певного розміру відповідно до параметрів, заданих користувачем. Визначення цих параметрів може ґрунтуватися на досвіді і інтуїції аналітика, а також на деяких «діагностичних повідомленнях» системи, що конструює дерево рішень. Друга стратегія полягає у використанні набору процедур, що визначають «відповідний розмір» дерева, вони розроблені Бріманом, Куїлендом і ін. в 1984 році. Проте, як відзначають автори, не можна сказати, що ці процедури доступні початкуючому користувачеві. Процедури, які

використовують для запобігання створення надмірно великих дерев, включають: скорочення дерева шляхом відсікання гілок; використання правил зупинки навчання. Слід зазначити, що не всі алгоритми при конструюванні дерева працюють за однією схемою. Деякі алгоритми включають два окремі послідовні етапи: побудова дерева і його скорочення; інші чергують ці етапи в процесі своєї роботи для запобігання нарощуванню внутрішніх вузлів.

Зупинка побудови дерева. Розглянемо правило зупинки. Воно повинне визначити, чи є даний вузол внутрішнім вузлом, при цьому він розбиватиметься далі, або ж він є кінцевим вузлом, тобто вузлом рішенням.

Означення. Зупинка - такий момент в процесі побудови дерева, коли слід припинити подальші галуження.

Один з варіантів правил зупинки використовує статистичні методи для оцінки доцільності подальшого розбиття – «рання зупинка» (prepruning), вона визначає доцільність розбиття вузла. Перевага використання такого варіанту - зменшення часу на навчання моделі. Проте тут виникає ризик зниження точності класифікації. Тому рекомендується замість зупинки використовувати відсікання.

Другий варіант зупинки навчання - обмеження глибини дерева. В цьому випадку побудова закінчується, якщо досягнута задана глибина. Ще один варіант зупинки - завдання мінімальної кількості прикладів, які міститимуться в кінцевих вузлах дерева. При цьому варіанті галуження тривають до того моменту, поки всі кінцеві вузли дерева не будуть чистими або міститимуть не більше ніж задане число об'єктів. Існує ще ряд правил, але слід зазначити, що жодне з них не має великої практичної цінності, а деякі можуть застосовуватися лише в окремих випадках.

Скорочення дерева або відсікання гілок. Дуже часто алгоритми побудови дерев рішень дають складні дерева, які мають багато вузлів і гілок. Такі дерева дуже важко зрозуміти. До того ж гіллясте дерево, що має багато вузлів, розбиває навчальну множину на досить велику кількість підмножин, що складаються зі все меншої кількості об'єктів. Цінність правила, при цьому,

вкрай низьке, і в цілях аналізу даних таке правило практично непридатне. Багато важливіше мати дерево, що складається з малої кількості вузлів, яким би відповідала велика кількість об'єктів з навчальної вибірки. І тут виникає питання: а чи не побудувати всі можливі варіанти дерев, які відповідають навчальній множині, і з них вибрати дерево з найменшою глибиною? На жаль, ця задача є NP-повною, і, як відомо, цей клас задач не має ефективних методів рішення.

Вирішенням проблеми дуже гіллястого дерева є його скорочення шляхом відсікання (pruning) деяких гілок. Якість класифікаційної моделі, побудованої за допомогою дерева рішень, характеризується двома основними ознаками: точністю розпізнавання і помилкою. Точність розпізнавання розраховується як відношення об'єктів, правильно класифікованих в процесі навчання, до загальної кількості об'єктів набору даних, які брали участь в навчанні. Помилка розраховується як відношення об'єктів, неправильно класифікованих в процесі навчання, до загальної кількості об'єктів набору даних, які брали участь в навчанні.

Відсікання гілок або заміну деяких гілок піддеревом слід проводити там, де ця процедура не приводить до зростання помилки. Процес проходить від низу до верху. Це популярніша процедура, ніж використання правил зупинки. Дерева, що отримуються після відсікання деяких гілок, називають усіченими. Якщо таке усічене дерево все ще не є інтуїтивним і складно для розуміння, використовують витягання правил, які об'єднують в набори для опису класів. Кожна дорога від кореня дерева до його вершини або листа дає одне правило. Умовами правила є перевірки на внутрішніх вузлах дерева. Хоча відсікання не є панацеєю, але в більшості практичних задач дає добрі результати, що дозволяє говорити про правомірність використання подібної методики (рис. 4.34).

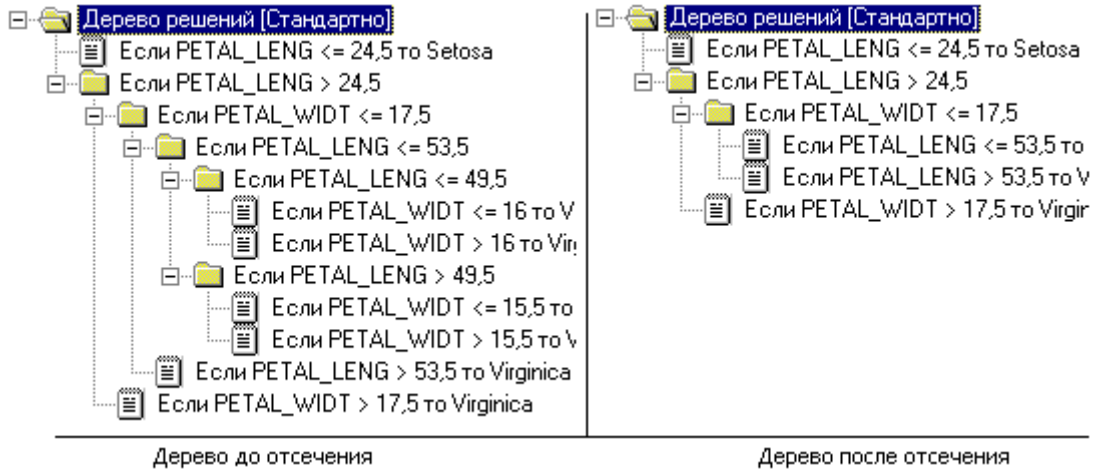


Рис. 4.34. Процедура відсікання деяких гілок.

У користувачів систем інтелектуального аналізу даних досить часто виникає питання: чи є якась суттєва відмінність алгоритму дерева рішень від асоціативних правил в задачах класифікації. Нагадаємо, що алгоритм асоціативних правил призначений для виявлення в даних залежностей типа $A \geq B$, де A і B є наборами пар атрибут = значення. Алгоритм дерева рішень призначений для вирішення задач регресії і класифікації, тобто для виявлення залежності цільового параметра від значення інших параметрів. Так само, як і у випадку з алгоритмом асоціативних правил, алгоритм дерева рішень дозволяє виявляти правила в даних типа $A \geq B$. Для цього в якості лівої частини правила (умови A) треба об'єднати умови по всіх вузлах у вітці дерева від кореня до листа, що містить опис умов на цільову змінну B.

Не дивлячись на схожість вирішуваної задачі і форму представлення результату, правила, отримувані обома алгоритмами, можуть істотно розрізнятися. Вся річ у тому, що процес побудови дерева рішень заснований на максимізації приросту інформації, тоді як алгоритм асоціативних правил заснований на виділенні частих наборів, в яких частота появи однієї частини (права частина правила) відносно наборів, в яких є інша частина (ліва частина правила), висока. Виходячи з приведеного аналізу відмінності між алгоритмами, ми можемо побудувати приклад, що наводить до принципово різних результатів.

Передбачимо, що магазин містить в асортименті три товари: товар А, товар Б і товар В Допустимо, що ми хочемо виділити правила, які передбачають факт покупки товару А. Побудуємо дерево рішень. Зі всіх транзакцій (10000) одна частина (6000) містить товар А, а інша (4000) не містить. Далі алгоритм дерева рішень повинен перевірити розбиття транзакцій за фактом наявності товару Б і наявність товару В. Кінцеве розбиття буде здійснено по атрибуту, що найбільшою мірою збільшує приріст інформації за фактом покупки товару А.

Розглянемо спочатку можливість розбиття за фактом покупки товару Б. Передбачимо, що товар Б міститься в 8000 транзакцій. У одній частині цих транзакцій (4800) міститься товар А, а в іншій (3200) - ні. Далі, в 2000 транзакціях з тих, що не містять товар Б, частина (1200) містить товар А, а частина (800) - ні. Варіант дерева рішень має слідуючий вигляд (рис. 4.35):

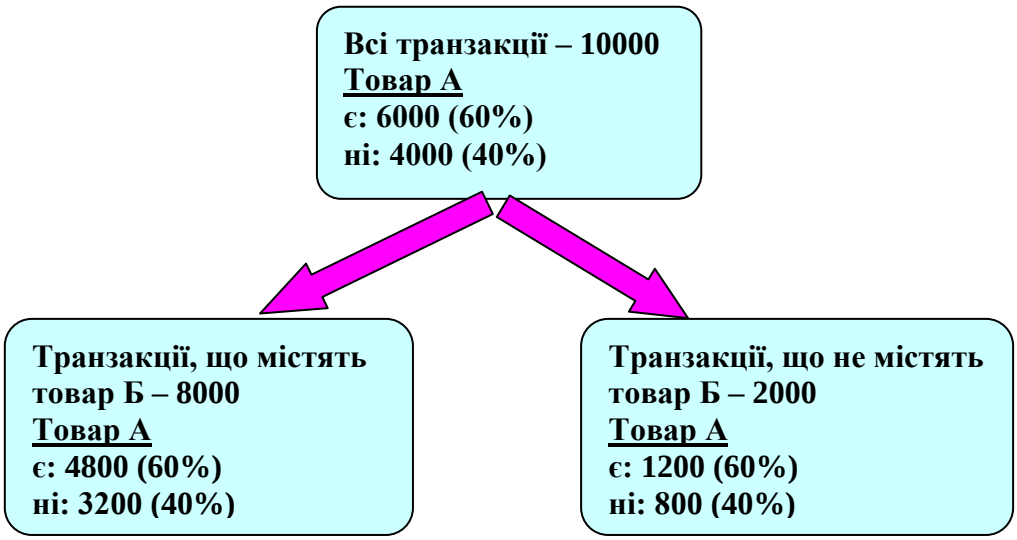


Рис. 4.35. Дерево рішень факту покупки товару А (варіант 1).

Таким чином, розбиття по наявності товару Б в транзакції не збільшує приріст інформації (розподіл транзакцій, що містять товар А зберігається після розбиття за фактом покупки товару Б). Тобто алгоритм дерева рішень не здійснюватиме розбиття за ознакою наявності товару Б в транзакції. При цьому,

алгоритм асоціативних правил швидше за все сформулює правило вигляду товар Б ≥ товар А, оскільки транзакції, що містять одночасно товар А і товар Б є дуже частими (4800 або 48% транзакцій), а точність дотримання правила складає 60%, що часто входить в порогове значення аналізованих правил в алгоритмі асоціативних правил.

Тепер розглянемо варіант побудови дерева рішень з розбиттям за фактом покупки товару В. Передбачимо, що 1000 транзакцій містить товар В, а ті 9000, що залишилися - ні. З 1000 транзакцій, що містять товар В, 900 транзакцій містить товар А, а 100 - ні. З 9000 транзакцій, що не містять товар В, 5100 транзакцій містять товар А, а 3900 - ні. Нижче побудоване відповідне дерево рішень (рис. 4.36).

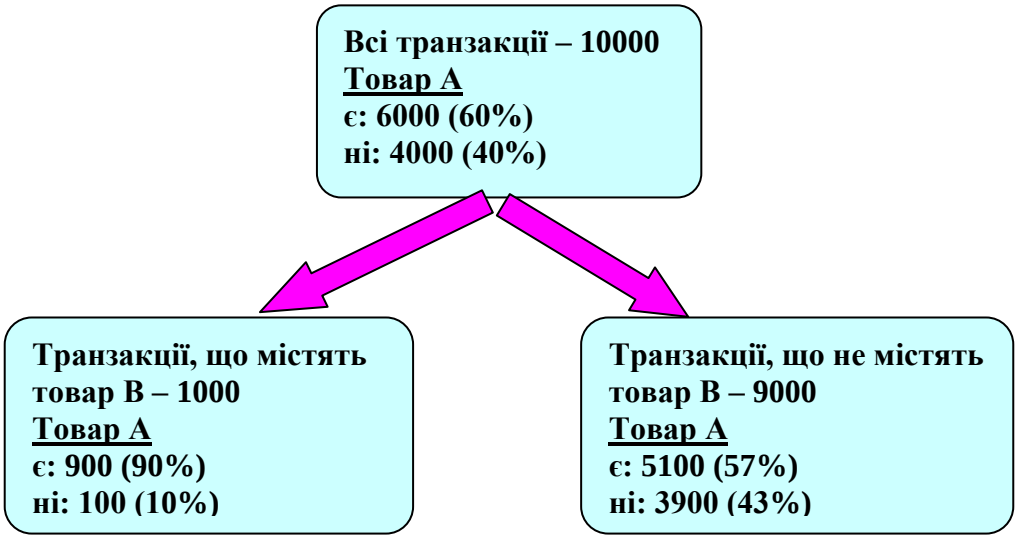


Рис. 4.36. Дерево рішень факту покупки товару А (варіант 2).

Розбиття за ознакою наявності товару В дає в разі позитивної відповіді значний приріст інформації: 90% транзакцій, що містять товар В, містять товар А, тобто дерево рішень сформулює правило товар В ≥ товар А. Алгоритм асоціативних правил може не дати аналогічного правила, оскільки транзакції, що одночасно містять товари А і В, складають лише 9% від загального числа транзакцій, що може не пройти поріг підтримки правила в алгоритмі.

Таким чином, на даних сконструйованого прикладу, алгоритм дерева рішень дасть правило товар В ≥ товар А, а алгоритм асоціативних правил - товар Б ≥ товар А. Вибір алгоритму виявлення правил в даних може істотно вплинути на результат аналізу. Можлива відмінність в результатах пов'язана з особливістю реалізації кожного з описаних алгоритмів і не є помилкою одного з них. В зв'язку з цим має сенс рекомендувати їх спільне вживання і використання експертного знання для вибору найбільш адекватного для кожної конкретної моделі.

Розглянувши основні проблеми, що виникають при побудові дерев, було б несправедливо не згадати про їх достоїнства.

1. *Швидкий процес навчання*. На побудову класифікаційних моделей за допомогою алгоритмів конструювання дерев рішень потрібно значно менше часу, чим, наприклад, на навчання нейронних мереж. Більшість алгоритмів конструювання дерев рішень мають можливість спеціальної обробки пропущених значень. Багато класичних статистичних методів, за допомогою яких вирішуються задачі класифікації, можуть працювати лише з числовими даними, тоді як дерева рішень працюють і з числовими, і з категоріальними типами даних.

2. *Генерація правил в областях, де експертові важко формалізувати свої знання*. Дерева рішень дозволяють створювати класифікаційні моделі в тих областях, де аналітику досить складно формалізувати знання. Алгоритм конструювання дерева рішень не вимагає від користувача вибору вхідних атрибутів (незалежних змінних). На вхід алгоритму можна подавати всі існуючі атрибути, алгоритм сам вибере найбільш значимі серед них, і лише вони будуть використані для побудови дерева. У порівнянні, наприклад, з нейронними мережами, це значно полегшує користувачеві роботу, оскільки в нейронних мережах вибір кількості вхідних атрибутів істотно впливає на час навчання.

3. *Витягання правил на природній мові*. Дерева рішень дають можливість витягувати правила з бази даних на природній мові. Приклад правила: «Якщо Вік > 35 і Дохід > 200, то видати кредит».

4. *Інтуїтивно зрозуміла класифікаційна модель*. Класифікаційна модель, представлена у вигляді дерева рішень, є інтуїтивною і спрощує розуміння вирішуваного завдання. Результат роботи алгоритмів конструювання дерев рішень, на відмінність, наприклад, від нейронних мереж, що є «чорними ящиками», легко інтерпретується користувачем. Ця властивість дерев рішень не лише важлива при віднесенні до певного класу нового об'єкту, але і корисна при інтерпретації моделі класифікації в цілому. Дерево рішень дозволяє зрозуміти і пояснити, чому конкретний об'єкт відноситься до того або іншого класу.

5. *Висока точність прогнозу, порівняно з іншими методами*. Точність моделей, створених за допомогою дерев рішень значно вища, порівняно з іншими методами побудови класифікаційних моделей (статистичні методи, нейроні мережі та інше). Розроблений ряд масштабованих алгоритмів, які можуть бути використані для побудови дерев рішень на надвеликих базах даних; масштабність тут означає, що із зростанням числа прикладів або записів бази даних час, що витрачається на навчання, тобто побудова дерев рішень, зростає лінійно. Приклади таких алгоритмів: SLIQ, SPRINT.

6. *Побудова непараметричних моделей*. Багато статистичних методів є параметричними, і користувач повинен заздалегідь володіти певною інформацією, наприклад, знати вигляд моделі, мати гіпотезу про вигляд залежності між змінними, передбачати, який вигляд розподілу мають дані. Дерева рішень, на відміну від таких методів, будують непараметричні моделі. Таким чином, дерева рішень здатні вирішувати такі задачі інтелектуального аналізу даних, в яких відсутня апріорна інформація про вигляд залежності між досліджуваними даними.

Через ці і багато інших причин, методологія дерев рішень є важливим інструментом в роботі кожного фахівця, що займається аналізом даних, незалежно від того практик він або теоретик.

На сьогоднішній день існує значне число алгоритмів, що реалізовують дерева рішень CART, C4.5, NewId, ITrule, CHAID, CN2 і так далі Але найбільше поширення і популярність отримали наступні два:

- CART (Classification and Regression Tree) – це алгоритм побудови бінарного дерева рішень – дихотомічної класифікаційної моделі. Кожен вузол дерева при розбитті має лише двох нащадків. Як видно з назви алгоритму, він вирішує задачі класифікації і регресії.
- C4.5 – алгоритм побудови дерева рішень, у якого кількість нащадків у вузла не обмежена. Алгоритм вміє працювати з безперервним цільовим полем, тому вирішує лише задачі класифікації.

Алгоритм CART. CART, скорочення від Classification And Regression Tree, перекладається як «Дерево класифікації і регресії» – алгоритм бінарного дерева рішень. Він був розроблений в 1974-1984 роках чотирма професорами статистики - Leo Breiman (Berkeley), Jerry Friedman (Stanford), Charles Stone (Berkeley) і Richard Olshen (Stanford). Алгоритм призначений для вирішення задач класифікації і регресії. Існує також декілька модифікованих версій – алгоритми IndCART і DB-CART. Алгоритм IndCART, є частиною пакету Ind і відрізняється від CART використанням іншого способу обробки пропущених значень, не здійснює регресійну частину алгоритму CART і має інші параметри відсікання. Алгоритм DB-CART базується на наступній ідеї: замість того аби використовувати навчальний набір даних для визначення розбиття, використовуємо його для оцінки розподілу вхідних і вихідних значень і потім використовуємо цю оцінку, аби визначити розбиття. DB, відповідно означає – «Distribution based». Стверджується, що ця ідея дає значне зменшення помилки класифікації, в порівнянні із стандартними методами побудови дерева. Основними відмінностями алгоритму CART від алгоритмів сімейства ID3 є:

- бінарне представлення дерева рішень;
- функція оцінки якості розбиття;
- механізм відсікання дерева;
- алгоритм обробки пропущених значень;

- побудова дерев регресії.

Бінарне представлення дерева рішень. У алгоритмі CART кожен вузол дерева рішень має двох нащадків. На кожному кроці побудови дерева правило, яке формується у вузлі, ділить задану множину прикладів (навчальну вибірку) на дві частини – частину, в якій виконується правило (нащадок – right) і частину, в якій правило не виконується (нащадок – left). Для вибору оптимального правила використовується функція оцінки якості розбиття.

Функція оцінки якості розбиття. Навчання дерева рішень відноситься до класу навчання з вчителем, тобто навчальна і тестова вибірки містять класифікований набір прикладів. Оціночна функція, використовувана алгоритмом CART, базується на інтуїтивній ідеї зменшення нечистоти (невизначеності) у вузлі. Розглянемо задачу з двома класами і вузлом, що має по 50 прикладів одного класу. Вузол має максимальну «нечистоту». Якщо буде знайдено розбиття, яке розбиває дані на дві підгрупи 40:5 прикладів в одній і 10:45 в іншій, то інтуїтивно «нечистота» зменшиться. Вона повністю зникне, коли буде знайдено розбиття, яке створить підгрупи 50:0 і 0:50. У алгоритмі CART ідея «нечистоти» формалізована в індексі Gini. Якщо набір T розбивається на дві частини T_1 і T_2 з числом прикладів в кожному N_1 і N_2 відповідно, тоді показник якості розбиття буде рівний

$$Gini_{split}(T) = \frac{N_1}{N} Gini(T_1) + \frac{N_2}{N} Gini(T_2).$$

Найкращим вважається те розбиття, для якого $Gini_{split}(T)$ мінімально.

Позначимо N – число прикладів у вузлі – предку, L, R – число прикладів відповідно в лівому і правому нащадку, l_i і r_i – число екземплярів i класу в лівому/правому нащадку. Тоді якість розбиття оцінюється по наступній формулі

$$Gini_{split} = \frac{L}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{l_i}{L}\right)^2\right) + \frac{R}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{r_i}{R}\right)^2\right) \to \min.$$

Аби зменшити об'єм обчислень формулу можна перетворити до вигляду

$$Gini_{split}(T) = \frac{1}{L}\sum_{i=1}^{n} l_i^2 + \frac{1}{R}\sum_{i=1}^{n} r_i^2 \to \max .$$

В результаті, кращим буде те розбиття, для якого величина максимальна. Рідше в алгоритмі CART використовуються інші критерії розбиття Twoing, Symmetric Gini і ін.

Правила розбиття. Вектор предикторних змінних, що подається на вхід дерева може містити як числові (порядкові) так і категоріальні змінні. В будь-якому разі в кожному вузлі розбиття йде лише по одній змінній. Якщо змінна числового типа, то у вузлі формується правило вигляду $x_i \le c$. Де c – деякий поріг, який найчастіше вибирається як середнє арифметичне двох сусідніх впорядкованих значень змінної x_i навчальної вибірки. Якщо змінна категоріального типа, то у вузлі формується правило $x_i * V(x_i)$, де $V(x_i)$ – деяка непорожня підмножина множини значень змінної x_i в навчальній вибірці. Отже, для n значень числового атрибуту алгоритм порівнює $n-1$ розбиття, а для категоріального $(2^{n-1}-1)$. На кожному кроці побудови дерева алгоритм послідовно порівнює все можливе розбиття для всіх атрибутів і вибирає найкращий атрибут і найкраще розбиття для нього.

Механізм відсікання дерева. Механізм відсікання дерева, оригінальна назва minimal cost-complexity tree pruning, – найбільш серйозна відмінність алгоритму CART від інших алгоритмів побудови дерева. CART розглядає відсікання як здобуття компромісу між двома проблемами: здобуття дерева оптимального розміру і здобуття точної оцінки вірогідності помилкової класифікації.

Основна проблема відсікання – велика кількість всіх можливих відсічених піддерев для одного дерева. Точніше, якщо бінарне дерево має $|T|$ – листів, тоді існує ~$[1.5028369^{|T|}]$ відсічених піддерев. І якщо дерево має хоч би 1000 листів, тоді число відсічених піддерев стає просто величезним. Базова ідея методу – не розглядати всі можливі піддерева, обмежившись лише «кращими представниками» згідно приведеній нижче оцінці.

Позначимо $|T|$ – число листів дерева, $R(T)$ – помилка класифікації дерева, рівна відношенню числа неправильно класифікованих прикладів до прикладів в навчальній вибірці. Визначимо $C_\alpha(T)$ – повну вартість (оцінку/показник витрати-складність) дерева T як

$$C_\alpha(T) = R(T) + \alpha|T|,$$

де $|T|$ – число листів (термінальних вузлів) дерева, α – деякий параметр, що змінюється від 0 до + ∞. Повна вартість дерева складається з двох компонент – помилки класифікації дерева і штрафу за його складність. Якщо помилка класифікації дерева незмінна, тоді із збільшенням повна вартість дерева збільшуватиметься. Тоді залежно від неї гіллясте дерево, що дає більшу помилку класифікації може коштувати менше, ніж те, що дає меншу помилку, але більш гіллясте.

Визначимо $T_{\max}$ – максимальне за розміром дерево, яке належить обрізувати. Якщо ми зафіксуємо значення α, тоді існує найменше мінімізуєме піддерево α, яке виконує наступні умови

$$C_\alpha(T(\alpha)) = \min_{T \le T_{\max}} C\alpha(T),$$

$$if C_\alpha(T) = C_\alpha(T(\alpha)) then T(\alpha) \le T.$$

Перша умова говорить, що не існує такого піддерева дерева $T_{\max}$, яке мало б меншу вартість, чим $T(\alpha)$ при цьому значенні α . Друга умова говорить, що якщо існує більш за одне піддерево, яке має таку ж повну вартість, тоді ми вибираємо найменше дерево.

Вибір фінального дерева. Отже, якщо ми маємо послідовність дерев, то нам необхідно вибрати краще дерево з неї. Найбільш очевидним є вибір фінального дерева через тестування на тестовій вибірці. Дерево, що дало мінімальну помилку класифікації, і буде кращим. Проте, це не єдино можлива дорога.

Перехресна перевірка. Перехресна перевірка (V-fold cross-validation) – найоригінальніша і складніша частина алгоритму CART. Цей шлях вибору фінального дерева використовується, коли набір даних для навчання малий або

кожен запис в ньому по своєму унікальний так, що ми не можемо виділити вибірку для навчання і вибірку для тестування. В такому разі будуємо дерево на всіх даних, обчислюваний $\alpha_1, \alpha_2, ..., \alpha_k$ і $T_1 > T_2 > ... > T_n$. Позначимо T_k – найменше піддерево, що мінімізується, для (α_k, α_{k+1}). Тепер ми хочемо вибрати дерево з послідовності, але вже використали все наявні дані. Хитрість в тому, що ми збираємося обчислити помилку дерева T_k з послідовності непрямим шляхом.

Крок 1. Встановимо $\beta_1 = 0, \beta_2 = \sqrt{\alpha_2 \alpha_3}, \beta_3 = \sqrt{\alpha_3 \alpha_4}, ..., \beta_{N-1} = \sqrt{\alpha_{N-1} \alpha_N}, \beta_N = \infty$. Вважається, що β_k буде типовим значенням для (α_k, α_{k+1}) і, отже, як значення відповідає T_k.

Крок 2. Розділимо весь набір даних на V груп однакового розміру $G_1, G_2, ..., G_V$. Рекомендується брати $V = 10$. Потім для кожної групи G_i:

1. Обчислити послідовність дерев за допомогою описаного вище механізму відсікання на всіх даних, виключаючи G_i. І визначити $T^{(i)}(\beta_1), ..., T^{(i)}(\beta_N)$ для цієї послідовності.

2. Обчислити помилку дерева $T^{(i0)}(\beta_k)$ на G_i. Тут $T^{(i0)}(\beta_k)$ означає найменше мінімізоване піддерево з послідовності, побудоване на всіх даних, виключаючи G_i.

Крок 3. Для кожного β_k підсумовувати помилку $T^{(i0)}(\beta_k)$ по всіх G_i. Хай β_h буде з найменшою загальною помилкою. Оскільки β_h відповідає дереву T_h, ми вибираємо T_h з послідовності побудованої на всіх даних як фінальне дерево. Показник помилки, обчислений за допомогою перехресної перевірки можна використовувати як оцінку помилки дерева.

Алгоритм обробки пропущених значень. Більшість алгоритмів інтелектуального аналізу даних передбачають відсутність пропущених значень. У практичному аналізі це припущення часто є невірним. Найбільш загальне рішення – відкинути дані, які містять один або декілька порожніх атрибутів. Проте це рішення має свої недоліки:

- Зсув даних. Якщо викинуті дані лежать декілька в стороні від залишених, тоді аналіз може дати упереджені результати.
- Зменшення потужності. Може виникнути ситуація, коли доведеться викинути багато даних. В такому разі точність прогнозу сильно зменшується.

Якщо ми хочемо будувати і використовувати дерево на неповних даних нам необхідно вирішити наступні питання - як визначити якість розбиття та у яку гілку необхідно послати спостереження, якщо пропущена змінна є такою, на яку доводиться найкраще розбиття.

Аби визначити якість розбиття CART просто ігнорує пропущені значення. Це вирішує першу проблему, але ми ще повинні вирішити, по якій дорозі посилати спостереження з пропущеною змінною що містить найкраще розбиття. З цією метою CART обчислює так зване «сурогатне» розбиття. Воно створює найбільш близькі до кращого підмножини прикладів в поточному вузлі.

Регресія. Побудова дерева регресії багато в чому схожа з деревом класифікації. Спочатку будується дерево максимального розміру, потім обрізується дерево до оптимального розміру.

Основне достоїнство дерев в порівнянні з іншими методами регресії – можливість працювати з багатовимірними задачами і задачами, в яких присутня залежність вихідний змінної від змінної або змінних категоріального типа. Основна ідея – розбиття всього простору на прямокутники, необов'язкового однакового розміру, в яких вихідна змінна вважається постійною (рис. 4.37).

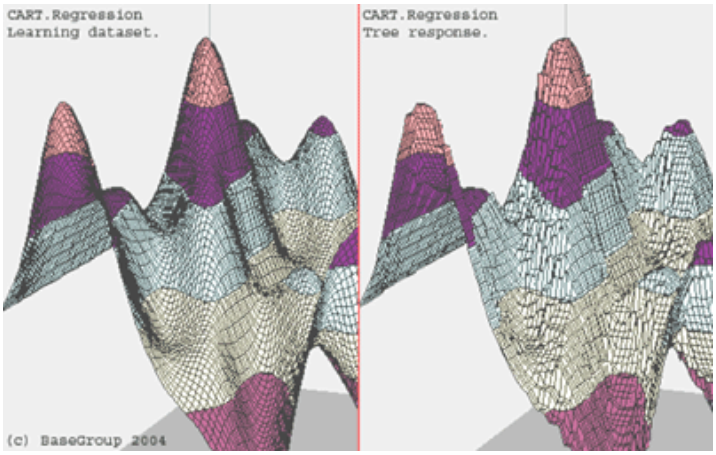


Рис. 4.37. Приклад двовимірної задачі.

Процес побудови дерева відбувається послідовно. На першому кроці отримуємо регресійну оцінку просто як константу по всьому простору прикладів. На другому кроці ділимо простір на дві частини. Якщо ми позначимо всі значення вихідний змінної як $Y_1, Y_2, ..., Y_n$ тоді регресійна оцінка має вигляд:

$$\bar{f}(x) = \left(\frac{1}{n} \sum_{i=1}^{n} Y_i \right) I_R(x),$$

де R – простір навчальних прикладів, n – число прикладів, $I_R(x)$ – індикаторна функція простору – фактично, набір правил, що описують попадання змінної x в простір.

Алгоритм C4.5. Алгоритм C4.5 будує дерево рішень з необмеженою кількістю гілок у вузла. Даний алгоритм може працювати лише з дискретним залежним атрибутом і тому може вирішувати лише задачі класифікації. C4.5 вважається одним з найвідоміших і широко використовуваних алгоритмів побудови дерев класифікації, який вперше був запропонований Р. Куїнленом (R. Quinlan). Перш ніж приступити до опису алгоритму побудови дерева рішень, визначимо обов'язкові вимоги до структури даних і безпосередньо до самих даних, при виконанні яких алгоритм C4.5 буде працездатний:

- Опис атрибутів. Дані, необхідні для роботи алгоритму, мають бути представлені у вигляді плоскої таблиці. Вся інформація про об'єкти з предметної області повинна описуватися у вигляді кінцевого набору ознак (атрибутів). Кожен атрибут повинен мати дискретне або числове значення. Самі атрибути не повинні мінятися від прикладу до прикладу, і кількість атрибутів має бути фіксованою для всіх прикладів.
- Певні класи. Кожен приклад має бути асоційований з конкретним класом, тобто один з атрибутів має бути вибраний як мітка класу.
- Дискретні класи. Класи мають бути дискретними, тобто мати кінцеве число значень. Кожен приклад повинен однозначно відноситися до конкретного класу. Випадки, коли приклади належать до класу з імовірнісними оцінками, виключаються. Кількість класів має бути значно менше кількості прикладів.

Розглянемо алгоритм побудови дерева. Хай нам задана множина прикладів T, де кожен елемент цієї множини описується m атрибутами. Кількість прикладів в множині T називатимемо потужністю цієї множини і позначатимемо $|T|$. Хай мітка класу набуває наступних значень $C_1, C_2, ..., C_k$. Задача полягатиме в побудові ієрархічної класифікаційної моделі у вигляді дерева з множиною прикладів T. Процес побудови дерева відбуватиметься зверху вниз. Спочатку створюється корінь дерева, потім нащадки кореня і так далі

На першому кроці ми маємо порожнє дерево (є лише корінь) і вихідну множину T (асоційовану з коренем). Потрібно розбити вихідну множину на підмножини. Це можна зробити, вибравши один з атрибутів як перевірку. Тоді в результаті розбиття виходять n (по числу значень атрибуту) підмножин і, відповідно, створюються n нащадків кореня, кожному з яких поставлена у відповідність своя підмножина, отримана при розбитті множини T. Потім ця процедура рекурсивно застосовується до всіх підмножин (нащадкам кореня) і так далі.

Отже, ми маємо дерево рішень і хочемо використовувати його для розпізнавання нового об'єкту. Обхід дерева рішень починається з кореня дерева. На кожному внутрішньому вузлі перевіряється значення об'єкту Y по атрибуту, який відповідає перевірці в даному вузлі, і, залежно від отриманої відповіді, знаходиться відповідне галуження, і по цій дузі рухаємося до вузла, що знаходить на рівень нижче і так далі Обхід дерева закінчується як тільки зустрінеться вузол рішення, який і дає назву класу об'єкту Y.

Остання версія алгоритму - алгоритм C4.8 - реалізована в інструменті Weka як J4.8 (Java), комерційна реалізація методу - C5.0, розробник RuleQuest, Австралія. Слід зазначити, що алгоритм C4.5 повільно працює на надвеликих і зашумлених наборах даних.

Найбільш серйозна вимога, яка зараз пред'являється до алгоритмів конструювання дерев рішень, - це масштабність, тобто алгоритм повинен володіти масштабованим методом доступу до даних. Розроблений ряд нових

масштабованих алгоритмів, серед них - алгоритм Sprint, запропонований Джоном Шафером і його колегами. Sprint, що є масштабованим варіантом розглянутого алгоритму CART, пред'являє мінімальні вимоги до об'єму оперативної пам'яті.

4.5. Комп’ютерні системи та напрямки застосування дерев рішень.

Дерева рішень корисні для бізнес-користувачів, оскільки надають логічний результат, який можна обговорювати в бізнес-термінах. Вони просто створюються із застосуванням різних програмних рішень, і дуже ефективні і точні при забезпеченні хорошим набором даних. Практично всі відомі виробники комп'ютерних систем включають до складу своїх програмних продуктів алгоритми побудови і аналізу дерев рішень. Розглянемо деякі з них.

Скорингові системи. Підвищення прибутковості кредитного портфеля банку безпосередньо залежить від грамотного управління кредитними ризиками. І саме скорингові системи дозволяють понизити ризики без втрати прибутковості, запропонувавши відповідь на ключові питання: наскільки проблематичною буде робота банку з конкретним позичальником, яке значення кредитного ліміту встановити, і поверне клієнт кредит чи ні.

У 1941 р. Девід Дюран вперше застосував методику класифікації кредитів на «хороших» і «поганих». Він визначив не лише групи чинників, що дозволяють максимально визначити міру кредитного ризику, але і коефіцієнти, що характеризують кредитоспроможність приватного клієнта. Таким чином, позичальник, який здолав порогове значення, набравши достатню кількість балів, потенційно міг отримати запрошувану суму. Ідея Дюрана отримала продовження - незабаром в Сан-Франциско утворилася перша консалтингова фірма в області скоринга Fair Issac, а декілька пізніше, з появою нових масових кредитних продуктів (кредитних карт), до ідеї скоринга звернулися всі фінансові установи США. По суть, скоринг є методом класифікації сукупності

позичальників на різні групи, коли необхідна характеристика не відома, проте, відомі інші характеристики, які яким-небудь чином корелюють з тією, що цікавить. На практиці, залежно від задач аналізу позичальника, кредитний скоринг включає application-скоринг - оцінку кредитоспроможності претендентів на здобуття кредиту (скоринг за анкетними даними використовується в першу чергу), behavioral-скоринг — оцінка вірогідності повернення виданих кредитів (поведінковий аналіз), а також collection-скоринг - оцінка можливості повного або часткового повернення кредиту при порушенні термінів погашення заборгованості (розрахунок ризиків по портфелю). На сьогодні відомі розробки SAS, KXEN, Experian, SPSS, EGAR, які є не спеціалізованими програмними засобами для скоринга, а універсальними аналітичними інструментами інтелектуального аналізу даних, які можна використовувати для побудови власних скорингових моделей. Тому, в повнішому розумінні, скорингова система зсередини є складною системою автоматизації видачі споживчих кредитів в банківських відділеннях, торгівельних точках, через інтернет, яка як аналітичне ядро використовує рішення однієї з відомих компаній-розробників. Сам по собі скоринг — це не лише робота з певними скоринговими моделями, але і побудова скорингової інфраструктури. Так, в багатьох програмних продуктах результат аналізу статистичних даних (мат. модель) можна зберегти у вигляді програмного коду, а його вставити в банківське програмне забезпечення. Тобто, під скоринговою системою мають на увазі спеціальне програмне забезпечення, за допомогою якого можна розрахувати необхідний показник на основі вихідних даних. Скорингова карта - це набір затверджених банком певних характеристик і відповідних вагових коефіцієнтів (у балах). Скорингових карт в банках зазвичай декілька, оскільки вони сильно залежать від кредитних продуктів. Наприклад, під нерухомість необхідна одна карта, а на покупку автомобіля вже абсолютно інша. На думку експертів, можна використовувати і одну загальну карту, проте це незручно для користувачів. Моделей також майже завжди декілька. Зазвичай заявка на кредит проходить через велику кількість моделей,

причому для різних категорій осіб можуть застосовуватися різні моделі навіть на одній скоринговій карті.

Однією з основних труднощів відомих скорингових систем, як і всіх технологічних рішень у сфері core banking, є їх погана адаптуємість. Річ у тому, що з часом можуть мінятися умови, в яких функціонує позичальник. А значить, скорингові моделі необхідно актуалізувати на найбільш «свіжих» клієнтах, періодично перевіряючи і, при необхідності, розробляючи нову модель, як для різних періодів часу, так і для різних регіонів. Подолання таких обмежень скоринга вирішується за допомогою інструментів інтелектуального аналізу даних. Найбільш поширеним методом автоматичного аналізу даних є побудова дерева рішень. Постачальники рішень упевнені: для того, щоб отримати обґрунтовані висновки, не обов'язково бути статистиком. Наприклад, AnswerTree (продукт SPSS) автоматично будує дерево, дозволяючи на базі діалогових вікон навіть непідготовленому користувачеві почати роботу з програмою. Сама AnswerTree автоматично просіює дані і знаходить статистично значимі групи. За допомогою інтуїтивно зрозумілих деревовидних діаграм, графіків і таблиць програма самостійно сегментує дані, при цьому аналітикові необхідно лише вказати цільову змінну, змінні-предикати і вибрати алгоритм побудови дерева рішень. Зручно, що деревовидна діаграма, яка схожа на блок-схему, дозволяє візуалізувати виділені сегменти і закономірності в даних. Для здобуття максимально достовірних результатів зазвичай рекомендується навчити модель на підвибірці, а потім протестувати надійність на даних, що залишилися. Наскільки добре модель описує дані, можна побачити, перемикаючись з навчальної моделі на контрольну. Представити результати аналізу можна в будь-якому форматі, наприклад, вивести інформацію по кожному вузлу у вигляді таблиці або графіка. Правильно побудоване на даних минулих періодів дерево рішення володіє ще однією дуже важливою особливістю, а саме, «здібністю до узагальнення». Це означає, що якщо виникає нова ситуація, можна з досить великою часткою упевненості

сказати, що позичальник, який знов звернувся, поведеться так само, як і ті позичальники, характеристики яких схожі з характеристиками нових клієнтів.

Скорингова система традиційно складається з модуля підготовки вихідних даних, аналітичного модуля і модуля звітності. Дані системи скоринга, можуть бути трьох типів. Перший тип - знання персоналу кредитних відділів банків про конкретні типи кредитних продуктів і своїх клієнтів. Другий тип даних - статистика по вже виданих кредитах, що враховує «добрих» і «поганих» позичальників. І, якщо банк не володіє жодним з типів вказаних даних - ні експертними знаннями, ні статистикою виданих кредитів, модель, що лежить в основі системи скоринга, переважно будується на основі регіональних і галузевих даних.

Всі фронт-офісні рішення для автоматизації процесу споживчого кредитування в більшості випадків є Web-додатками, що забезпечують хорошу масштабність системи і простоту підключення до процесу видачі кредитів нових відділень банку і представництв в торгівельних точках. Найбільш відомими західними скоринговими системами сьогодні є SAS Credit Scoring, EGAR Scoring, Transact SM (Experian-Scorex), K4Loans (KXEN), Clementine (SPSS). Серед розробників з СНД — BNS, Basegroup Labs. Найбільш серйозними і дорогими є рішення SAS (близько 200 тис. дол.), гідними також вважаються розробки KXEN (близько 30 тис. дол.). Практично посередині цінового діапазону знаходиться пропозиція по скорингу компанії EGAR Technology, яка, з одного боку, є західним вендором, що пропонує скорингову систему, яка використовує класичні західні моделі, з іншого боку, - це рішення (EGAR Scoring) максимально адаптоване до українських умов і доповнене спеціальними підходами - наприклад, макроекономічним підходом до оцінки кредитоспроможності позичальника, обліком особливостей самих кредитних продуктів і іншими можливостями. Порівнюючи західні і вітчизняні системи, необхідно відмітити наступне:

1. Західні системи з'явилися набагато раніше, у них великий термін експлуатації, відповідно великий об'єм кредитних історій, але ці історії не підходять для українського ринку.

2. У західних системах немає інструментів (можливостей) для роботи з малими об'ємами кредитних історій (що необхідне для українського ринку). Розробки в області класичного скоринга дозволяють працювати з обмеженими об'ємами кредитних історій.

3. Ще одна особливість - велика різниця між скоринговими картами залежно від локальних ринків і для різних банківських продуктів. Відповідно, західні системи недостатньо гнучкі для вітчизняного ринку.

Застосування технології дерев рішень для оцінки кредитоспроможності фізичних осіб на основі пакету Deductor. При кредитуванні фізичних осіб характерні невеликі розміри позик, що породжує великий об'єм роботи по їх оформленню і досить дорогу процедуру оцінки кредитоспроможності відносно отримуваного в результаті прибутку. Для оцінки кредитоспроможності фізичних осіб банку необхідно оцінити як фінансове положення позичальника, так і його особисті якості. При цьому кредитний ризик складається з ризику неповернення основної суми боргу і відсотків по цій сумі. Зараз для оцінки ризику кредитування позичальника використовується скоринг кредитування. Сутність цієї методики полягає в тому, що кожен чинник, що характеризує позичальника, має свою кількісну оцінку. Підсумовуючи отримані бали, можна отримати оцінку кредитоспроможності фізичної особи. Кожен параметр має максимально можливий поріг, який вище для важливих питань і нижче для другорядних. На сьогоднішній день відомо досить багато методик кредитного скоринга. Однією з найвідоміших є модель Дюрана. Основним недоліком скорингової системи оцінки кредитоспроможності фізичних осіб є те, що вона дуже погано адаптуєма, а також має високу вартість адаптації використовуваної моделі під поточне положення справ і велику ймовірність помилки моделі при визначенні

кредитоспроможності потенційного позичальника, обумовленої суб'єктивною думкою фахівця.

Одним з варіантів розв’язання такої задачі є застосування алгоритмів, які вирішують задачі класифікації. Такого роду задачі з великим успіхом вирішуються за допомогою дерев рішень. Для демонстрації подібної технології скористаємося програмою Tree Analyzer з пакету Deductor. За вихідні дані була взята вибірка, що складається з 1000 записів, де кожен запис – це опис характеристик позичальника і параметр, що описує його поведінку під час погашення позики. При навчанні дерева використовувалися наступні чинники, що визначають позичальника: «N Паспорта», «ФІО», «Адреса», «Розмір позики», «Термін позики», «Мета позики» «Середньомісячний дохід», «Середньомісячна витрата» «Основний напрям витрат», «Наявність нерухомості» «Наявність автотранспорту», «Наявність банківського рахунку», «Наявність страховки», «Назва організації», «Галузева приналежність підприємства», «Термін роботи на даному підприємстві», «Напрям діяльності позичальника», «Термін роботи на даному напрямі», «Стать», «Забезпеченість позики», «Давати кредит» та інші. При цьому поля: «N Паспорта», «ФІО», «Адреса», «Назва організації» визначені алгоритмом вже до початку побудови дерева рішень як непридатні внаслідок практичної унікальності кожного із значень (рис. 4.38).

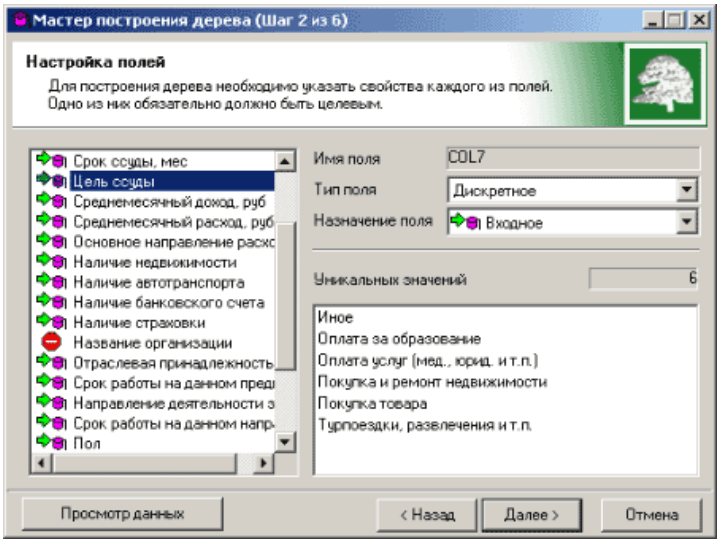


Рис. 4.38. Настройка визначальних і цільових чинників.

Цільовим полем є поле «Давати кредит», що набуває значень «Так» (True) і «Ні» (False). Ці значення можна інтерпретувати таким чином: «Ні» – платник або сильно прострочив з платежами, або не повернув частину грошей, «Так» – протилежність «Ні».

Після процесу побудови дерева рішень за допомогою програми Tree Analyzer отримуємо наступну модель оцінки кредитоспроможності фізичних осіб, що описує ситуацію, яка відноситься до певного банку. Ця модель представлена у вигляді ієрархічної структури правил – дерева рішень (рис. 4.39).

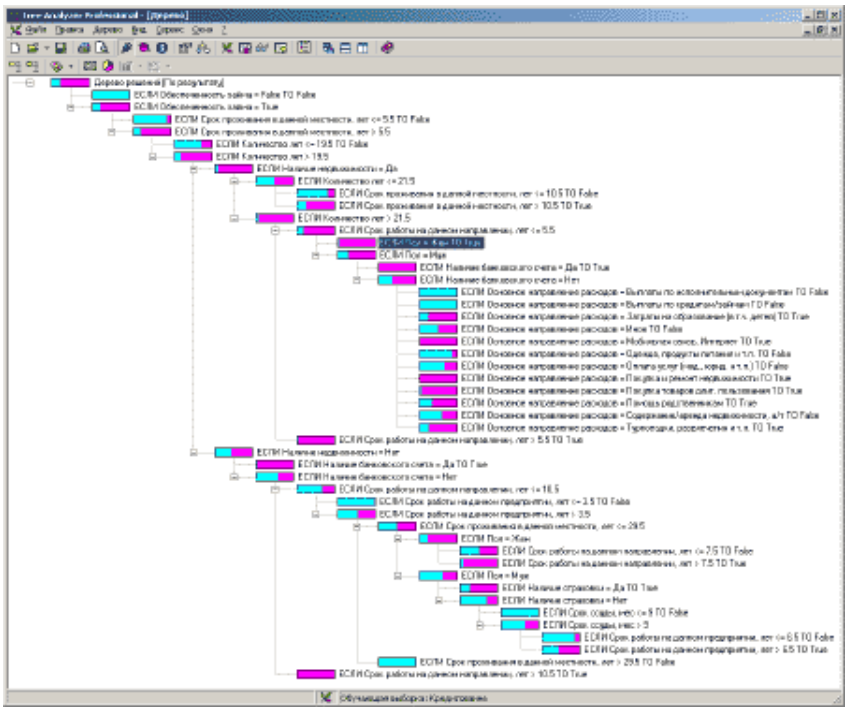


Рис. 4.39. Дерево рішень – модель визначення кредитоспроможності фізичних осіб.

Аналізуючи отримане дерево рішень, можна відмітити наступне:

1. За допомогою дерева рішень можна проводити аналіз значущих чинників. Таке можливе завдяки тому, що при визначенні параметра на кожному рівні ієрархії, по якому відбувається розділення на дочірні вузли, використовується критерій найбільшого усунення невизначеності. Таким чином, значиміші чинники, по яких проводиться класифікація, знаходяться на ближчій відстані

(глибині) від кореня дерева, чим менш значимі. Наприклад, чинник «Забезпеченість позики» більш значущий, ніж чинник «Термін мешкання в даній місцевості». А чинник «Основний напрям витрат» є значущим лише у поєднанні з іншими чинниками. Ще одним цікавим прикладом значущості різних чинників служить відсутність в побудованому дереві параметра «Наявність автотранспорту», що говорить про те, що на сьогоднішній день ця наявність не є визначальною при оцінці кредитоспроможності фізичної особи.

2. Можна відмітити, що такі показники як «Розмір позики», «Термін позики», «Середньомісячний дохід» і «Середньомісячна витрата» взагалі відсутні в отриманому дереві. Даний факт можна пояснити тим, що у вихідних даних присутній такий показник як «Забезпеченість позики», і оскільки цей чинник є точним узагальненням чотирьох вищеописаних показників, алгоритм побудови дерева рішень вибрав саме його.

Важливою особливістю побудованої моделі є те, що правила, по яких визначається приналежність позичальника до тієї або іншої групи, записані на природній мові. Наприклад, на основі побудованої моделі виходять наступні правила:

- ЯКЩО Забезпеченість позики = Так І Термін мешкання в даній місцевості, років > 5.5 І Кількість років > 19.5 І Наявність нерухомості = Так і Наявність банківського рахунку = Так ТО Давати кредит = Так (Достовірно на 98%)
- ЯКЩО Забезпеченість позики = Так І Термін мешкання в даній місцевості, років > 5.5 І Наявність нерухомості = Так і Кількість років > 21.5 І Термін роботи на даному напрямі, років <= 5.5 І Стать = Чол І Наявність банківського рахунку = Немає І Основний напрям витрат = Одяг, продукти харчування і тому подібне ТО Давати кредит = Ні (Достовірно на 88%).

На основі побудованої моделі також можна визначати приналежність потенційного позичальника до одного з класів. Для цього необхідно скористатися діалоговим вікном «Експеримент» програми Tree Analyzer, в

якому, послідовно відповідаючи на питання, можна отримати відповідь на питання: «Чи давати кредит» (рис. 4.40).

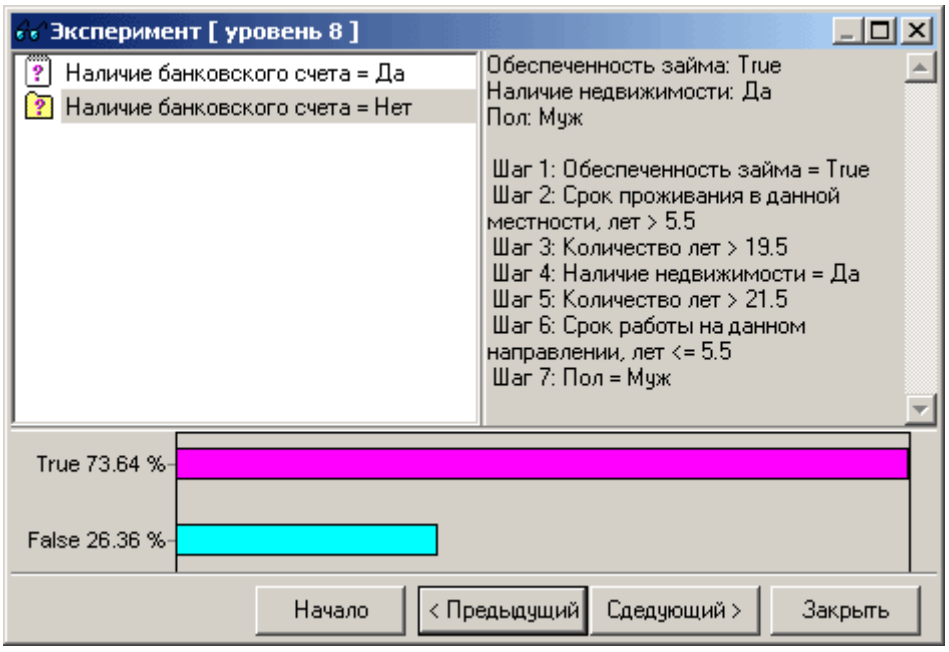


Рис. 4.40. Вікно «Експеримент».

Використовуючи такий підхід, можна усунути відразу обоє вищеописані недоліки скорингової системи оцінки кредитоспроможності.

Програмний комплекс Oracle Data Miner. Використання дерева рішень - це спосіб класифікації існуючих даних, визначення чинників або правил, які мають відношення до цільового результату (target result), і їх вживання для прогнозування результату, що означає:

1. Бізнес-користувачі можуть визначати чинники, які найбільшою мірою впливають на рішення про покупки.

2. Департаменти маркетингу можуть «цілитися» в «правильні» групи потенційних клієнтів, виключаючи тих, хто з малою вірогідністю купуватиме.

3. Аналітики даних і фінансові аналітики можуть прогнозувати продажу завдяки аналізу атрибутів потенційних клієнтів, про яких є дані.

4. Бізнес-аналітики можуть коректувати цілі і стратегії при змінах тенденцій

5. Компанії можуть реорганізовувати підтримку (support, enhancements, and desupport) для забезпечення максимального задоволення клієнтів.

Виробник пропонує два продукти, A і B. В цілому відгуки споживачів були позитивні, але власник підприємства-виробника хоче взнати, що можна змінити в підтримці для того щоб може підвищити рівень задоволення споживачів. У підприємства є вельми обмежена інформація про своїх споживачів, включаючи лише дані про продукт, який вони використовують, його версію і час здобуття його останньої модифікації. З використанням цієї інформації, отриманої від вибірки по клієнтській базі (sample customer population), і Oracle Data Miner це підприємство може створити модель дерева рішення (рис. 4.41).

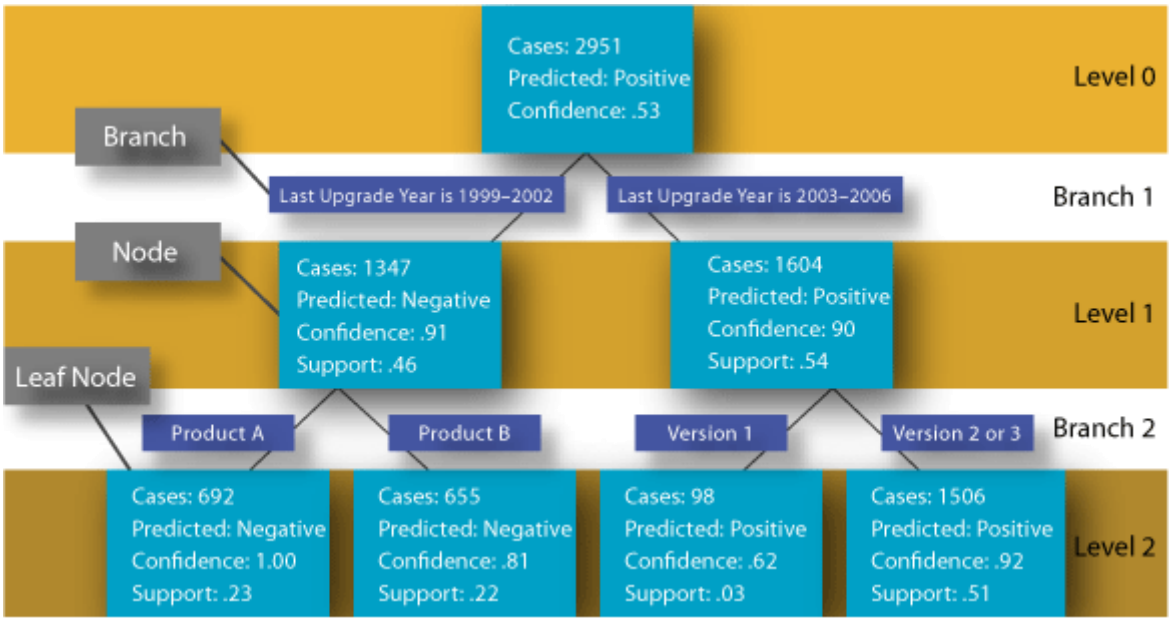


Рис. 4.41: Дерево рішень.

Кожен прямокутник в дереві називається вузлом (node) і кожна лінія називається віткою (branch) або ребром. Верхній прямокутник в дереві (або його корінь (root)) включає всі значення (all cases) цієї вибірки. Дерево рішень розділяє дані по атрибутах в спробі визначити кращих провісників (predictors) цільового значення (target value). Ці провісники формують правило (rule) або набір правил, які будучи застосовані до вузла, сформують результат. Ви можете думати про них, як про пропозиції IF-THEN для ухвалення рішень.

Oracle Data Miner аналізує всі атрибути в наборі даних. Якщо, наприклад, в даному наборі даних три атрибути, то Oracle Data Miner аналізує ці три атрибути. Якщо ж атрибутів 80, то Oracle Data Miner аналізує всі ці 80 атрибутів. Він визначає атрибут для першого розщеплювання (split) дерева рішення, яке щонайкраще ділить цільові дані (target data) на різні секції. З цим набором даних, розділеним надвоє, Oracle Data Miner може визначити атрибути для розщеплювань на наступному рівні. Зверніть увагу, що на рис. 4.41 Oracle Data Miner розщепнув гілки 2-го рівня по різних атрибутах. До останнього ряду вузлів посилаються як до термінального вузла або листа (terminal node, or leaf). З 4-х вузлів 2-го рівня один показує, що клієнти, які використовують версії 2 або 3 продукти A або B, до яких вони перейшли між 2003 і 2006, з 92% упевненості висловлюють позитивну думку про продукт. Інший же вузол показує, що клієнти, що використовують продукт A, до якого вони перейшли між 1999 і 2001, будуть, як передбачається, негативно відноситися до цього продукту. Безумовно, ця інформація буде корисної при планеруванні підтримки (рис. 4.42).

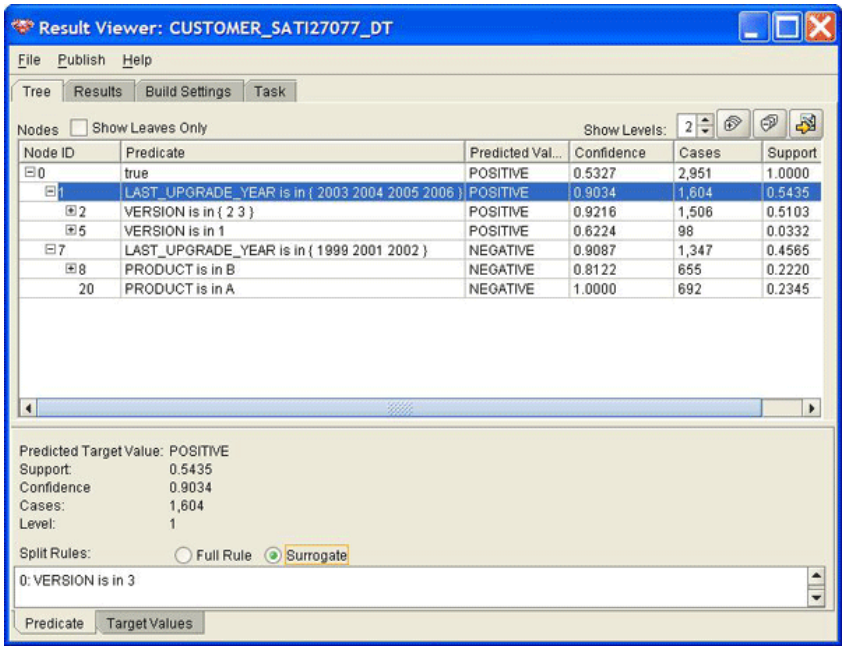


Рис. 4.42. Result Viewer показує правила розщеплювання.

З кожним листом асоціюється правило розщеплювання, яке з'являється внизу даного вікна, якщо клацнути по відповідному вузлу. Наприклад, останній лист (Node ID = 20 на рис. 4.42) передбачає негативну відповідь з 100% упевненістю. Oracle Data Miner дозволяє також побачити, які правила були використані для кожного запису. Натискуйте один із записів, а потім Rule button праворуч від результатів. Переглядач правив Rule Viewer з'явиться, як показано на рис. 4.43.

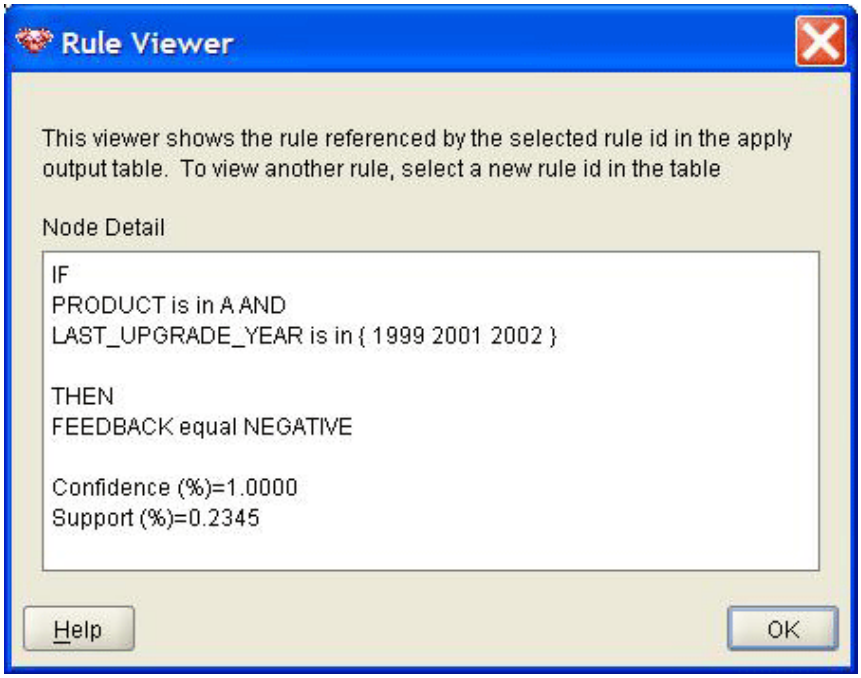


Рис 4.43. Переглядач правил Rule Viewer.

Система PolyAnalyst. У системі PolyAnalyst, реалізований алгоритм, заснований на критерії максимізації взаємної інформації (information gain). Тобто для розщеплювання вибирається незалежна змінна, що несе максимальну (у сенсі Шенона) інформацію про залежну змінну. Цей критерій на відміну від багатьох критеріїв, вживаних в інших системах Data Mining, має ясну інтерпретацію і дає розумні результати при найрізноманітніших статистичних параметрах даних, що вивчаються. Алгоритм DT є одним з найшвидших в PolyAnalyst (рис. 4.44).

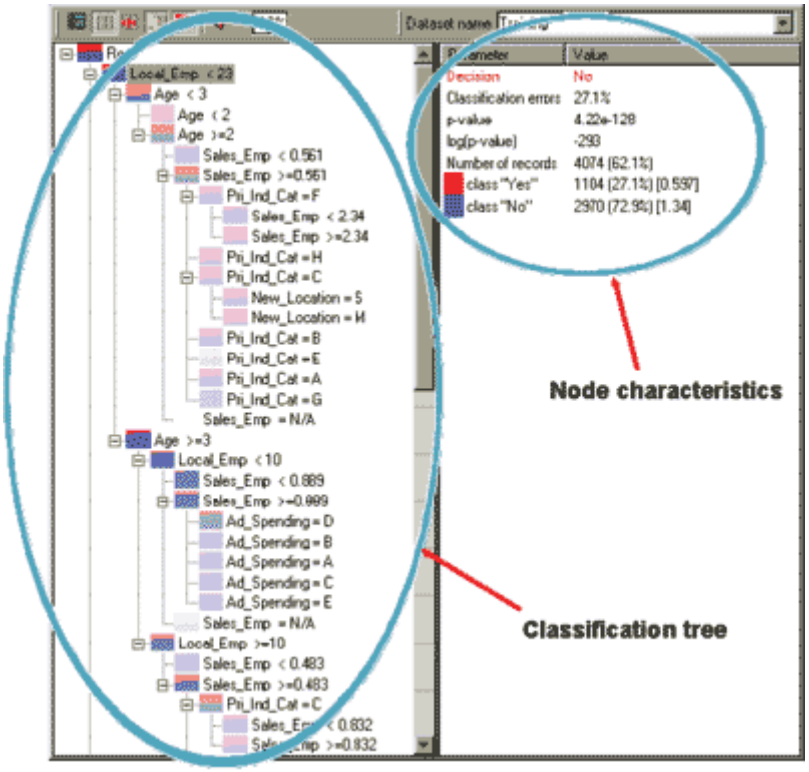
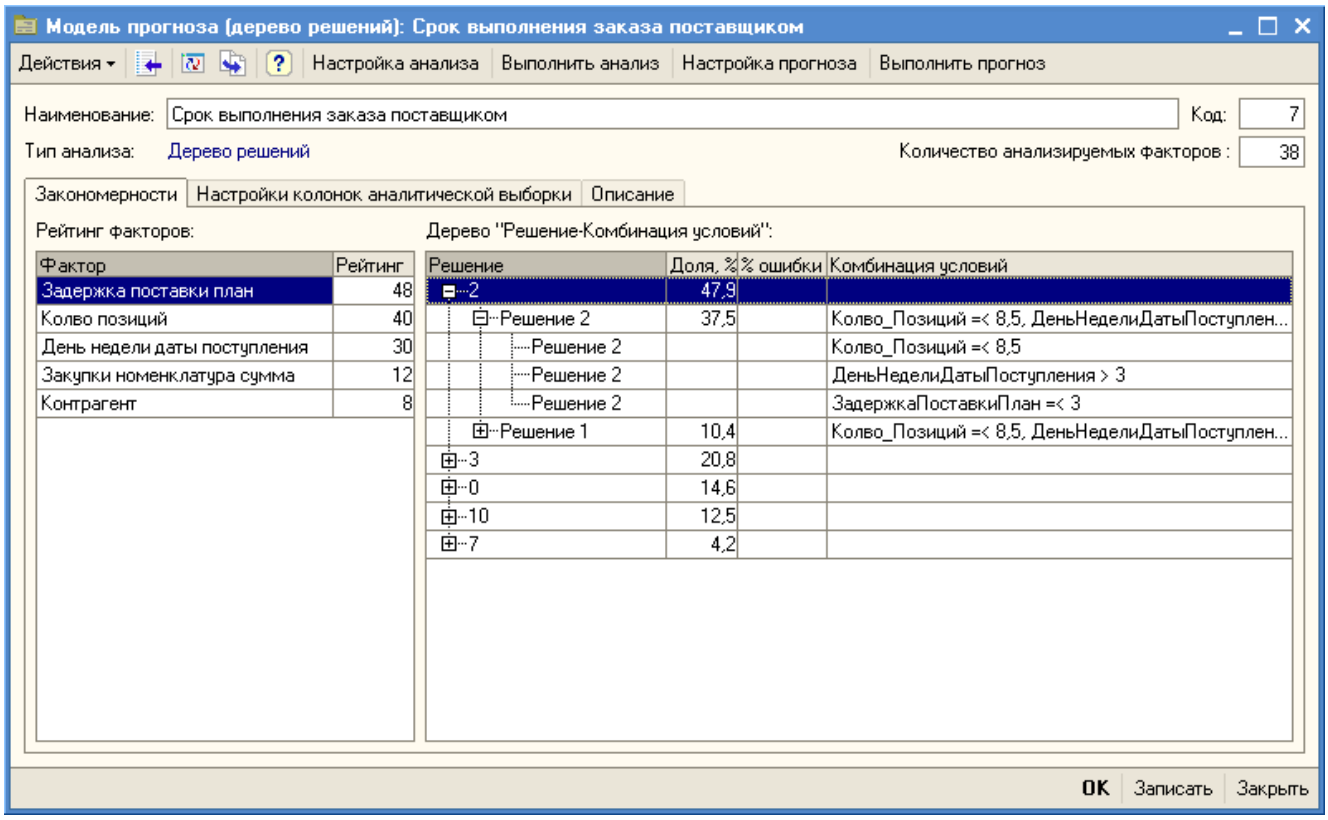


Рис. 4.44. Дерево рішень в системі PolyAnalyst.

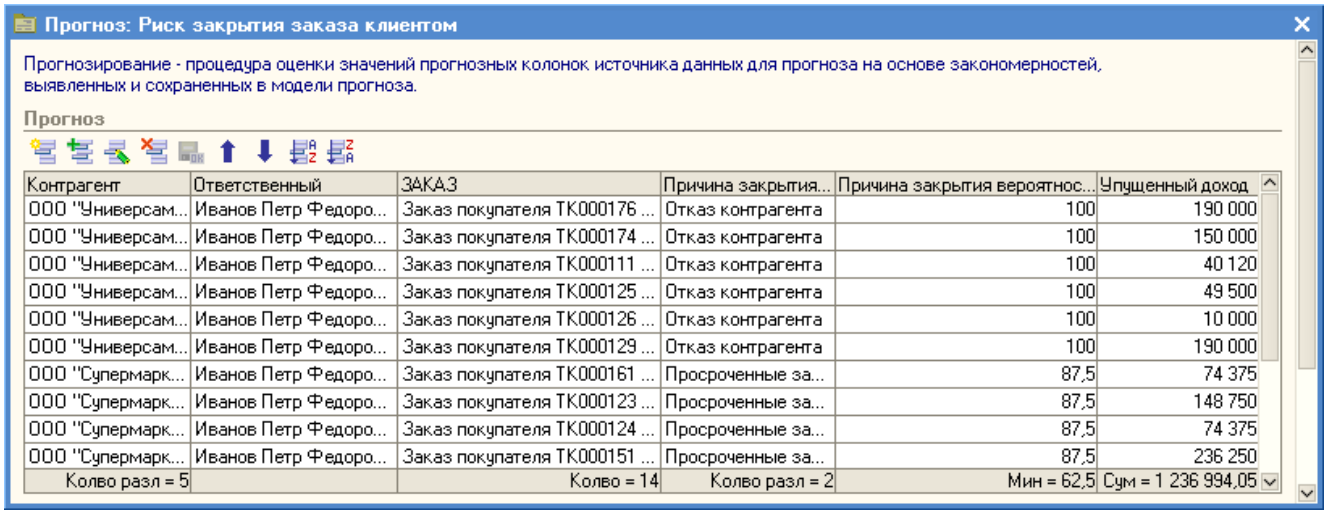
У разі, коли залежна змінна може приймати велику кількість різних значень, вживання методу дерев рішень стає неефективним. У такій ситуації в системі PolyAnalyst застосовується метод, званий лісом рішень (decision forest). При цьому будується сукупність дерев рішень - поодинці для кожного різного значення залежної змінної. Результатом прогнозу, заснованому на лісі рішень, є те значення залежної змінної, для якої відповідне дерево дає найбільш вірогідну оцінку.

Інформаційна система «1С: Підприємство 8.0». Даний алгоритм набув найбільшого поширення при виявленні причинно-наслідкових зв'язків в даних і описі поведінкових моделей. Типова зона застосовності дерев рішень – оцінка різних ризиків, наприклад, закриття замовлення клієнтом або його переходу до конкурента, невчасного постачання товару постачальником або прострочення оплати товарного кредиту. Як типові вхідні чинники моделі виступають сума і склад замовлення, поточне сальдо взаєморозрахунків, кредитний ліміт, відсоток передоплати, умови постачання і інші параметри, що характеризують об'єкт прогнозу. Адекватна оцінка ризику забезпечує ухвалення інформованих рішень

по оптимізації відношення доходность/ризик в діяльності компанії, а також корисна для збільшення реалістичності різних бюджетів (рис. 4.45).



а)



б)

Рис. 4.45. Вживання методу дерева рішень дозволяє на основі вхідних чинників моделі (а) отримувати оцінку ризиків ухвалення тих або інших управлінських рішень (б).

Як приклад, що ілюструє здатність цього алгоритму виявляти причинно-слідчі зв'язки, можна привести задачу оптимізації роботи відділу продажів. Для її вирішення як прогнозовану величину виберемо показник ефективності менеджерів по продажах, наприклад питому прибутковість на клієнта, а як чинники – сукупність даних, що потенційно впливають на результат. Алгоритм визначить чинники, що роблять найбільший вплив на результат, а також типові комбінації умов, що приводять до того або іншого результату. Більш того, підсистема «Аналіз даних» дозволить оцінити (спрогнозувати) очікувані значення цільового показника на підставі актуальних даних, а також провести прогноз «Що, якщо...», змінюючи показники, що подаються на вхід моделі. Результати аналізу і прогнозу за допомогою дерев рішень дозволяють істотно понизити вплив невизначеності бізнес-оточення на стан компанії, а також вирішити широкий спектр завдань, пов'язаних з виявленням складних і неочевидних причинно-наслідкових зв'язків.

Алгоритм «Дерево рішень» формує причинно-наслідкову ієрархію умов, що приводить до певних рішень. В результаті застосування цього методу до навчальної вибірки створюється ієрархічна (деревовидна) структура правил розщеплювання вигляду «Якщо... то...». Алгоритм аналізу (навчання моделі) зводиться до ітеративного процесу вичленення на кожному етапі найбільш значимих умов і переходів між ними. Умови можуть мати як кількісний, так і якісний характер і формують «гілки» цього абстрактного дерева. Його «листя» утворюють значення прогнозованого атрибуту (рішення), які також як і умови переходів можуть мати як якісне, таке і кількісне трактування. Сукупність цих умов, що накладаються на чинники і структура переходів між ними до кінцевого рішення і утворюють модель прогнозу.

Даний алгоритм набув найбільшого поширення при оцінці результатів різних подієвих ланцюжків і виявленні причинно-наслідкових зв'язків у вибірках. Управління значущістю і достовірністю моделі даного алгоритму здійснюється за допомогою параметрів «Тип спрощення», «Максимальна

глибина дерева» і «Мінімальна кількість елементів у вузлі». Як результат аналізу вибірки за допомогою алгоритму «Дерево рішень» виступають:

- рейтинг чинників, що є списком чинників, які зробили вплив на рішення, відсортований в порядку убування значущості («цитування» у вузлах дерева);
- зіставлення рішень (значень прогнозної колонки) і умов, що визначили їх, іншими словами - дерево «Наслідок-причина»;
- дерево «Причина-наслідок», що є сукупністю переходів між умовами, які визначають те або інше рішення (по суті - візуальне представлення моделі прогнозу).

В якості типового прикладу вживання інтелектуального аналізу даних стосовно конфігурації «1С: Управління торгівлею 8.0.» приведемо бізнес-сценарій – «Управління персоналом».

1. Сценарій – «Профілізація менеджерів відділу продажів за ключовими показниками ефективності». Визначення ефективності менеджерів (утримання, пошук клієнтів, ефективність комунікацій, інкасація умовної і безумовної дебіторської заборгованості, питомим показникам ефективності на клієнта і так далі) представляє інтерес не лише з точки зору формування системи матеріального стимулювання менеджерів, але і з точки зору ефективного нормування параметрів їх діяльності.

2. Алгоритм – «Дерева рішень».

3. Прогнозні атрибути - ключові показники ефективності відділу продажів (кількість ключових клієнтів, коефіцієнти відтоку і залучення, упущений дохід в місяць, притягнений дохід в місяць, дохід в місяць з клієнта, сумарні поступлення від клієнтів і так далі).

4. Основні чинники - кількість активних клієнтів, виручка, дохід, питомі показники на клієнта, ефективність комунікації. Залежно від прогнозних атрибутів склад чинників може істотно варіюватися.

5. Приклад закономірності. Менеджери, що забезпечують кращі показники інкасації дебіторської заборгованості мають коефіцієнт утримання > 0.8, коефіцієнт залучення > 0.25, кількість одночасно відкритих операцій не

більше 15, але не менше 10, інтенсивність подій в день не більше 10, але не менше 3, кількість активних клієнтів в періоді не менше 50, але не більше 100.

Автоматична класифікація тексту. Внутрішні вузли є термами, гілки, що відходять від них, характеризують вагу терма в аналізованому документі, а листя - категорії. Такий класифікатор категоризує випробовуваний документ, рекурсивно перевіряючи ваги вектора ознак по відношенню до порогів, виставлених для кожної з ваг, поки не досягне листа дерева (категорії). До цієї категорії (листа якого досяг класифікатор) і приписується аналізований документ.

На сьогоднішній день розроблена чимала кількість алгоритмів, що використовують для класифікації текстів технологією дерева рішень. Серед цих алгоритмів такі як CART, NewId, ITrule, CHAID і так далі Але найбільшою популярністю користується ID3 (Iterative Dichotomizer) і його розширені версії: загальнодоступна C4.5 і комерційна C5, які додають до алгоритму ID3 нові можливості. У основі багатьох сучасних алгоритмів лежить стратегія розділення і захвату (divide and conquer). Хай нам задана деяка навчальна множина, що містить отрубрицірованні документи, кожен з яких характеризується m атрибутами, причому один з них вказує на приналежність об'єкту до певного класу, та позначені класи (значення мітки класу). Тоді алгоритмом проводяться наступні перевірки:

- чи мають всі навчальні приклади одні і ті ж мітки класу (в цьому випадку дерево рішень для цього випадку є лист, що позначає клас);
- якщо немає, вибирають терм, який розділяє множину на класи документів, що мають ті ж самі значення вибраного параметра, і поміщає кожен такий клас в окреме піддерево;
- процес рекурсивно повторюється на піддеревах до тих пір, поки кожен лист дерева, не міститиме навчальні приклади, приписані до однієї з категорій.

Найважливіший крок - вибір значення параметра, який управляє вибором гілки. Вибраний параметр повинен розбити множину так, щоб отримувані у результаті підмножини складалися з документів, що належать до

одного класу, або мали мінімальне число «викидів» (тобто документів, що не відносяться до основного класу). Зазвичай, як такі параметри використовують приріст інформації (information gain) або критерій ентропії (entropy criterion). Часто результатом роботи алгоритмів побудови дерев рішень є складні дерева, що мають безліч вузлів і гілок. Такі дерева володіють правилами побудови класифікатора, що важко інтерпретуються. Окрім цього дерево, яке має величезну кількістю вузлів, розбиває навчальну множину на велике число підмножин, що складаються з малого числа документів.
Для вирішення цієї проблеми зазвичай використовується метод, що отримав назву обрізка гілок (pruning). При використанні цього підходу, обрізання гілок, які не приводять до зростання помилки класифікації, відбувається від низу до верху. Класифікатор рухається з листя дерева, позначаючи вузли як листя, або замінюючи їх піддеревом. Не дивлячись на недоліки, переваги дерев рішень дозволили їм стати одними з найпопулярніших методів автоматичної класифікації тексту.

Машинне навчання. Мета методів машинного навчання – здобуття простих класифікуючих виразів, які були б легко зрозумілі для людини. Достоїнством таких методів є те, що під час роботи того або іншого методу не потрібна участь людини. У дослідженні, проведеному в рамках європейського проекту StatLog, був проведений аналіз статистичних методів (аналіз дискримінанта, кластер-аналіз і так далі), дерев рішень(C4.5, AC2, CART, NEWID, CN2, ITrule) і нейронних мереж (багатошарові мережі, РБФ-мережі, карти Кохонена) для вирішення задач класифікації. Дані були взяті з різних предметних областей: розпізнавання образів (рукописного тексту, автомобілів), медична діагностика (діабет, травми голови, сердечні захворювання), молекулярної біології (розпізнавання структури ДНК), видача кредитів і так далі В ході дослідження з'ясувалося, що дерева рішень показали найкращі результати у вирішенні наступних задач:

- оцінка кредитоспроможності кандидата на здобуття кредиту;
- діагностика несправностей в технічних системах;

- розміщення радіаторів в Space Shuttle.

Достоїнствами дерев рішень є:

1. На навчання дерев рішень потрібний значно менше часу, чим, наприклад, на навчання нейронних мереж. Результат роботи представляється у вигляді, що легко інтерпретується для людини. Класифікаційна модель, представлена у вигляді дерева є інтуїтивно зрозумілою для людини, на відміну від нейронних мереж, що є за своєю природою чорним ящиком.

2. На вхід алгоритму дерев рішень можна подавати будь-яку кількість параметрів, алгоритм сам вибере найбільш значимі параметри і лише вони фігуруватимуть в побудованому дереві. Це позбавляє користувача від необхідності визначати вхідні параметри. Знову ж таки, при використанні нейронних мереж ми повинні дуже обережно підходити до питання про вхідні поля, так, із зростанням кількості вхідних полів, збільшується час що витрачається на процес навчання, який і так є дуже довгим і викликає багато нарікань.

3. Точність прогнозу дерев рішень співпідставна з іншими методами побудови класифікаційних моделей (статистичні методи, нейроні мережі).

4. Існують масштабовані алгоритми дерев рішень SLIQ, SPRINT, тобто із зростанням числа прикладів час, що витрачається на навчання, зростає лінійно для побудови дерев рішень на надвеликих базах даних. Алгоритми побудови дерев рішень мають методи спеціальної обробки пропущених даних.

5. Класичні і сучасні методи статистики використовувані в задачах класифікації працюють лише з числовими даними, дерева рішень успішно працюють як з числовими, так і строковими значеннями. Крім того, деякі із статистичних методів є параметричними, тобто необхідно заздалегідь знати вигляд моделі або залежність між залежними і незалежними змінними. Наприклад, класифікатори, побудовані за принципом максимальної правдоподібності, передбачають, що дані мають нормальний розподіл. Вони також дозволяють витягувати правила на природній мові.

Самонавчальні системи прийняття рішень. Сьогодні актуальною проблемою є створення систем, що самостійно навчаються для роботи на фінансових ринках. Здібна до самонавчання система автоматичного управління (самоналагоджувальна система), в якій пристосування до умов, які випадково змінюються, забезпечується автоматичною зміною параметрів налаштування або шляхом автоматичного пошуку оптимального налаштування. У будь-якій іншій автоматичній системі управління є параметри, які впливають на стійкість і якості процесів управління і можуть бути змінені при регуляції (налаштуванні) системи. Якщо ці параметри залишаються незмінними, а умови функціонування (характеристики керованого об'єкту, збуджуючі дії) істотно змінюються, то процес управління може погіршати або навіть стати нестійким. Ручне налаштування системи часто виявляється складним, а інколи і неможливим. Використання в таких випадках самонавчальної системи технічно і економічно доцільно і навіть може виявитися єдиним способом надійного управління. У тих випадках, коли самоналаштування застосовується в системах управління в наслідок невірогідності знання властивостей об'єкту, система самооптимізації нагадує процес самонавчання. Система при цьому шляхом автоматичного пошуку немов би сама пізнає невідомі властивості керованого об'єкту і вчиться управляти цим об'єктом щонайкраще. Вживання таких систем вигідне в тих випадках, коли зовнішня дія може бути виміряна з метою аналізу його властивостей і коли зміна його форми є вирішальною для якостей роботи системи.

Особливість системи полягає в тому, що вона з часом набудовуватиметься, навчатиметься і нагромаджуватиме досвід користувача. Актуальність самонавчальної системи саме у тому, що людина не втручається в налаштування і її навчання. У системі, яка була розроблена, експертна оцінка буде індивідуальною для кожного інвестора, його набору чинників і отриманих трендів.

Побудова моделі прийняття рішення зводиться до рішення задачі класифікації. Для вирішення такої задачі з невизначеними початковими даними

на практиці найбільшого поширення набув метод дерева рішень, який дозволяє визначити набір правил класифікації типа «Якщо - То». Ці правила і будуть порадою системи користувачеві.

Синтез дерева рішень зводиться до такої задачі: відома вибірка інвестора

$$T = \{X^i, Y^i, i = \overline{1,n}\},$$

де X^i - вектор стану ринку, сформований інвестором у вигляді i оцінки; Y^i - реальний результат прийняття рішення для i оцінки інвестора; n - кількість оцінок. Необхідно за допомогою сформованої початкової вибірки побудувати функцію

$$X_1 \times X_2 \times ... \times X_n \to Y,$$

яка перетворює простір чинників на прогнозування класу зростання прибутку.

Автоматизована система, здібна до самонавчання і вироблення порад користувачеві складається з трьох блоків: база оцінок стану ринку, класифікація оцінки методом дерева рішень, автоматизована підсистема прийняття рішень (рис. 4.46).

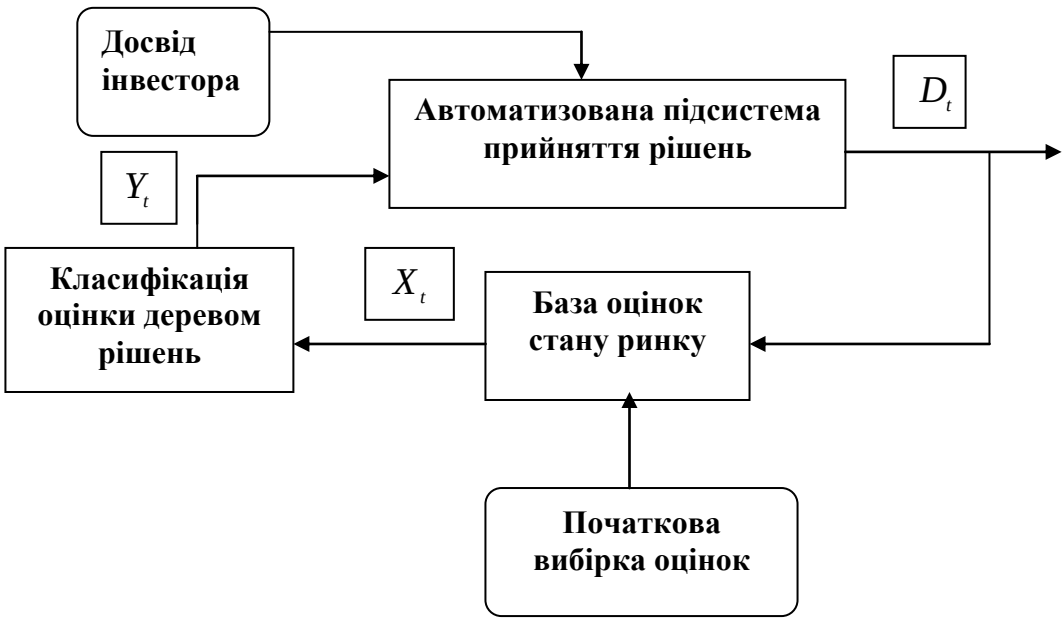


Рис. 4.46. Структурна схема автоматизованої системи прийняття рішень.

За допомогою початкової вибірки оцінок стану ринку система формує дерево рішень, яке надалі дозволить класифікувати подальші вектори оцінок у клас зростання прибутку. Отримавши результати класифікації, система виробляє пораду інвесторові, на основі якого і враховуючи власний досвід інвестор приймає рішення що заноситься в базу оцінок стану ринку і надалі використовуватиметься системою для класифікації наступної оцінки.

Тест.

1. В якому випадку, на Ваш погляд, виникають асоціації?

а) якщо виникає потреба аналізу великої кількості даних;

б) в випадку, коли треба зв'язати дані з різних баз даних;

в) якщо потрібно підвищити ефективність процес продажу товарів;

г) в випадку, коли декілька подій зв'язано одна з одною.

2. Під асоціативними правилами Ви розумієте …

а) кількісний аналіз подій;

б) опис зв'язків між декількома подіями;

в) якісний аналіз подій.

3. Цікавими вважаються такі асоціативні правила, які …

а) перевищують певний поріг;

б) мають кількісні та якісні характеристики подій;

в) пов'язують найбільшу кількість подій.

4. Дайте формальне визначення поняттю «асоціативне правило»:

а) асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \subset I$, $Y \subset I$ і $X \cap Y = 0$;

б) асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \not\subset I$, $Y \not\subset I$ і $X \cap Y = 0$;

в) асоціативним правилом називається імплікація $X \Rightarrow Y$, де $X \subset I$, $Y \subset I$ і $X \in Y$.

5. Під поняттям «підтримки асоціативного правила» Ви розумієте …

а) правило $X \Rightarrow Y$ має підтримку s (support), якщо s% транзакцій з D, містять $X \cap Y$, $\sup p(X \Rightarrow Y) = \sup p(X \cap Y)$;

б) правило $X \Rightarrow Y$ має підтримку s (support), якщо s% транзакцій з D, містять $X \cup Y$, $\sup p(X \Rightarrow Y) = \sup p(X \cup Y)$;

в) правило $X \Rightarrow Y$ має підтримку s (support), якщо s% транзакцій з D, містять $X \in Y$, $\sup p(X \Rightarrow Y) = \sup p(X \in Y)$.

6. Який зміст вкладається в поняття «достовірність»?

а) достовірність правила показує яка частина X міститься в Y;

б) достовірність правила показує яка ймовірність того, що з X виходить Y;

в) достовірність правила показує співвідношення X і Y.

7. Які типи правил серед перелічених, Ви відносите до асоціативних правил:

а) арифметичні асоціативні правила;

б) булеві асоціативні правила;

в) узагальнені асоціативні правила;

г) статистичні асоціативні правила;

д) кількісні асоціативні правила;

ж) формальні асоціативні правила;

з) непрямі асоціативні правила.

8) Визначте сутність алгоритму Apriori:

а) робота даного алгоритму складається з таких основних кроків як пошуку асоціацій та формування правил;

б) робота даного алгоритму складається з таких основних кроків як формування кандидатів та підрахунку асоціацій;

в) робота даного алгоритму складається з таких основних кроків як формування кандидатів та підрахунок кандидатів.

9. Якою особливістю володіє алгоритм AprioriTid на відміну від алгоритму Apriori?

а) AprioriTid покращує Apriori за рахунок того, що використовує базу даних при багатьох проходах;

б) алгоритм AprioriTid покращує Apriori за рахунок того, що використовує базу даних лише при першому проході;

в) AprioriTid покращує Apriori за рахунок того, що використовує базу даних лише при непарних проходах;

г) AprioriTid покращує Apriori за рахунок того, що використовує базу даних лише при парних проходах.

10. Які нові механізми були запропоновані в алгоритмі DHP:

а) механізми прямого хешування і обрізання даних;

б) механізми оптимізації запитів і обрізання даних;

в) механізми прямого хешування і розрахунку даних.

11. Під поняттям «дерева рішень» Ви розумієте …

а) технологію представлення правил в ієрархічній, послідовній структурі, де кожному об'єкту відповідає єдиний вузол, що дає рішення;

б) технологію представлення правил в послідовній структурі, де кожному об'єкту відповідає єдиний вузол, що дає рішення;

в) технологію представлення правил в ієрархічній, послідовній структурі, де кожному об'єкту відповідає один або декілька вузлів, що дають рішення.

12. Що розуміється під поняттям «правило» в технології дерев рішень?

а) під правилом розуміється логічна конструкція, представлена у вигляді «чому ... як ...»;

б) під правилом розуміється логічна конструкція, представлена у вигляді «якщо ... то ...»;
в) під правилом розуміється логічна конструкція, представлена у вигляді «якщо ... тоді ...».

13. Вважається, що сферою застосування дерев рішень є …
а) задачі ідентифікації і кластеризації;
б) задачі пошуку асоціацій і прогнозування
в) задачі класифікації і прогнозування.

14. Критерій розщеплювання це є …
а) правило, що дозволяє розбити множину на підмножини, які б асоціювалися з даним вузлом перевірки;
б) число, що дозволяє розбити множину на підмножини, які б асоціювалися з даним вузлом перевірки
в) ознака, що дозволяє розбити множину на підмножини, які б асоціювалися з даним вузлом перевірки.

15. При побудові дерев рішень найчастіше застосовуються такі критерії розщеплювання:
а) міра оптимальності і індекс споживчих цін;
б) міра ентропії і індекс Gini;
в) міра різноманітності і індекс активності.

16. Побудувати оптимальне дерево рішень можна на основі стратегій …
а) довільного нарощування дерева будь якого розміру з подальшим аналізом вузлів;
б) нарощування дерева до певного розміру відповідно до параметрів, заданих користувачем;

в) скорочення дерева шляхом відсікання гілок та використання правил зупинки навчання.

17. Якість класифікаційної моделі, побудованої за допомогою дерева рішень, характеризується ознаками:

а) кількістю вузлів та мірою інформативності;

б) мірою оптимальності та кількістю гілок;

в) точністю розпізнавання і помилкою.

18. Сутність алгоритму CART полягає в тому, що …

а) кожен вузол дерева при розбитті має багато нащадків і вирішує задачі класифікації і регресії;

б) кожен вузол дерева при розбитті має лише двох нащадків і вирішує задачі класифікації і регресії;

в) кожен вузол дерева при розбитті має лише двох нащадків і вирішує задачі класифікації.

19. Сутність алгоритму C4.5 полягає в тому, що …

а) кількість нащадків у вузла не обмежена і вирішує задачі класифікації;

б) кількість нащадків у вузла обмежена і вирішує задачі класифікації та регресії;

в) кількість нащадків у вузла не обмежена і вирішує задачі пошуку асоціацій.

20. Визначте етапи конструювання дерев рішень:

а) оцінки навчальної множини;

б) побудови дерева на основі навчальної вибірки;

в) розділення даних на класи згідно статистичним критеріям;

г) зупинки побудови дерева;

д) скорочення дерева.

Розділ 5.

ЕВОЛЮЦІЙНІ ТЕХНОЛОГІЇ ТА ГЕНЕТИЧНІ АЛГОРИТМИ

5.1. Концептуальні засади еволюційної теорії.

Еволюційна теорія, синонімом якої в зарубіжній літературі є термін «Evolutionary computation», довела свою ефективність як при вирішенні складно формалізованих задач штучного інтелекту (розпізнавання образів, кластеризація, асоціативний пошук), так і при вирішенні трудомістких задач оптимізації, апроксимації, інтелектуальної обробки даних. До переваг еволюційної теорії відносяться адаптивність, здібність до навчання, паралелізм, можливість побудови гібридних інтелектуальних систем на основі комбінування з парадигмами штучних нейромереж і нечіткої логіки. Багатообіцяючою виглядає передумова створення єдиної концепції еволюційних обчислень, що включають генетичні алгоритми, генетичне програмування, еволюційні стратегії і еволюційне програмування. На думку багатьох дослідників, ці парадигми є аналогами процесів, що відбуваються в живій природі, і на практиці довели свою непримітивність. Один з піонерів еволюційної теорії Л. Фогель взагалі бачить теорію еволюції і самоорганізації як базову концепцію для всіх інтелектуальних процесів і систем, що значно розширює сферу застосування традиційної парадигми інтелектуального аналізу даних. Навіть якщо це не так, і в природі відбувається революция, ніхто не може сказати, що алгоритми еволюційних обчислень невірні [7, 12, 18].

Основна теза підходу, названого еволюційним, - замінити процес моделювання складного об'єкту моделюванням його еволюції. Він спрямований на застосування механізмів природної еволюції при аналізі складних систем інтелектуальної обробки інформації. У своїй теорії походження видів Ч. Дарвін відкрив і обґрунтував основний закон розвитку органічного світу, охарактеризувавши його взаємодією трьох наступних чинників:

- спадкоємній мінливості;
- боротьби за існування;
- природного відбору.

Дарвінівська теорія отримала підтвердження і розвиток в генетиці та інших науках. Одним з найвідоміших еволюціоністів, нашим співвітчизником І. І. Шмальгаузеном були висунуті наступні необхідні і достатні умови, що визначають неминучість еволюції:

1. Спадкоємна мінливість, тобто мутації як передумова еволюції, її матеріал.
2. Боротьба за існування як контролюючий і направляючий чинник.
3. Природний відбір як перетворюючий чинник.

Моделювання еволюції можна розділити на дві категорії:

1. Системи, які використовують тільки еволюційні принципи. Вони успішно використовувалися для задач типу функціональної оптимізації і можуть легко бути описані на математичній мові. До них відносяться еволюційні алгоритми, такі як еволюційне програмування, генетичні алгоритми та еволюційні стратегії.

2. Системи, які є біологічно реалістичнішими, але які не виявилися корисними в прикладному сенсі. Вони більш схожі на біологічні системи і менш направлені на вирішення технічних задач та інтелектуального аналізу даних. Вони володіють складною і цікавою поведінкою, і, мабуть, незабаром отримають практичне застосування. До цих систем відносять так зване «штучне життя».

Еволюційні обчислення - термін, зазвичай використовують для загального опису алгоритмів пошуку, оптимізації або навчання, заснованих на формалізованих принципах природного еволюційного процесу. Еволюційні методи призначені для пошуку переважних рішень і засновані на статистичному підході до дослідження ситуацій та ітераційному наближенні до шуканого стану систем. На відміну від точних методів дослідження операцій еволюційні методи дозволяють знаходити рішення, близькі до оптимальних, за прийнятний час, а у відмінності від відомих евристичних методів оптимізації

характеризуються істотно меншою залежністю від особливостей додатку (тобто більш універсальні) і у багатьох випадках забезпечують кращий ступінь наближення до оптимального рішення. Основна перевага еволюційної теорії полягає в можливості вирішення багатомодальних (що мають декілька локальних екстремумів) задач із великою розмірністю за рахунок поєднання елементів випадковості і детермінованості точно так, як це відбувається в природному середовищі.

Детермінованість цих методів полягає в моделюванні природних процесів відбору, розмноження і спадкоємства, що відбуваються по строго певних правилах, основним з яких є закон еволюції - «виживає сильніший». Іншим важливим чинником ефективності еволюційних обчислень є моделювання процесів розмноження і спадкоємство. Дані варіанти рішень можуть за певним правилом породжувати нові рішення, які успадковуватимуть кращі риси своїх «предків». Як випадковий елемент в теорії еволюційних обчислень використовується моделювання процесу мутації. З її допомогою характеристики того або іншого рішення можуть бути випадково змінені, що приведе до нового напряму в процесі еволюції рішень і може прискорити процес вироблення кращого рішення.

Методи еволюційної теорії також часто використовуються для опису процесів еволюції програм або функцій (генетичне програмування), кінцевих автоматів (еволюційне програмування) і систем, заснованих на продукційних правилах (класифікаційні системи). Вони застосовуються для навчання штучних нейронних мереж або разом з нейронними мережами для локального пошуку екстремуму цільової функції, а також часто з нечіткою логікою. Для того, щоб описувати генетичні алгоритми, нейроні мережі і нечітку логіку в їх різних поєднаннях, був навіть введений спеціальний термін «м'які обчислення».

Історія еволюційних обчислень почалася з розробки ряду різних незалежних моделей еволюційного процесу. Серед цих моделей слід виділити декілька основних парадигм: генетичні алгоритми, генетичне програмування, еволюційні стратегії, еволюційне програмування. Поняття «еволюційне

моделювання» вперше сформувалося в роботах Л. Фогеля, А. Оуені, М. Уолша. У 1966 році вийшла їх спільна книга «Штучний інтелект і еволюційне моделювання». У далекі 60-і роки Інго Рехенберг (I. Rechenberg), зацікавлений методом «органічної еволюції», висунув ідею вирішення оптимізаційних проблем в аеродинаміці, застосовуючи мутації до вектора параметрів. Ця процедура стала відома під назвою «еволюційної стратегії» (Evolution Strategies). У 1981 році Швефель (H. Schwefel) при дослідженні гідродинамічних задач увів рекомбінації в еволюційну систему та виконав порівняльний аналіз з класичними методами оптимізації. Приблизно в той же час в США незалежно виконувалися дослідження Лоуренсом Фогелем (Lawrence Fogel) еволюції штучного інтелектуального автомата з скінченим числом станів, використовуючи метод, названий «еволюційним програмуванням» (Evolutionary Programming). Джон Холланд (John Holland) аналізував клас репродуктивних систем методом, який нам відомий як генетичний алгоритм (Genetic Algorithms). Така класифікація еволюційних алгоритмів була б неповною без робіт Lynn Cramer (1985), Jac Hicklin (1986), Gory Fujiki (1987), результати яких узагальнив і розширив Джон Коза (John Koza). Запропонований метод назвали генетичним програмуванням (Genetic Programming) [34, 62, 66, 67]. Еволюційні алгоритми відрізняються один від одного. Але всі вони базуються на принципах еволюції:

1. Особі мають скінчений час життя; розмноження необхідне для продовження роду.
2. В деякій мірі нащадки відрізняються від батьків.
3. Особі існують в середовищі, в якому виживання є боротьбою за існування, і їх зміни сприяють кращій адаптації до умов зовнішнього середовища.
4. За допомогою природної селекції краще адаптовані особі мають тенденцію до довшого життя і виробництва більшої кількості нащадків.
5. Нащадкам властиво успадковувати корисні характеристики своїх батьків, що спричиняє збільшенню пристосованості особі в часі.

Розвиток теорії еволюційних обчислень в нашій країні почався ще з початку 80-х років, проте отримані результати довгий час залишалися, в основному, мало відомими. Ці дослідження в значній мірі базуються на раніше опубліковані роботи по самонавчальних системах (Івахненко, Ципкін), по стохастичній оптимізації (Растрігин). Кожна з цих «шкіл» взяла за основу ряд принципів, що існують в природі, і спростила їх до такого ступеня, щоб їх можна було реалізувати на комп'ютерах того часу. Були розроблені цікаві теорії, щоб описати переваги запропонованих підходів, але всі вони були змушені розділити і загальну проблему того часу - комп'ютери були недостатньо потужними, щоб вирішувати прикладні задачі такої розмірності, які не могли бути розв’язані традиційними методами. Це пояснюється тим, що еволюційні методи не здатні «забиратися» на найближчий «пагорб», виявляти оптимальні навчальні правила і тому подібне з такою ж швидкістю, як це можуть робити інші методи. Проте, коли задача не може бути вирішена іншими, простішими методами, еволюційні обчислювальні методи можуть знайти оптимальні або близькі до них рішення. Необхідний при цьому обсяг підрахунків може виявитися великим, але швидкість, з якою цей обсяг зростає при збільшенні «розмірності» задачі, часто менше, ніж для інших методів. Тому, як тільки обчислювальні потужності стали відносно доступними і недорогими, еволюційні обчислення перетворилися на важливий інструмент пошуку субоптимальних рішень тих задач, які до останнього часу вважалися за нерозв'язні.

В даний час відомою світовою школою, що представляє новий напрям в еволюційному моделюванні, є школа Кандіда Феррера (Candida Ferreira) у Великобританії. Основний напрям її досліджень зосереджений в програмуванні генетичних виразів. Нові алгоритми, які розробляються представниками школи, використовують специфічні оператори комбінаторного пошуку, що включають інверсію, вставку і видалення генів та їх послідовностей, обмеження і узагальнення перестановок, які збільшують їх ефективність. Самі автори визначають програмування генетичних виразів як мультигенне

генотип/фенотип кодування дерев виразів, зв'язаних приватною взаємодією. Іншою відомою школою, в якій досліджують генетичні алгоритми, еволюційні стратегії, генетичне програмування і еволюційне програмування є лабораторія еволюційних обчислень Департаменту комп'ютерних наук в університеті Джорджа Мейсона. У США керівництво школою здійснює учень Джона Холланда К. Де Йонг (K. De Jong). Лабораторія працює над проектами і додатками моделей еволюції (у дарвінівському сенсі). Такі моделі необхідні для кращого розуміння еволюційних систем, вони використовуються для забезпечення робастності, гнучкості і адаптивності обчислювальних систем. Головну увагу фахівці лабораторії приділяють вирішенню складних наукових і технічних проблем, таких, як інноваційне проектування, оптимізація і машинне навчання. У аналогічному напрямі, але з акцентом на генетичні алгоритми працює наукова школа Девіда Голдберга (David E. Goldberg). Лабораторія генетичних алгоритмів знаходиться в Іллінойському університеті США (рис. 5.1).

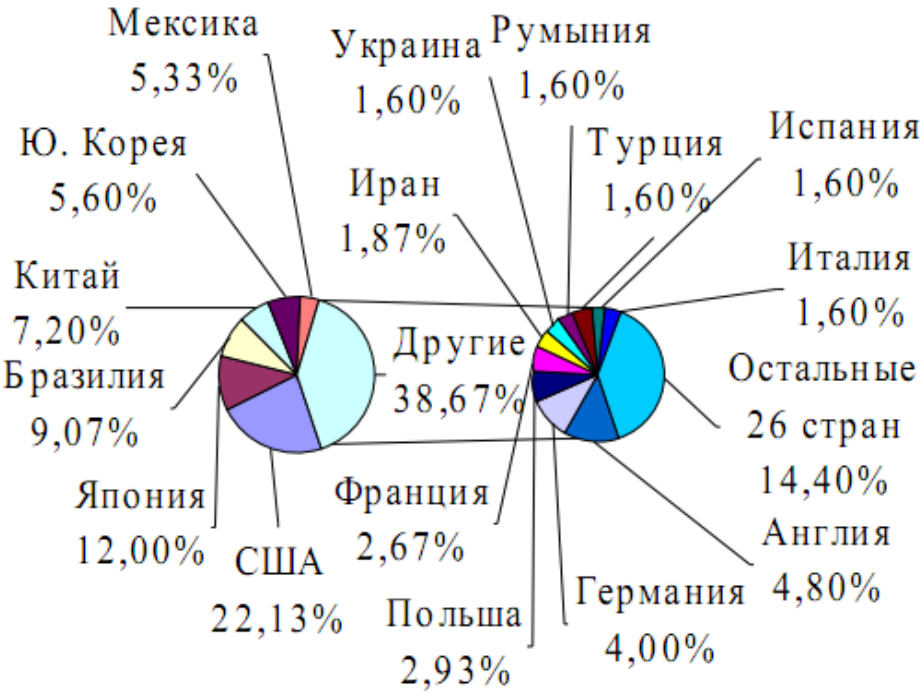


Рис. 5.1. Внесок фахівців різних країн в розвиток теорії еволюційних обчислень за 2000 – 2008 р.

Оскільки концепція еволюційних обчислень має бути заснована на деяких формалізованих принципах природного еволюційного процесу, то виникають питання про методологічні відмінності і сферу застосування основних форм еволюційних технологій. Результати порівняльного аналізу відомих форм еволюційних алгоритмів показують певні методологічні відмінності між ними. Ці відмінності стосуються форми представлення цільової функції і альтернативних рішень, операторів рекомбінації, мутації і ймовірності їх використання, стратегії селективного відбору і методів підвищення ефективності еволюційних обчислень шляхом адаптації.

Методологічні відмінності між різними формами еволюційних обчислень дозволяють говорити про базові постулати, такі як універсальність і фундаментальність, властивих еволюції незалежно від форми і рівня абстракції моделі. Вказана спільність може бути виражена у вигляді наступної схеми абстрактного еволюційного алгоритму:

1. Установка параметрів еволюції;
2. Ініціалізація початкової популяції $P(0)$;
3. $t = 0$;
4. Оцінка рішень, що входять в популяцію;
5. $t = t + 1$;
6. Селекція (відбір);
7. Реплікація (повторення, копіювання, аутосинтез);
8. Варіація (видозміна);
9. Оцінка рішень-нащадків;
10. Утворення нової популяції $P(t)$;
11. Виконання алгоритму до тих пір, поки параметр t не досягне заданого значення $t_{\max}$ або не будуть виконані інші умови зупинення;
12. Виведення результатів і зупинення.

Вирішальною обставиною для оцінки практичної придатності і результативності еволюційного алгоритму є швидкість (час, необхідний для виконання заданого користувачем числа ітерацій) і стійкість пошуку (здатність

постійно від покоління до покоління збільшувати якість популяції та стійкість до попадання в точки локальних екстремумів).

Спроба порівняльного аналізу конкуруючих евристичних алгоритмів з погляду їх результативності демонструє два важливі практичні наслідки:

1. Пошук еволюційного алгоритму, який перевершує всі алгоритми, що конкурують з ним, не має сенсу без точного опису конкретних задач і цільових функцій, для яких еволюційний алгоритм має перевагу перед іншими алгоритмами. Не можна розраховувати знайти один алгоритм, який буде результативніше останніх для будь-яких цільових функцій оптимізації.

2. Щоб знайти добре рішення для заданого класу задач, необхідно спочатку ідентифікувати характеристичні особливості класу задач і тоді на їх основі шукати відповідний алгоритм.

З цієї точки зору еволюційні обчислення володіють наступними достоїнствами і недоліками. Достоїнства еволюційних обчислень:

- широка сфера застосування;
- можливість проблемно-орієнтованого кодування рішень, підбору початкової популяції, комбінування еволюційних обчислень з не еволюційними алгоритмами, продовження процесу еволюції до тих пір, поки є необхідні ресурси;
- придатність для пошуку в складному просторі рішень великої розмірності;
- відсутність обмежень на вигляд цільової функції;
- ясність схеми і базових принципів еволюційних обчислень;
- інтегрованість еволюційних обчислень з іншими некласичними парадигмами штучного інтелекту, такими, як штучні нейромережі і нечітка логіка.

Недоліками еволюційних обчислень є:

- евристичний характер еволюційних обчислень не гарантує оптимальності отриманого рішення (правда, на практиці, часто, важливо за заданий час отримати одне або декілька субоптимальних альтернативних рішень, тим

більше, що початкові дані в задачі можуть динамічно мінятися, бути неточними або неповними);

- відносно висока обчислювальна трудомісткість, яка проте долається за рахунок розпаралелювання на рівні організації еволюційних обчислень і на рівні їх безпосередньої реалізації в обчислювальній системі;
- відносно невисока ефективність на завершальних фазах моделювання еволюції (оператори пошуку в еволюційних алгоритмах не орієнтовані на швидке попадання в локальний оптимум);
- невирішеність питань самоадаптації.

Таким чином, порівняльний аналіз показує, що еволюційні обчислення найменше підходять для вирішення задач, в яких потрібно знайти глобальний оптимум; є ефективний не еволюційний алгоритм; змінні, від яких залежить рішення незалежні; для задачі характерний високий ступінь епістазії (одна змінна пригнічує іншу); значення цільової функції в усіх точках, за винятком оптимуму, є приблизно однаковими. Принципово підходящими для вирішення за допомогою еволюційних обчислень є задачі багатовимірної оптимізації з мультимодальними цільовими функціями, для яких немає відповідних не еволюційних методів рішення; стохастичні задачі; динамічні задачі з блукаючим оптимумом; задачі комбінаторної оптимізації; задачі прогнозування і кластеризації.

Еволюційні обчислювальні методи сьогодні успішно застосовуються для вирішення ряду великих і економічно значущих задач в бізнесі і інженерних розробках. За їх допомогою були розроблені промислові проекти, що дозволили заощадити мільйони доларів. Фінансові компанії широко використовують еволюційні методи прогнозування розвитку фінансових ринків для управління портфелями цінних паперів і так далі Методи еволюційної теорії зазвичай використовуються для оцінки значень безперервних параметрів моделей великої розмірності, для вирішення NP-повних комбінаторних задач, для оптимізації моделей, що включають одночасно безперервні і дискретні параметри, в системах видобування нових знань з великих баз даних (data

mining), для побудови і навчання стохастичних мереж та байєсовських мереж довіри, для навчання нейронних мереж, для оцінки параметрів в багатовимірного статистичного аналізу, а також у поєднанні з різними евристичними процедурами. Сферами застосування таких технологій є проектування механічних пристроїв, розводка друкованих плат і розміщення на них електронних компонентів, проектування деталей реактивних двигунів, розробка складних планів і розкладів, оптимізація розміщення промислового устаткування, а також багаточисельні приклади вирішення оптимізаційних задач великої розмірності звеликим числом обмежень. Ніхто не стверджує, що еволюційні обчислення однаково «гарні» для вирішення всіх типів задач; при цьому є досить багато прикладів їх успішного застосування, хоча успіх зазвичай досягається тільки після ретельного налаштування структури і параметрів еволюційного методу.

Одним з найбільш відомих еволюційних алгоритмів є алгоритм на основі методу групового обліку аргументів (МГОА). Сутність цього методу в тому, що розробляється сімейство індуктивних алгоритмів для моделювання мультипараметричних даних. Метод заснований на рекурсивному селективному відборі моделей, на основі яких будуються складніші моделі. Точність моделювання на кожному наступному кроці рекурсії збільшується за рахунок ускладнення моделі. Згідно термінології теорії еволюції його в загальному вигляді можна .представити як слідуючий цикл:

1. Беремо найостанніший шар моделей.
2. Генеруємо з них по певних правилах новий шар моделей (які тепер самі стають останнім шаром).
3. Відбираємо з них F кращих, де F - ширина відбору (селекції).
4. Якщо не виконується умова припинення селекції (настання звироднілості), переходити на п. 1.
5. Сама краща модель оголошується шуканим рішенням задачі ідентифікації.

Як ми бачимо, в наявності всі ознаки еволюційного алгоритму - відбір (селекція) і генерація нового покоління. Розглянемо метод групового обліку аргументів більш докладніше.

Цей метод розробляється академіком НАН України А.Г. Івахненко і його школою і є типовим методом індуктивного моделювання і одним з найбільш ефективних методів структурно-параметричної ідентифікації складних об'єктів, процесів і систем за даними спостережень в умовах неповноти інформації. В цілому задача полягає в генеруванні за даними спостережень деякої множини F моделей різної структури виду

$$\tilde{y}_f = f(X, \tilde{\Theta}_f)$$

і відшуканні оптимальної моделі по умові мінімізації деякого критерію якості моделей

$$f^* = \arg\min C(y, f(X, \tilde{\Theta}_f)),$$

причому оцінки параметрів для кожної моделі є рішенням ще однієї екстремальної задачі виду

$$\tilde{\Theta}_f = \arg\min Q(y, X, \Theta_f),$$

де s_j називається складністю моделі f і дорівнює числу невідомих параметрів в моделі f^*, а Q - критерій якості рішення задачі параметричної ідентифікації кожної приватної моделі, що генерується в задачі структурної ідентифікації.

МГОА володіє певною різноманітністю можливостей на всіх етапах процесу моделювання в порівнянні з іншими методами побудови моделей. Це стосується перш за все генераторів моделей і вживаних критеріїв якості структур, а також класів моделей (базисних функцій). Метод відрізняється активним застосуванням принципів автоматичної генерації варіантів, послідовній селекції моделей і зовнішніх критеріїв для побудови моделей оптимальної складності. Він має оригінальну процедуру багаторядності автоматичної генерації структур моделей, що імітує процес біологічної селекції з попарним обліком послідовних ознак. Така процедура в сучасній термінології

називається поліноміальною нейронною мережею, причому її структура є явною і будується автоматично, в режимі самоорганізації.

Для порівняння і вибору кращих моделей застосовуються зовнішні критерії, засновані на розділенні вибірки на дві і більш частин, причому оцінювання параметрів і перевірка якості моделей виконується на різних підвибірках. Це дозволяє обійтися без обтяжливих апріорних припущень, оскільки розділення вибірки дозволяє неявно (автоматично) врахувати різні види апріорної невизначеності при побудові моделі. МГОА володіє перевагою при малих вибірках даних за рахунок вибору складності моделі, що оптимально враховує інформативність даних.

Ефективність методу багато разів підтверджувалася вирішенням множини конкретних задач з областей екології, економіки, гідрометеорології і так далі. Зокрема, на основі аналогії між задачею побудови моделі по зашумленим експериментальним даним і задачею проходження сигналу через канал з шумом побудовані начала теорія помехостійкого моделювання. Основний результат цієї теорії полягає в тому, що складність оптимальної прогнозуючої моделі залежить від рівня невизначеності в даних: чим він вищий - тим простіше (грубіше) має бути оптимальна модель (тим менше оцінюваних параметрів).

МГОА добре відомий і вельми активно розвивається у нас в країні і за кордоном. Розроблені основи теорії структурної ідентифікації моделей з мінімальною дисперсією помилки прогнозування. Ефективним апаратом цієї теорії є метод критичних дисперсій, що дозволяє вперше аналітично вирішувати актуальні задачі: порівняльний аналіз критеріїв структурної ідентифікації, планування експериментів, аналіз властивостей методів і тому подібне, причому як при обмеженій вибірці, так і в асимптотиці. При цьому досліджуються умови вибору оптимальної структури моделі залежно від дисперсії (рівня) шуму, довжини вибірки, вхідних дій (плану експерименту) і параметрів об'єкту, причому встановлений тісний взаємозв'язок між ними. Засобами цієї теорії встановлено, що МГОА є методом побудови моделей з

мінімальною дисперсією помилки прогнозування, і виконано його порівняння з іншими методами. Із цього виходить, що МГОА як основний інструмент теорії індуктивного моделювання відноситься до найбільш сучасних методів інтелектуального аналізу даних і м'яких обчислень. Цей метод є оригінальним і ефективним засобом вирішення широкого спектру задач штучного інтелекту, зокрема ідентифікації і прогнозування, розпізнавання образів і кластеризації, інтелектуального аналізу даних і пошуку закономірностей.

У останнє десятиліття інтерес до МГОА активно зростає у всьому світі, що можна пояснити, окрім відомої ефективності методу, також зростанням популярності технології штучних нейромереж. Річ у тому, що структуру МГОА можна інтерпретувати як нейромережу, оригінальність якої полягає в самоорганізації як її структури, так і параметрів. При цьому виявляється, що до явних переваг методу відносяться автоматичне формування структури мережі, простота і швидкодія настроювання параметрів, а також можливість «згорнути» настроєну мережу безпосередньо в явний математичний вираз [2, 9, 39].

Підхід індуктивного моделювання, побудований на принципах самоорганізації, розвивається протягом 40 років, застосовується в багатьох областях і присутній в таких поширених технологіях аналізу даних, як поліноміальні нейроні мережі, адаптивні і статистичні мережі, що навчаються. У нових розробках для побудови моделей на основі даних використовуються також еволюційні і генетичні алгоритми, ідея активних нейронів і багаторівнева самоорганізація, та інші ідеї. Комп’ютерний варіант системи МГОА реалізован в програмному комплексі NeuroShell компанії Ward Systems Group.

Спроби організувати штучну еволюцію в обчислювальному середовищі, ґрунтуючись на основних принципах біологічної еволюції - принципах мінливості і відбору, робилися ще в 70-х роках минулого століття. Досліджувалася еволюція машин Тьюрінга, клітинних автоматів, був розроблений генетичний алгоритм, але відсутність достатніх обчислювальних потужностей заважала поширенню еволюційних алгоритмів. Адже еволюція в природі «обраховується» на гігантській паралельній машині, що складається зі

всіх живих істот. Кожен новонароджений організм - це нове рішення для певного завдання.

Із зростанням продуктивності комп'ютерів еволюційні алгоритми набули широкої популярності. Спочатку параметри алгоритмів підстроювалися вручну. Відома історія про деякого аспіранта на ім'я Олівер, який задавав параметри і запускав еволюційний алгоритм оптимізації вагів для роботи штучної нейронної мережі, а сам йшов в паб. Потім з'явилися методи самоорганізації параметрів еволюційних алгоритмів. Тепер Олівер запускав алгоритм для пошуку оптимальних параметрів еволюційного алгоритму оптимізації вагів для роботи штучної нейронної мережі і йшов в паб. Олівер, напевно, незабаром спився б при такій «науковій роботі», але допомогли колеги, що довели так звану No Free Lunch Theorem. Теорема стверджує, що для будь-якого стохастичного алгоритму пошуку, до яких відносяться і еволюційні алгоритми, середня ефективність дорівнює простому випадковому пошуку. Іншими словами, якщо нам необхідно вирішувати задачі, які не схожі одна на одну, то ми можемо використовувати еволюційний алгоритм з тим же успіхом, що тикати пальцем в небо. Але ж нам не обов'язково потрібний алгоритм, щоб вирішувати будь-які задачі, зазвичай ми маємо справу із задачами, що відносяться до класу, який в теорії еволюційних обчислень отримав назву «Real World Problems» (реальні задачі). Деякі дослідники вважають, що для реальних задач універсальний алгоритм існувати може.

Останніми роками відмінним полігоном для еволюційних технологій став Інтернет. Так, в сумісному проекті Колумбійського університету і університету штату Айова розроблена мультиагентна система InfoSpiders призначена для пошуку інформації в Інтернеті. Інфопаучки сканують мережу у пошуках ресурсів, що задовольняють запиту користувача. Тим, хто знайде «правильні» ресурси, видається енергія. Чим більше у павука енергії, тим вище вірогідність того, що він виживе і залишить потомство - так популяція інфопаучків еволюціонує.

Ще одна сфера застосування еволюційних алгоритмів в Інтернеті - оптимізація мережевих екранів (firewall). Ідея застосувати еволюційні методи для аналізу мережевого трафіку виникла у фахівців компанії MASA Group , що займаються розробкою комп'ютерних адаптивних систем. На той час компанією була створена бібліотека алгоритмів по оптимізації, машинному навчанню, а також по визначенню змін в складних образах (novelty detection). Фахівці MASA Group вирішили, що при захисті мережі основний упор треба робити не на порівняння мережевої активності з шаблонами відомих атак, як в звичайних екранах, а на визначенні відхилень від норми в роботі мережі. Адже якщо вторгнення проводиться новим способом, невідомим системі, то звичайний підхід не зможе визначити атаку. В результаті співпраці MASA Group і MATRAnet ці ідеї втілилися в програмний продукт M>Detect, який, по завіренню представників компаній, повинен з'явитися на ринку восени цього року. Робота M>Detect складається з декількох етапів. На першому етапі система за допомогою еволюційного алгоритму навчається «нормальному» режиму роботи мережі. До кінця навчання формується набір фільтрів, що описують дозволені потоки даних. Тепер M>Detect може переходити до стежачого режиму, в якому використовується система розпізнавання новизни в образах (на основі тимчасових вікон різної тривалості, що дозволяє детектувати як «швидкі», так і «повільні» атаки). Як тільки в мережевому трафіку виникають аномалії, програма посилає системному адміністраторові попередження і лог активності. Все це дозволяє M>Detect виявляти мережеві атаки на ранній стадії, а також реагувати на невідомі типи атак. Якщо попередження було помилковим, то оператор перемикає систему в режим «дообучання», що дозволяє уникнути повторного помилкового спрацьовування. По оцінках розробників, помилкових попереджень буде не більше одного-двох в добу, при цьому об'єм трафіку може досягати 30 Гбайт/с, що, поза сумнівом, є добрим результатом.

Використовують еволюційні алгоритми в боротьбі з комп'ютерними вірусами. В цьому випадку антивірусна система схожа на імунну систему

людини, яка постійно генерує антитіла. При виникненні в машині шкідливої активності штучна імунна система «збуджується», що означає зростання числа випадково створених модифікацій шаблонів вірусів, які були відомі системі. Комп'ютер починає «хворіти». Як і в хворому організмі, під «гарячу руку» збудженої імунної системи можуть потрапляти не тільки чужорідні елементи, але і те, що необхідне для нормальної роботи самого «хворого».

Одна з областей еволюційного комп’ютинга, що найактивніше розвивається, - еволюційна робототехніка. Створення евоботів (еволюційних роботів) зазвичай засноване на еволюції комп'ютерної моделі. Спочатку прототип майбутнього евобота вчиться пересуватися. Якщо це крокуючий робот, то спочатку його ноги заплітають, він часто падає і не може встати, але вже декілька поколінь через окремі віртуальні прототипи майбутнього евобота жваво бігають. Потім наступає етап формування системи управління поведінкою. Залежно від цілей конструктора в процесі штучної еволюції відбираються нейроні мережі, що забезпечують лавірування між перешкодами або, наприклад, збирання об'єктів. Після того, як прототип успішно витримав всі випробування, програма «заливається» в «залізо» реального робота.

Інше цікаве застосування технологія побудови евоботів знайшла в створенні комп'ютерних ігор. Сучасні комп'ютерні ігри насичені масою персонажів, які взаємодіють з гравцем і між собою. Як зробити їх поведінку непередбачуваною, індивідуальною і навіть осмисленою? Надайте можливість попрацювати природному відбору на віртуальному ігровому світі, і ось вона - нова реальність, гра стала набагато багатша.

Сьогодні наші технології управляють навколишнім світом прямо. Створюючи всілякі пристрої і програми, людина обмежує міри свободи середовища. Але вже з'являються системи, в яких управління перенесене на наступний рівень - рівень управління процесами самоорганізації. Це і роботи, що самоконфігуруються, мікророботи, що самозбираються, і, звичайно ж, еволюційні алгоритми.

Бурхливий розвиток еволюційних алгоритмів привів до появи великої кількості програмних комплексів, що мають в своєму складі компоненту еволюційного пошуку. Аналіз таких програмних засобів, показує, що майже всі комерційні програмні пакети, що випускаються, є вбудовуваними в інші системи, так що всі деталі перетворення форматів даних і технології кодування, вибору типів еволюційних операторів, оціночних функцій, способу організації обчислень і так далі приховані від користувача. У існуючих гібридних системах інтеграція здійснюється тільки на рівні окремих модулів. Хоча у багатьох випадках це дає позитивний ефект, в цілому можливості закладених в інтегровані системи алгоритмів повною мірою не використовуються [17, 25].

Розглянемо деякі сучасні концепцій програмної реалізації основних технологій теорії еволюційних обчислень. Одним з відомих програмних комплексів, призначених для комп'ютерного використання технологій інтелектуального аналізу даних є система PolyAnalyst. У цій системі значна увага приділена однієї з найбільш молодих і багатообіцяючих технологій Data Mining - еволюційному програмуванню. Основна ідея методу полягає у формуванні гіпотез про залежність цільової змінної від інших змінних у вигляді автоматично синтезуємих спеціальним модулем програм на внутрішній мові програмування. Використання універсальної мови програмування теоретично дозволяє виразити будь-яку залежність, причому вид цієї залежності заздалегідь не відомий [79].

Процес виробництва внутрішніх програм організовується як еволюція в просторі програм, що нагадує генетичні алгоритми. Коли система знаходить перспективну гіпотезу, яка описує досліджувану залежність досить добре по цілому ряду критерій, в роботу включається механізм так званих "узагальнених перетворень" (GT-search). За допомогою цього механізму в «гарну» програму вводяться незначні модифікації, не погіршуючи її якість, і проводиться відбір кращої дочірньої програми. До нової популяції потім знову застосовуються механізми синтезу нових програм, і цей процес рекурсивно повторюється. Таким чином, система створює деяке число генетичних ліній програм, що

конкурують одна з однією по точності, статистичній значущості і простоті виразу залежності.

Спеціальний модуль безперервно перетворює «кращу» на даний момент програму з внутрішнього уявлення в зовнішню мову PolyAnalyst - мову символьних правил (Symbolic Rule Language), зрозумілу людині: математичні формули, умовні конструкції і так далі. Це дозволяє користувачеві зрозуміти суть отриманої залежності, контролювати процес пошуку, а також отримувати графічну візуалізацію результатів. Контроль статистичної значущості отриманих результатів здійснюється цілим комплексом ефективних і сучасних статистичних методів, включаючи методи рандомізованого тестування (рис. 5.2).

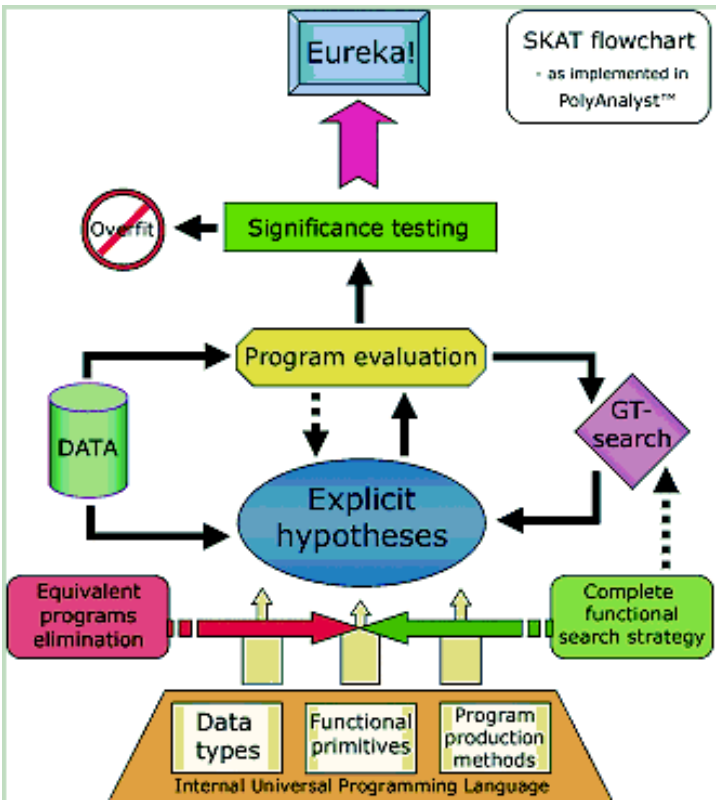


Рис. 5.2. Модуль еволюції програм системи PolyAnalyst.

Іншою сучасною концепцією є еволюційне проектування баз даних. У останні декілька років спостерігається розвиток нового підходу до розробки

програмного забезпечення, а саме застосування так званих гнучких методологій. Такий підхід диктує нові вимоги до проектування баз даних. Одна з центральних вимог пов'язана з ідеєю еволюційного проектування. У гнучкому проекті приймається той факт, що не можна заздалегідь виявити і зафіксувати вимоги до системи. В результаті цього фаза детального проектування на початковій стадії стає непродуктивною. Альтернативний варіант - розробка системи за допомогою багатьох ітерацій. Гнучкі методології, зокрема, екстремальне програмування, пропонують ряд методів, що роблять еволюційне проектування здійсненним.

Одна з ключових особливостей гнучких методик – це ставлення до змін. Більшість міркувань щодо процесу розробки програмного забезпечення зводяться до наступного: на ранньому етапі виявити вимоги, узгодити їх, спроектувати на їх основі систему, узгодити проект і потім приступити до кодування. Такий, керований планом, життєвий цикл часто називають моделлю водопаду. Подібна модель має на увазі мінімізацію змін за рахунок виконання величезної попередньої роботи. Проте після закінчення цієї роботи вартість змін різко зростає. Як наслідок, значні труднощі з'являються при подальшій зміні вимог, а текучість вимог створює вже майже нерозв'язні проблеми.

Гнучкі процеси відносяться до змін інакше. Вони прагнуть управляти змінами, допускаючи їх навіть на пізніх стадіях проектування. Зміни тримаються під контролем, але принципова позиція - максимально спростити можливість змін. Це частково є відповіддю на принципову нестабільність вимог в багатьох проектах, а також забезпечує кращу підтримку бізнес-середовища, що динамічно змінюється. Щоб все це працювало, необхідно змінити відношення до проектування. Замість того щоб розуміти проектування як фазу, що в основному завершується на початок написання коду, слід розглядати проектування як безперервний процес, який перемежається з кодуванням, тестуванням і навіть постачанням замовникові. У цьому проявляється зіставлення планового і еволюційного підходів до проектування. Замість звичного хаосу, який часто виникає за відсутності попереднього планування,

гнучкі методології дозволяють зробити еволюційне проектування здійсненним і керованим.

Важлива частина цього підходу - ітераційна розробка, при якій багато разів повторюється весь життєвий цикл програмного продукту в рамках одного проекту. Гнучкі процеси реалізують повний життєвий цикл в кожній ітерації, завершуючи кожну з них працюючим кодом, що відтестован, інтегрованим, для невеликої підмножини вимог кінцевого продукту. Ітерації короткі: від тижня до пари місяців, причому коротші переважні. Впродовж останніх декількох років був реалізований в крупний проект Atlas, в якому було використано еволюційне проектування баз даних. У проекті брало участь майже 1000 чоловік на декількох майданчиках по всьому світу (у США, Австралії, Індії). Об'єм розробки - приблизно півмільйона рядків коду і більше 2000 таблиць. База даних розроблялася впродовж півтора року, і в даний час вона експлуатується декількома замовниками і продовжує розвиватися.

Ще одним сучасним напрямом є технологія еволюційної побудови нейромереж. Як вже відомо, при побудові нейронної мережі виникає необхідність експериментувати з великою кількістю мереж, порівнюя отримані результати. Для кожної нової досліджуваної мережі визначається її придатність для рішення задачі, при необхідності воно змінюється або відбраковується. Простір всіляких нейронних мереж, в середині якого виконується пошук оптимальної мережі, дуже великий. Тому доводиться застосовувати евристичні методи. Процес – багатократне повторення циклу «вибір нової мережі – оцінка – відбраківка або модифікація» вельми нагадує природний процес еволюції, якщо в кожен момент часу спостерігати тільки за однією особою. Еволюційні методи добре зарекомендували себе в задах пошуку оптимальних рішень в складному просторі. Тому перехід до застосування еволюційних методів для пошуку хороших структур нейронної мережі є закономірним.

При застосуванні еволюційних алгоритмів для побудови нейронних мереж (Evolutionary design of neural architectures - EDNA) спочатку необхідно визначити фенотип і генотип нейронної мережі. Під фенотипом зазвичай

розуміють тільки структуру нейронної мережі (структуру зв'язків між нейронами). Такі параметри як вид та функція активації задаються в генотипі. Вагові коефіцієнти зв'язків для побудови мережі найчастіше задаються випадковим чином, після чого коректуються в процесі навчання нейронної мережі по одному з відомих методів. Проте багато дослідників використовують еволюційні методи і для корректировки вагів зв'язків (рис. 5.3).

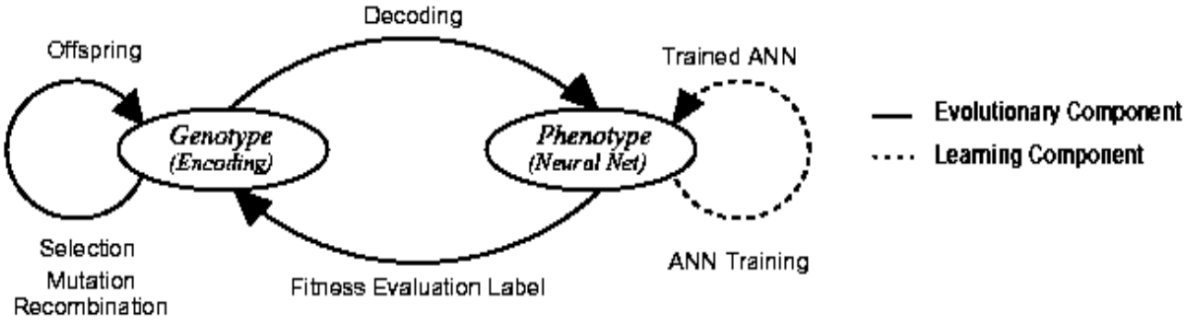


Рис. 5.3. Взаємодія між нейронною мережею та її записом в гені.

На рисунку 5.3 схематично зображена взаємодія між основними об'єктами, які беруть участь у еволюційному процесі. Таких об'єктів всього два - генотипи та фенотипи. Над генотипами проводяться всі еволюційні операції. По генотипу відновлюється фенотип, який проходить процес навчання. По результатах тестування фенотипу розраховується придатність відповідного генома.

5.2. Основні положення теорії генетичних алгоритмів.

Еволюційна теорія стверджує, що кожен біологічний вид цілеспрямовано розвивається і змінюється для того, щоб якнайкраще пристосуватися до навколишнього середовища. В процесі еволюції багато видів комах і риб придбали захисне забарвлення, їжак став невразливим завдяки голкам, людина стала володарем складної нервової системи. Можна сказати, що

еволюція - це процес оптимізації всіх живих організмів. Розглянемо, якими ж засобами природа вирішує цю задачу оптимізації [7, 12,18, 31].

Основний механізм еволюції - це природний відбір. Його суть полягає в тому, що більш пристосовані особини мають більше можливостей для виживання і розмноження і, отже, приносять більше потомства, чим погано пристосовані особини. При цьому завдяки передачі генетичній інформації (генетичному спадкоємству) нащадки успадковують від батьків основні їх якості. Таким чином, нащадки сильних індивідуумів також будуть відносно добре пристосованими, а їх частка в загальній масі особин зростатиме. Після зміни декількох десятків або сотень поколінь середня пристосованість особин даного вигляду помітно зростає.

Щоб зробити зрозумілими принципи роботи генетичних алгоритмів, пояснимо також, як влаштовані механізми генетичного спадкоємства в природі. У кожній клітці будь-якої тварини міститься вся генетична інформація цієї особини. Ця інформація записана у вигляді набору дуже довгих молекул ДНК (Дезоксирибонуклеїнова Кислота). Кожна молекула ДНК - це ланцюжок, що складається з молекул нуклеотидів чотирьох типів, що позначаються A, T, C і G. Власне, інформацію несе порядок проходження нуклеотидів в ДНК. Таким чином, генетичний код індивідуума - це просто дуже довгий рядок символів, де використовуються всього 4 букви. У тваринній клітці кожна молекула ДНК оточена оболонкою – таке утворення називається **хромосомою.**

Кожна природжена якість особини (колір очей, спадкові хвороби, тип волосся і так далі) кодується певною частиною хромосоми, яка називається **геном** цієї властивості. Наприклад, ген кольору очей містить інформацію, що кодує певний колір очей. Різні значення гена називаються його **алелями.**

При розмноженні тварин відбувається злиття двох батьківських статевих кліток і їх ДНК взаємодіють, утворюючи ДНК нащадка. Основний спосіб взаємодії - **кросовер** (cross-over, схрещування). При кросовері ДНК предків діляться на дві частини, а потім обмінюються своїми половинками.

При спадкоємстві можливі мутації із-за радіоактивності або інших впливів, в результаті яких можуть змінитися деякі гени в статевих клітинах одного з батьків. Змінені гени передаються нащадкові і наділяють його новими властивостями. Якщо ці нові властивості корисні, вони, швидше за все, збережуться в даному виді - при цьому відбудеться стрибкоподібне підвищення пристосованості виду.

Як вже було відмічено, еволюція - це процес постійної оптимізації біологічних видів. Знаючи, як вирішується задача оптимізації видів в природі, застосуємо схожий метод для вирішення різних реальних задач.

Хай дана деяка складна функція (цільова функція), залежна від декількох змінних, і потрібно знайти такі значення змінних, при яких значення функції максимальне. Один з найбільш наглядних прикладів - задача розподілу інвестицій. У цієї задачі змінними є обсяги інвестицій в кожен проект (наприклад, 10 змінних), а функцією, яку потрібно максимізувати, - сумарний дохід інвестора. Також задані значення мінімального і максимального обсягу вкладення в кожен з проектів, які задають область зміни кожній із змінних. Спробуємо вирішити цю задачу, застосовуючи відомі нам природні способи оптимізації. Розглядатимемо кожен варіант інвестування (набір значень змінних) як індивідуума, а прибутковість цього варіанту - як пристосованість цього індивідуума. Тоді в процесі еволюції пристосованість індивідуумів зростатиме, а значить, з'являтимуться все більш і більш прибуткові варіанти інвестування. Зупинивши еволюцію в деякий момент і вибравши самого кращого індивідуума, ми отримаємо досить хороше рішення задачі.

Генетичний алгоритм (ГА) - це проста модель еволюції в природі, реалізована у вигляді комп'ютерної програми. Він широко використовується, для проектування структури мостів, для пошуку максимального відношення міцність/вага, для визначення найменш марнотратного розміщення при нарізці форм з тканини. ГА можуть також використовуватися для інтерактивного управління процесом, наприклад на хімічному заводі, або балансуванні завантаження на багатопроцесорному комп'ютері. Наприклад, ізраїльська

компанія Schema розробила програмний продукт Channeling для оптимізації роботи стільникового зв'язку шляхом вибору оптимальної частоти, на якій вестиметься розмова. У основі цього програмного продукту використовуються генетичні алгоритми.

У генетичному алгоритмі використовується як аналог механізму генетичного спадкоємства, так і аналог природного відбору. При цьому зберігається біологічна термінологія в спрощеному вигляді. От як моделюється генетичне спадкоємство:

Хромосома	Вектор (послідовність) з нулів і одиниць. Кожна позиція (біт) називається геном.
Індивідуум = генетичний код	Набор хромосом = варіант рішення задачі.
Кросовер	Операція, при якій дві хромосоми обмінюються своїми частинами.
Мутація	Випадкова зміна однієї або декількох позицій в хромосомі.

Щоб змоделювати еволюційний процес, згенеруємо спочатку випадкову популяцію - декілька індивідуумів з випадковим набором хромосом (числових векторів). Генетичний алгоритм імітує еволюцію цієї популяції як циклічний процес схрещування індивідуумів і зміни поколінь (рис. 5.4).

Життєвий цикл популяції - це декілька випадкових схрещувань (за допомогою кросовера) і мутацій, в результаті яких до популяції додається якась кількість нових індивідуумів. Відбір в генетичному алгоритмі - це процес формування нової популяції із старої, після чого стара популяція гине. Після відбору до нової популяції знову застосовуються операції кросовера і мутації, потім знову відбувається відбір, і так далі.

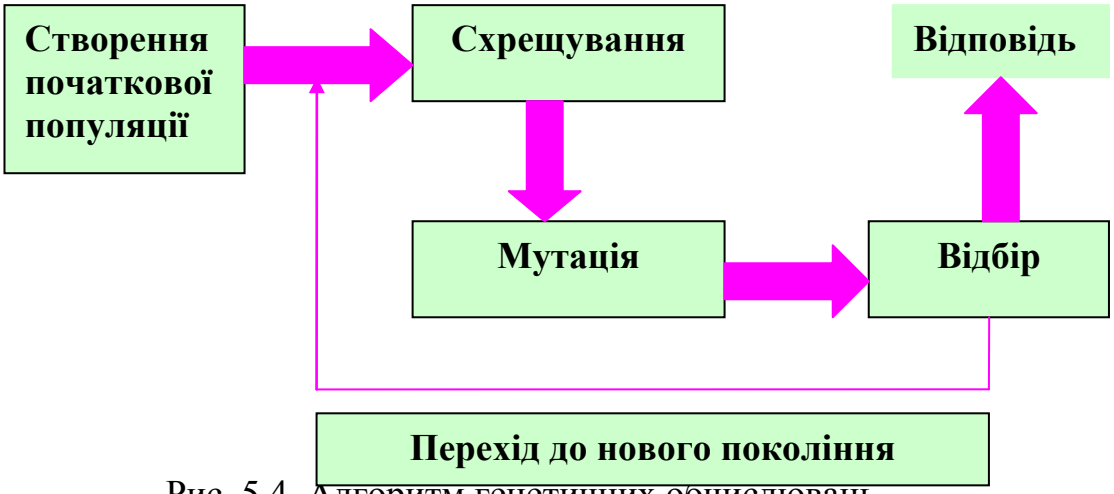


Рис. 5.4. Алгоритм генетичних обчислювань.

Відбір в генетичному алгоритмі тісно пов'язаний з принципами природного відбору в природі таким чином:

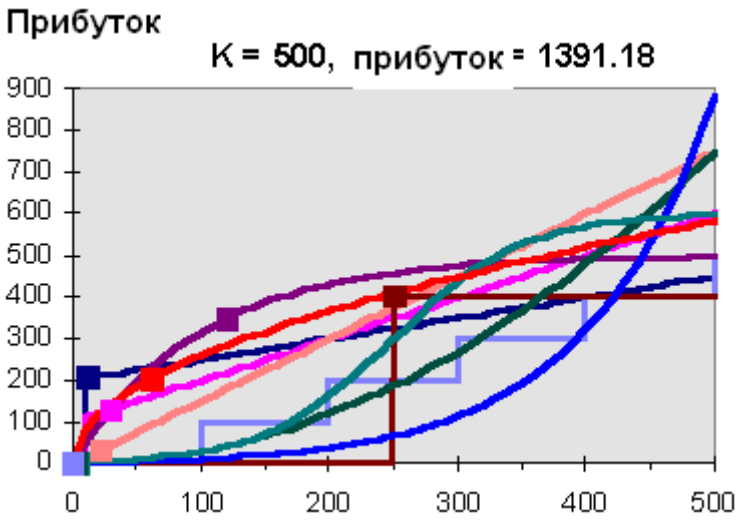
Пристосованість індивідуума	Значення цільової функції на цьому індивідуумі
Виживання найбільш пристосованих	Популяція наступного покоління формується відповідно до цільової функції. Чим пристосованіше індивідуум, тим більше вірогідність його участі в кросовері, тобто розмноженні.

Таким чином, модель відбору визначає, яким чином слід будувати популяцію наступного покоління. Як правило, вірогідність участі індивідуума в схрещуванні береться пропорційній його пристосованості. Часто використовується так звана стратегія елітизма, при якій декілька кращих індивідуумів переходять в наступне покоління без змін, не беручи участь в кросовері і відборі. У будь-якому випадку кожне наступне покоління буде в середньому краще попереднього. Коли пристосованість індивідуумів перестає помітно збільшуватися, процес зупиняють і як рішення задачі оптимізації беруть якнайкращого із знайдених індивідуумів.

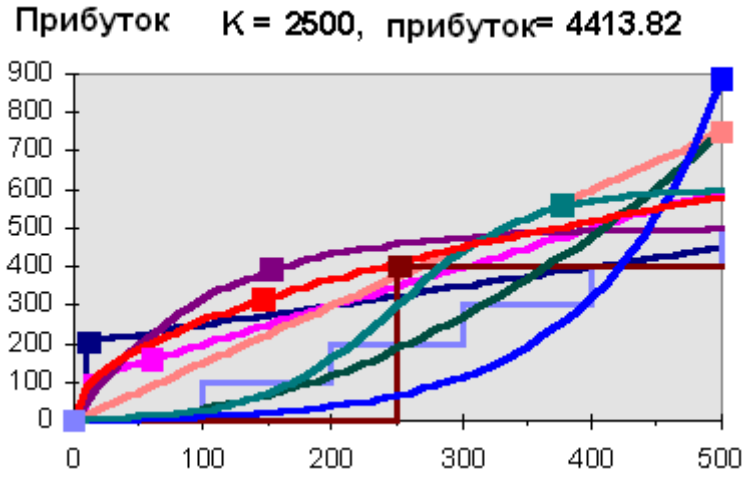
Повертаючись до задачі оптимального розподілу інвестицій, пояснимо особливості реалізації генетичного алгоритму в цьому випадку.

- Індивідуум = варіант рішення задачі = набір з 10 хромосом X_j.
- Хромосома X_j = об'єм вкладення в проект j = 16-розрядний запис цього числа
- Оскільки обсяги вкладень обмежені, не всі значення хромосом є допустимими. Це враховується при генерації популяцій.
- Оскільки сумарний обсяг інвестицій фіксований, то реально варіюються лише 9 хромосом, а значення 10-ої визначається по ним однозначно.

Нижче приведені результати роботи генетичного алгоритму для трьох різних значень сумарного обсягу інвестицій K (рис. 5.5). З рисунка 5.5. (а) видно, що при малому значенні K інвестуються тільки ті проекти, які прибуткові при мінімальних вкладеннях. Якщо збільшити сумарний обсяг інвестицій, стає прибутковим вкладати гроші і в дорожчі проекти.



а) капіталовкладення в проект;



б) капіталовкладення в проект.

Рис. 5.5. Розрахунок обсягу інвестицій на основі генетичного алгоритму.

При подальшому збільшенні K досягається поріг максимального вкладення в прибуткові проекти, і інвестування в малоприбуткові проекти знову набуває сенсу (рис. 5.5. б)Природа приголомшує своєю складність і багатством всіх своїх проявів. Серед прикладів можна назвати складні соціально-економічні системи, імунні і нейроні системи, складні взаємозв'язки між видами. Вони - всього лише деякі з чудес, які сталі очевидніші, коли вчені стали глибшими досліджувати світ довкола нас. Багато що з того, що ми бачимо і спостерігаємо, можна пояснити єдиною теорією: теорією еволюції через спадковість, мінливість і відбір.

Перша публікація, яку можна віднести до генетичних алгоритмів з'явилася в 1957 році, автором якої був Н.А. Барічеллі (N.A. Barricelli), і називалася вона «Symbiogenetic Evolution Processes realised by artificial methods». У 1962 році з'явилася ще одна його робота «Numerical testing of evolution theories». Приблизно в той же час ще один дослідник А.С.Фрейзер (A.S.Fraser) також опублікував дві статті: «Simulation of genetic systems by automatic digital computers: S-linkage, dominance, and epistasis» (1960) і «Simulation of genetic systems» (1962). Не дивлячись на те, що роботи обох авторів були направлені перш за все на розуміння природного феномену

спадковості, робота Фрейзера має багато спільного з сьогоднішнім баченням генетичних алгоритмів. Він моделював еволюцію 15-бітових рядків і підраховував процентний зміст особин з вдалим фенотипом в успішних поколіннях. Його роботи нагадують оптимізацію функцій, проте в роботах Фрейзера немає жодної згадки про можливість використовувати ГА для штучних задач.

У багатьох джерелах саме Холланда називають батьком сучасної теорії генетичних алгоритмів. Проте, Холланд займався ними не із самого початку. Його цікавила, перш за все, здібність природних систем до адаптації, а його мрією було створення такої системи, яка могла б пристосуватися до будь-яких умов довкілля. Заслуга Холланда в тому, що він усвідомив значення еволюційних принципів в адаптації і розвинув свої припущення. Разом зі своїми студентами, що слухали курси по адаптивних системах в університеті штату Мічіган, він розробляє те, що згодом назве «генетичним планом», а нам це відомо як «генетичний алгоритм». Про те, наскільки серйозно велися роботи в цьому напрямі говорить вже те, що один із студентів Холланда - Кенет Де Йонг (Kenneth De Jong) захистив дисертацію «An Analysis of the Behavior of a Class of Genetic Adaptive Systems» на ступінь доктора філософії в 1975 році. У тому ж році виходить знаменита книга Холланда «Адаптація в природних і штучних системах». Після цього ГА починають привертати до себе все більше уваги, ними займається все більше дослідників, які знаходять їм нові сфери вживання.

В даний час генетичні алгоритми в різних формах застосовуються до багатьох наукових і технічних проблем, зокрема для інтелектуального аналізу даних. Слід зазначити, що Data Mining не основна сфера застосування генетичних алгоритмів. Їх потрібно розглядати швидше як потужний засіб вирішення всіляких комбінаторних задач і задач оптимізації. Проте генетичні алгоритми увійшли зараз до стандартного інструментарію методів Data Mining.

Можливо найбільш популярним додатком генетичних алгоритмів є оптимізація багатопараметричних функцій. Багато реальних задач можуть бути

сформульовані як пошук оптимального значення, де значення - складна функція, залежна від деяких вхідних параметрів. В деяких випадках, представляє інтерес знайти ті значення параметрів, при яких досягається найкраще точне значення функції. У інших випадках, точний оптимум не потрібний - рішенням може вважатися будь-яке значення, яке краще за деяку задану величину. В цьому випадку, генетичні алгоритми - часто найбільш прийнятний метод для пошуку «хороших» значень. Сила генетичного алгоритму поміщена в його здатності маніпулювати одночасно багатьма параметрами, ця особливість ГА використовувалося в сотнях прикладних програм, включаючи проектування літаків, налаштування параметрів алгоритмів і пошуку стійких станів систем нелінійних диференціальних рівнянь. Таким чином, перевагами генетичних алгоритмів є:

- не вимагання жодної інформації про поверхню відповіді;
- розриви, що існують на поверхні відповіді мають незначний ефект та не впливають на ефективність оптимізації;
- стійкість до попадання в локальний оптимум;
- добре працюють при вирішенні великомасштабних проблем оптимізації;
- Можуть бути використані для широкого класу задач, зокрема; з середовищем, що змінюється.

Недоліками генетичних алгоритмів слід вважати:

- складність роботи у разі, коли необхідно знайти точний глобальний оптимум;
- час виконання функції оцінки великий;
- конфігурація є не простою (кодування рішення).

Мета інтелектуального аналізу даних за допомогою ГА полягає в тому, аби знайти краще можливе рішення задачі по одному або декільком критеріям. Аби реалізувати генетичний алгоритм, потрібно спочатку вибрати відповідну структуру для представлення цих рішень. У постановці задачі пошуку об'єкт цієї структури даних представляється точкою в просторі пошуку всіх можливих рішень. Властивості об'єктів представлені значеннями параметрів, що

об'єднуються в запис, названий в еволюційних методах **хромосомою.** У генетичних методах оперують хромосомами, що відносяться до множини об'єктів - **популяції**. Імітація генетичних принципів - **імовірнісний вибір** батьків, серед членів популяції, схрещування їх хромосом, відбір нащадків для включення в нові покоління об'єктів на основі оцінки цільової функції - веде до еволюційного поліпшення значень **цільової функції** (функції пристосованості) від покоління до покоління.

Найчастіше хромосома - це бітовий рядок. Проте ГА не обмежені бінарним представленням даних. Деякі реалізації використовують цілочисельне або вещественне кодування. Не дивлячись на те, що для багатьох реальних задач, мабуть, більше личать рядки змінної довжини, в даний час структури фіксованої довжини найбільш поширені і вивчені. Спочатку і ми обмежимося лише структурами, які є одиночними рядками по n біт.

Кожна **хромосома** (рядок) є конкатенацією ряду підкомпонентів, названих генами. Гени розташовуються в різних позиціях або локусах хромосоми і набувають значень, званих алелями. У представленнях з бінарними рядками **ген** - біт, **локус** - його позиція в рядку і алель - його значення (0 або 1). Біологічний термін «**генотип**» відноситься до повної генетичної моделі особини і відповідає структурі в ГА. Термін «**фенотип**» відноситься до зовнішніх спостережуваних ознак і відповідає вектору в просторі параметрів. Надзвичайно простий, але ілюстративний приклад - задача максимізації наступної функції двох змінних

$$f(x_1, x_2) = x_1 x_2,\ 0 \le x_1 \le 1,\ 0 \le x_2 \le 1.$$

Зазвичай методика кодування реальних змінних x_1 і x_2 полягає в їх перетворенні в двійкові цілочисельні рядки достатньої довжини - достатньою для того, щоб забезпечити бажану точність. Передбачимо, що 10-розрядне кодування достатнє і для x_1, і x_2. Встановити відповідність між генотипом і фенотипом закодованих особин можна розділивши відповідне двійковому представленню ціле число на значення $2^{10} - 1$. Наприклад, код [0000000000] відповідає вещественному значенню 0/1023 або 0, тоді як код [1111111111]

відповідає 1023/1023 або 1. Структура даних, що оптимізується - 20-бітовий рядок, що представляє конкатенацію кодувань x_1 і x_2. Змінна x_1 розміщується в крайніх лівих 10-розрядах, тоді як x_2 розміщується в правій частині генотипу особини (20-бітовому рядку). Генотип - точка в 20-вимірному бінарному просторі, досліджуваному ГА. Фенотип - точка в двовимірному просторі параметрів.

Після того, як вибрані параметри, їх число і розрядність, необхідно вирішити, як безпосередньо записувати дані. Можна використовувати звичайне кодування, коли, наприклад, $<1011>_2=<11>_{10}$, або **коди Грея**, коли $<1011>_G=<1110>_2=<14>_{10}$. Не дивлячись на те, що коди Грея потребують неминучого кодування/декодування даних, вони дозволяють уникнути деяких проблем, які з'являються в результаті звичайного кодування. Можна лише сказати, що перевага кода Грея в тому, що якщо два числа є послідовними при кодуванні, то і їх двійкові коди розрізняються лише на один розряд, а в двійкових кодах це не так. Варто відзначити, що кодувати і декодувати в коди Грея можна так:

- спочатку копіюється самий старший розряд, потім:
- з двійкового кода в код Грея $G[i] = XOR(B[i+1], B[i])$;
- з кода Грея в двійковий код $B[i] = XOR(B[i+1], G[i])$.

Тут, $G[i]$ - і-й розряд кода Грея, а $B[i]$ - і-й розряд бінарного кода. Наприклад, послідовність чисел від 0 до 7 в двійковому коді {000, 001, 010, 011, 100, 101, 110, 111}, а в кодах Грея {000, 001, 011, 010, 110, 111, 101, 100}.

Загальна схема такого алгоритму може бути записана таким чином.

1. Формування початкової популяції.
2. Оцінка особин популяції.
3. Відбір (селекція).
4. Схрещування.
5. Мутація.
6. Формування нової популяції.

7. Якщо популяція не зійшлася, то 2. Інакше - останов.

Зупинимося детальніше на всіх етапах цього алгоритму.

Формування початкової популяції. Стандартний генетичний алгоритм починає свою роботу з формування початкової популяції I_0 - кінцевого набору допустимих рішень задачі. Ці рішення можуть бути вибрані випадковим чином або отримані за допомогою простих наближених алгоритмів. Вибір початкової популяції не має значення для збіжності процесу в асимптотиці, проте формування «хорошої» початкової популяції (наприклад, з множини локального оптимуму) може помітно скоротити час досягнення глобального оптимуму. Якщо відсутні припущення про місце розташування глобального оптимуму, то індивіди з початкової популяції бажано розподілити рівномірно по всьому простору пошуку рішення.

Оцінка особин популяції. Аби оптимізувати яку-небудь структуру з використанням ГА, потрібно задати міру якості для кожного індивіда в просторі пошуку. Для цієї мети використовується **функція пристосованості**. У задачах максимізації цільова функція часто сама виступає як функція пристосованості (наприклад, як в розглянутому раніше двовимірному прикладі); для задач мінімізації цільову функцію слід інвертувати. Якщо вирішувана задача має обмеження, виконання яких неможливо контролювати алгоритмічно, то функція пристосованості, як правило, включає також штрафи за невиконання обмежень (вони зменшують її значення).

Відбір (селекція). Існує декілька підходів до вибору батьківської пари. Найбільш поширеними операторами вибору батьків є наступні.

Панміксія - найпростіший оператор відбору. Відповідно до його кожному членові популяції зіставляється випадкове ціле число на відрізку [1; n], де n - кількість особин в популяції. Розглядатимемо ці числа як номери особин, які візьмуть участь в схрещуванні. При такому виборі якісь з членів популяції не братимуть участь в процесі розмноження, оскільки утворюють пару самі з собою. Якісь члени популяції візьмуть участь в процесі відтворення не однократно з різними особинами популяції. Не дивлячись на простоту, такий

похід універсальний для вирішення різних класів задач. Проте він досить критичний до чисельності популяції, оскільки ефективність алгоритму, що реалізовує такий підхід, знижується із зростанням чисельності популяції.

Інбридинг є таким методом, коли перший з батьків вибирається випадковим чином, а другим з родини є член популяції найближчий до першого. Тут «найближчий» може розумітися, наприклад, в сенсі мінімальної відстані Хеммінга (для бінарних рядків) або евклідова відстані між двома вещественними векторами. Відстань Хеммінга дорівнює числу локусов (розрядів), що розрізняються, в бінарному рядку. При *аутбридинге* також використовують поняття схожості особин. Проте тепер шлюбні пари формують з максимально далеких особин.

Останні два способи по різному впливають на поведінку генетичного алгоритму. Так, інбридинг можна охарактеризувати властивістю концентрації пошуку в локальних вузлах, що фактично приводить до розбиття популяції на окремі локальні групи довкола підозрілих на екстремум ділянок ландшафту. Аутбридинг же направлений на попередження збіжності алгоритму до вже знайдених рішень, заставляючи алгоритм переглядати нові, недосліджені області. Інбридинг і аутбридинг буває генотипом (коли в якості відстані береться різниця значень цільової функції для відповідних особин) і фенотипним (в якості відстані береться відстань Хеммінга).

Селекція полягає в тому, що батьками можуть стати лише ті особини, значення пристосованості яких не менше порогової величини, наприклад, середнього значення пристосованості по популяції. Такий похід забезпечує швидшу збіжність алгоритму. Проте із-за швидкої збіжності селективний вибір батьківської пари не підходить тоді, коли ставиться задача визначення декількох екстремумів, оскільки для таких задач алгоритм, як правило, швидко сходиться до одного з рішень. Крім того, для деяких багатовимірних задач із складним ландшафтом цільової функції швидка збіжність може перетворитися на передчасну збіжність до квазіоптимального рішення. Цей недолік може бути частково компенсований використанням відповідного механізму відбору, який

би «гальмував» дуже швидку збіжність алгоритму. Порогова величина в селекції може бути обчислена різними способами. Тому виділяють різні варіації селекції. Серед операторів селекції найбільш поширеними є два імовірнісні оператори **пропорційної (рулеточної)** і **турнірної** селекції. В деяких випадках також застосовується **відбір усіканням.**

Пропорційний відбір (Proportional selection). При пропорційній селекції вірогідність на k-му кроці є вибір рішення i як одного з батьків, яке задається формулою $P\{i-вибрано\}=\dfrac{f(i)}{\sum_{j\in I_k} f(j)}$, у припущенні, що $f(i)>0$ для всіх $i\in I_k$ (I_k - популяція на k-му кроці).

Простий пропорційний відбір – *рулетка (roulette-wheel selection)* - відбирає особини за допомогою n «запусків» рулетки. Колесо рулетки містить по одному сектору для кожного i-го члена популяції. Розмір i-го сектора пропорційний відповідній величині $P(i)$. При такому відборі члени популяції з вищою пристосованістю з більшою вірогідністю частіше вибиратимуться, чим особини з низькою пристосованістю.

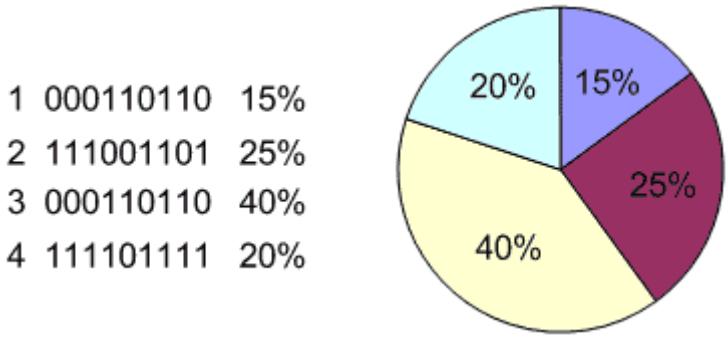


Рис. 5.6. Оператор селекції типу колеса рулетки.

Турнірний відбір (Tournament selection). Турнірний відбір може бути описаний таким чином: з популяції, що містить m рядків (особин), вибирається випадковим чином t рядків і кращий рядок записується в проміжний масив (між вибраними рядками проводиться турнір). Ця операція повторюється m разів. Рядки в отриманому проміжному масиві потім використовуються для схрещування. Розмір групи рядків, що відбираються для турніру, часто рівний

2. В цьому випадку говорять про двоїчним/парний турнір. Взагалі ж t називається чисельністю турніру (рис. 5.7).

Відбір усіканням (Truncation selection). Ця стратегія використовує відсортовану по убуванню популяцію. Число особин для схрещування вибирається відповідно до порогу $T \in [0;1]$. Поріг визначає, яка доля особин, починаючи з найпершої (самої пристосованої), братиме участь у відборі. В принципі, поріг можна задати і рівним 1, тоді всі особини поточної популяції будуть допущені до відбору. Серед особин, допущених до схрещування випадковим чином $m/2$ разів вибираються батьківські пари, нащадки яких утворюють нову популяцію.

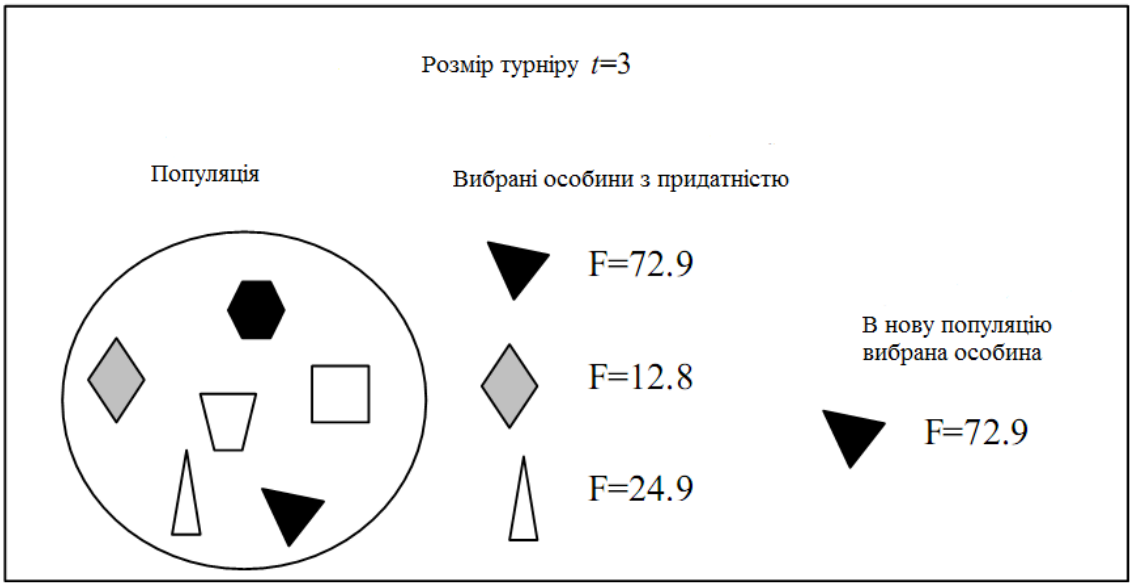


Рис.5.7. Турнірний відбір.

Схрещування. Оператори рекомбінації (схрещування) застосовують відразу ж після оператора відбору батьків для здобуття нових особин-нащадків. Сенс рекомбінації полягає в тому, що створені нащадки повинні успадковувати генну інформацію від обох батьків. Розрізняють **дискретну рекомбінацію** і **кросинговер.**

Дискретна рекомбінація (Discrete recombination) в основному застосовується до хромосом з вещественними генами. Основними способами

дискретної рекомбінації є власне дискретна рекомбінація, проміжна, лінійна і розширена лінійна рекомбінації.

Дискретна рекомбінація відповідає обміну генами між особинами. Для ілюстрації даного оператора порівняємо дві особини з трьома генами

Особина 1	12	25	7
Особина 2	116	4	34

Для створення двох нащадків з рівною імовірністю випадково виберемо номер особини для кожного гена

Схема 1	2	2	1
Схема 2	1	2	1

Метод рулетки. Сумарна придатність = 200, сумарна імовірність = 1.

Популяція з 5 особин	Придатність	Імовірність вибору
C_1	52	52/200 = 0.26
C_2	85	85/200 = 0.425
C_3	37	37/200 = 0.185
C_4	3	3/200 = 0.015
C_5	23	23/200 = 0.115

Згідно схемі створимо нащадків

Нащадок 1	116	4	7
Нащадок 2	12	4	7

Дискретна рекомбінація може застосовуватися для будь-якого типа генів (двоїчних, вещественних і символьних).

Проміжна рекомбінація (Intermediate recombination) застосовується лише до вещественних змінних, але не до бінарних. У даному методі заздалегідь визначається числовий інтервал значень генів нащадків, який повинен містити значення генів батьків. Нащадки створюються за наступним правилом: Нащадок = Батько 1 + α(Батько 2 — Батько 2), де множник α - випадкове число на відрізку $[-d,1+d]$, $d>0$. Як відзначають прибічники цього методу, найбільш оптимальне відтворення виходить при $d=0.25$. Для кожного гена створюваного нащадка вибирається окремий множник α. Розглянемо вживання оператора на прикладі. Хай двоє батьків мають наступні значення генів

Особина 1	12	25	7
Особина 2	116	4	34

Випадково виберемо значення $\alpha \in [-0.25;1.25]$ для кожного гена обох нащадків

Схема 1	0.5	1.1	-0.1
Схема 2	0.1	0.8	0.5

Обчислимо значення генів нащадків за запропонованою вище формулою

Нащадок 1	12+0.5(116–12) = 64	25+1.1(4-25) = 1.9	4.3
Нащадок 2	12+0.1(116-12) = 2.4	25+0.8(4-25) = 8.2	20.5

При проміжній рекомбінації виникають значення генів, відмінні від значення генів особин-батьків. Це приводить до виникнення нових особин, придатність яких може бути краще, ніж придатність батьків. У літературі такий оператор рекомбінації інколи називається *диференціальним схрещуванням*.

Лінійна рекомбінація (Line recombination) відрізняється від проміжної тим, що множник α вибирається для кожного нащадка один раз. Розглянемо гени приведених вище батьків. Хай значення α визначається таким чином

Схема 1	0.5
Схема 2	0.1

Тоді гени створених нащадків набудуть наступних значень

Нащадок 1	12+0.5(116–12) = 64	25+0.5(4-25) = 14.5	20.5
Нащадок 2	12+0.1(116-12) = 2.4	25+0.1(4-25) = 22.9	9.7

Якщо розглядати особині популяції як точки в k-мірному просторі, де k - кількість генів в одній особині, то можна сказати, що при лінійній рекомбінації точки нащадків, що генеруються, лежать на прямій, заданій двома точками - батьками.

Кросинговер (Crossover). Рекомбінацію бінарних рядків прийнято називати кросинговером або схрещуванням. Як тільки два рішення-батька вибрано, до них застосовується імовірнісний оператор схрещування, який будує на їх основі нові (1 або 2) рішення-нащадка. Відібрані особини піддаються кросинговеру із заданою вірогідністю P_c. Якщо кожна пара батьків породжує двох нащадків, для відтворення популяції необхідно схрестити $m/2$ пари. Для кожної пари з вірогідністю P_c застосовується кросинговер. Відповідно, з вірогідністю $1-P_c$ кросинговер не відбувається і тоді незмінені особини переходять на наступну стадію (мутації). Існує велика кількість різновидів оператора схрещування.

Одноточковий кросинговер (Single-point crossover) працює таким чином. Спочатку випадковим чином вибирається одна з можливих точок розриву. Точка розриву це ділянка між сусідніми бітами в рядку. Обоє батьківські

структури розриваються на два сегменти по цій точці. Потім відповідні сегменти різних батьків склеюються і виходять два генотипи нащадків (рис. 5.8)).

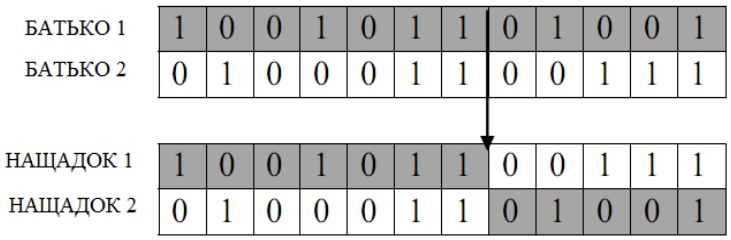
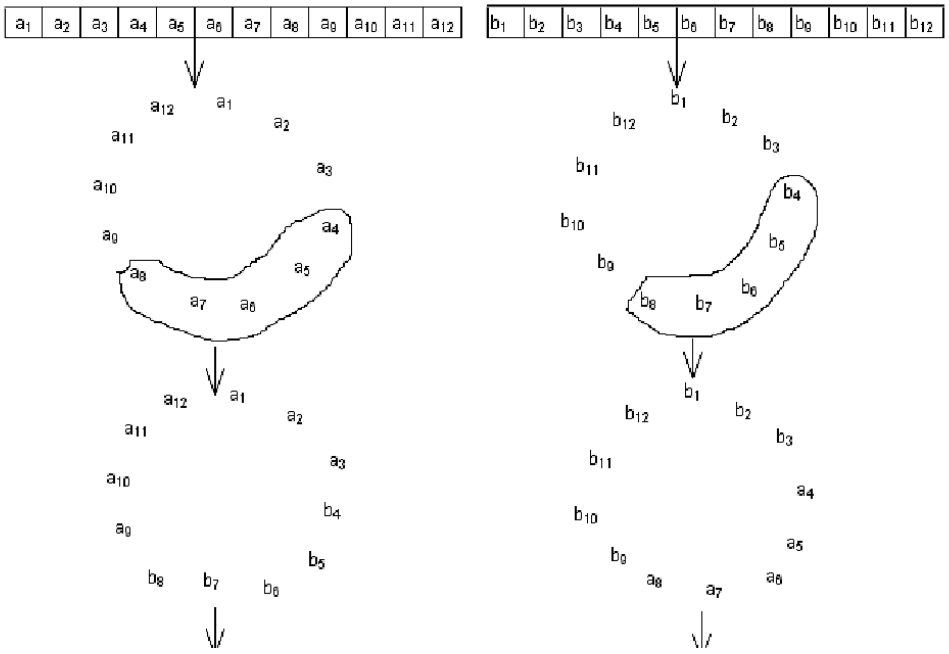


Рис. 5.8. Приклад роботи одноточкового кросинговера.

У двоточковому кросинговері хромосоми розглядаються як цикли, які формуються з'єднанням кінців лінійної хромосоми разом. Для заміни сегменту одного циклу сегментом іншого циклу потрібний вибір двох точок розрізу. В такому представленні, одноточковий кросинговер може бути розглянутий як кросинговер з двома точками, але з однією точкою розрізу, зафіксованою на початку рядка. Отже, двоточковий кросинговер вирішує ту ж саму задачу, що і одноточковий, але більш повно. Хромосома, що розглядається як цикл, може містити більшу кількість стандартних блоків, оскільки вони можуть зробити «циклічне повернення» в кінці рядка (рис. 5.9). Зараз багато дослідників погоджуються, що двоточковий кросинговер взагалі краще, ніж одноточковий.



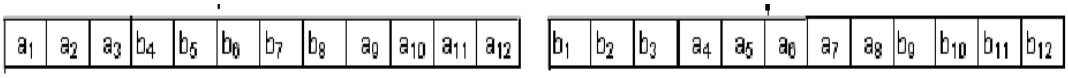


Рис. 5.9. Двоточковий кросинговер.

Для *багатоточкового кросинговера (Multi-point crossover)*, вибираємо m точок розрізу $k_i \in \{1,2,...,N_{\text{var}}\}$, $i=\overline{1,m}$, N_{var} - кількість змінних (генів) в особини. Точки розрізу вибираються випадково без повторень і сортуються в порядку зростання. При кросинговері відбувається обмін ділянками хромосом, обмеженими точками розрізу і таким чином отримують двох нащадків. Ділянка особини з першим геном до першої точки розрізу в обміні не бере участь. Порівняємо наступні дві особини по 11 двійковим генам.

Особина 1	0	1	1	1	0	0	1	1	0	1	0
Особина 2	1	0	1	0	1	1	0	0	1	0	1

Виберемо точки розрізу кросинговера

Точка розрізу ($m=3$)	2	6	10

Створимо двох нових нащадків

Нащадок 1	0	1	1	0	1	1	1	1	0	1	1
Нащадок 2	1	0	1	1	0	0	0	0	1	0	0

Вживання багатоточкового кросинговера вимагає введення декілька змінних (точок розрізу), і для відтворення вибираються особини з найбільшою пристосованістю.

Однорідний кросинговер (Uniform crossover) створює маску (схему) особини, в кожному локусі якою знаходиться потенційна точка кросинговера. Маска кросинговера має таку же довжину, що і особини, які схрещуються.

Створити маску можна таким чином: введемо деяку величину $0 < p_0 < 1$, і якщо випадкове число більше p_0, то на n позицію маски ставимо 0, інакше - 1. Таким чином, гени маски є випадковими двійковими числами (0 або 1). Згідно цим значенням, геном нащадка стає перша (якщо ген маски = 0) або друга (якщо ген маски = 1) особина-батько. Наприклад, розглянемо особині

Особина 1	0	1	1	1	0	0	1	1	0	1	0
Особина 2	1	0	1	0	1	1	0	0	1	0	1

Для кожного створюваного нащадка створимо маску з 11 випадково вибраних елементів з множини {0; 1}

Маска 1	0	1	1	0	0	0	1	1	0	1	0
Маска 2	1	0	0	1	1	1	0	0	1	0	1

Створимо нащадків за наступним правилом: якщо на i-му місці у відповідній масці стоїть 1, то ген 1 батька переходить нащадкові, інакше успадковується ген другого батька. Отримаємо наступні особини

Нащадок 1	1	1	0	1	1	1	1	1	1	1	1
Нащадок 2	0	0	1	1	0	0	0	0	0	0	0

Однорідний кросинговер дуже схожий на багатоточковий, але рядок випадкових бітових значень в ньому довше. Це гарантує, що в нащадках чергуватимуться короткі рядки особин-батьків.

Тріадний кросинговер (Triadic crossover). Даний різновид кросинговера відрізняється від однорідного тим, що після відбору пари батьків з останніх членів популяції випадковим чином вибирається особина, яка надалі використовується як маска. Далі 10 % генів маски мутують. Потім гени першого батька порівнюються з генами маски: якщо гени однакові, то вони

передаються першому нащадкові, інакше на відповідні позиції хромосоми нащадка переходять гени другого батька. Генотип другого нащадка відрізняється від генотипу першого тим, що на тих позиціях, де у першого нащадка стоять гени першого батька, у другого нащадка стоять гени другого батька і навпаки.

Як було показано вище, кросинговер генерує нове рішення (у вигляді особини-нащадка) на основі двох інших, комбінуючи їх частини. Тому число різних рішень, які можуть бути отримані кросинговером при використанні однієї і тієї ж пари готових рішень, обмежене. Відповідно, простір, який ГА може покрити, використовуючи лише кросинговер, жорстко залежить від генофонду популяції. Чим різніше генотипи рішень популяції, тим більше простір покриття. При виявленні локального оптимуму відповідний йому генотип прагнутиме зайняти всі позиції в популяції, і алгоритм може зійтися до помилкового оптимуму. Тому в генетичному алгоритмі важливу роль грають мутації.

Мутація (mutation). Після того, як закінчиться стадія кросинговера, нащадки можуть піддаватися випадковим модифікаціям, названим **мутаціями** (рис. 5.10). У простому випадку в кожній хромосомі, яка піддається мутації, кожен біт з вірогідністю P_m змінюється на протилежний (це так звана *одноточкова мутація*).

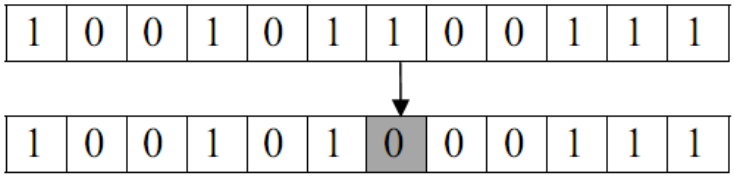


Рис. 5.10. Приклад дії мутації.

Складнішим різновидом мутації є оператори *інверсії* і *транслокації*. Інверсія - це перестановка генів в зворотному порядку усередині навдогад вибраної ділянки хромосоми (рис. 5.11).

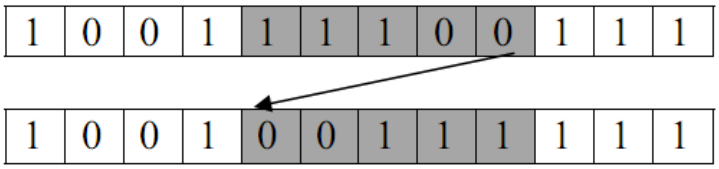


Рис. 5.11. Приклад дії інверсії.

Транслокація - це перенесення якої-небудь ділянки хромосоми, в інший сегмент цієї ж хромосоми (рис. 5.12).

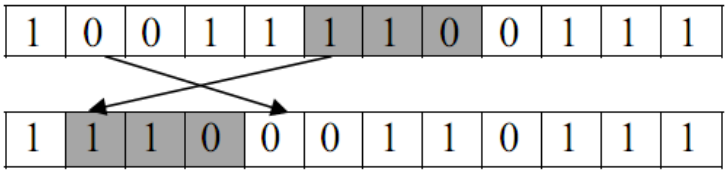


Рис. 5.12. Приклад дії транслокації.

Слід зазначити, що всі перераховані генетичні оператори (одноточковий і багатоточковий кросинговер, одноточкова мутація, інверсія транслокація) мають схожі біологічні аналоги.

У деяких роботах пропонується використовувати стратегію *інцеста* як механізму самоадаптації оператора мутації. Вона полягає в тому, що вірогідність мутації кожного гена P_m визначається для кожного нащадка на підставі генетичної близькості його батьків. Наприклад, це може бути відношення числа співпадаючих генів батьків до загального числа генів хромосоми. Це приводить до цікавого ефекту - при високій різноманітності генофонду популяції наслідки мутації будуть мінімальними, що дозволяє операторові схрещування працювати без стороннього втручання. У разі ж пониження різноманітності, що виникає в основному при застряванні алгоритму в локальному оптимумі, наслідки мутації стають відчутнішими, а при повному сходженні популяції алгоритм просто стає стахостичним, що збільшує вірогідність виходу популяції з локального оптимуму.

Інколи (з метою підвищення середньої пристосованості популяції) допустимо здійснювати *направлені мутації*, тобто після кожної зміни

хромосоми перевіряти, чи підвищилася в результаті цієї мутації її пристосованість і, якщо немає, повертати хромосому до вихідного стану.

Формування нового покоління. Після схрещування і мутації особин необхідно вирішити проблему про те, які з нових особин увійдуть до наступного покоління, а які - ні, і що робити з їх предками. Є два найбільш поширених способи.

1. Нові особини (нащадки) займають місця своїх батьків. Після чого настає наступний етап, в якому нащадки оцінюються, відбираються, дають потомство і поступаються місцем своїм «дітям».
2. Наступна популяція включає як батьків, так і їх нащадків.

У другому випадку необхідно додатково визначити, які з особин батьків і нащадків попадуть в нове покоління. У простому випадку, в нього після кожного схрещування включаються дві кращі особини з четвірки батьків і їх нащадків. Ефективнішим є механізм *витіснення*, який реалізується таким чином, що прагне видаляти «схожі» хромосоми з популяції і залишати ті, що відрізняються.

Елітарний відбір (Elite selection). Створюється проміжна популяція, яка включає як батьків, так і їх нащадків. Члени цієї популяції оцінюються, а потім з них вибираються N найкращих, які і увійдуть до наступного покоління. Часто даний метод комбінують з іншими - вибирають в нову популяцію, наприклад, 10% «елітних» особин, а останні 90% - одним з традиційних методів селекції. Інколи ці 90% особини створюють випадково, як при створенні початкової популяції перед запуском роботи генетичного алгоритму. Використання стратегії елітизма виявляється вельми корисним для ефективності ГА, оскільки не допускає втрату кращих рішень. Наприклад, якщо популяція зійшлася в локальному максимумі, а мутація вивела один з рядків в область глобального, то при попередній стратеги досить імовірно, що ця особина в результаті схрещування буде втрачена, і рішення задачі не буде отримано. Якщо ж використовується елітизм, то отримане хороше рішення залишатиметься в популяції до тих пір, поки не буде знайдено ще краще.

Відбір витісненням (Exclusion selection). У даному відборі вибір особини в нову популяцію залежить не лише від величини її придатності, але і від того, чи є вже у формованій популяції особина з аналогічним хромосомним набором. Відбір проводиться з числа батьків і їх нащадків. Зі всіх особин з однаковою пристосованістю перевага спочатку віддається особинам з різними генотипами. Таким чином, досягаються дві мета: по-перше, не втрачаються кращі знайдені рішення, що володіють різними хромосомними наборами, по-друге, в популяції постійно підтримується генетична різноманітність. Витіснення в даному випадку формує нову популяцію швидше з видалених особин, замість особин, що групуються біля поточного знайденого рішення. Даний метод найбільш придатний для багатоекстремальних задач, при цьому окрім визначення глобальних екстремумів з'являється можливість виділити і ті локальні максимуми, значення яких близькі до глобальних.

Метод Больцмана, або метод відпалу (Bolzman selection). У даному методі вірогідність відбору в нову популяцію залежить від управляючого параметра - температури T. Зазвичай імовірність попадання в нову популяцію обчислюється за наступною формулою

$$p = \frac{1}{1 + e^{\frac{f(i) - f(j)}{T}}},$$

де $f(i)$ та $f(j)$ - значення цільової функції i та j особин, відповідно. Номери особин i та j вибираються випадково. Якщо значення p виявиться більше випадкового числа на інтервалі (0; 1), то в нову популяцію попаде особина $f(i)$, інакше $f(j)$.

В деяких випадках застосовується альтернативна формула

$$p = \frac{\exp(f(i)/T)}{\{\exp(f(j)/T)\}}$$

де {} - середнє по популяції на ітерації з номером t. Якщо p виявиться більше випадкового числа на інтервалі (0; 1), то особина $f(i)$ попаде в нову популяцію.

У даному методі першим поколінням відповідають високі температури, і вірогідність відбору особин велика (підтримується різноманіття у новій популяції). Поступово із зростанням кількості поколінь ГА температура знижується, імовірність відбору зменшується і в нову популяцію потрапляють ті особини, пристосованість яких мінімальна.

Останов алгоритму. Робота ГА є ітераційним процесом, який продовжується до тих пір, поки не пройде задане число поколінь або не виконається який-небудь інший критерій останову. У оптимізаційних задачах традиційними критеріями останову алгоритму є, наприклад, тривала відсутність прогресу в сенсі поліпшення значення середньої (або кращої) пристосованості популяції, мала різниця між кращим і гіршим значенням пристосованості для поточної популяції і тому подібне

5.3. Моделі генетичних алгоритмів.

Канонічний ГА (Canonical GA - J. Holland).

Ця модель алгоритму є класичною. Вона була запропонована Джоном Холландом в його знаменитій роботі «Адаптація в природних і штучних середовищах» [62]. Часто можна зустріти опис простого ГА (Simple GA), він відрізняється від канонічного тим, що використовує або рулеточний, або турнірний відбір. Модель канонічного ГА має наступні характеристики.

- Фіксований розмір популяції.
- Фіксована розрядність генів.
- Пропорційний відбір.
- Одноточечний кросовер і одноточечна мутація.
- Наступне покоління формується з нащадків поточного покоління без «елітизму».

Алгоритм роботи ГА (*репродуктивний план Холланда*) має наступний вигляд.

1. Ініціалізація початкової популяції. Покласти номер епохи $t = 0$. Ініціалізувати випадковим чином m генотипів особин і сформувати з них випадкову популяцію. Обчислити пристосованість особин популяції $F(0) = (f_1(0),..., f_m(0))$, а потім - середню пристосованість по популяції $f_{ср}(0) = \sum_{i=1}^{m} f_i(0)/m$.

2. Вибір батьків для схрещування. Збільшити номер епохи на одиницю: $t = t + 1$. Визначити випадковим чином номер першого батька $l \in \{1,...,m\}$, призначивши імовірність випадання будь-якого номера h пропорційній величині $f_h(t)/f_{ср}(t)$. Повторним випробуванням визначити номер другого батька k.

3. Формування генотипу нащадків. Із заданою ймовірністю p_c провести над генотипами вибраних батьків одноточковий кросовер. Далі до кожного з отриманих нащадків з імовірністю p_M застосувати оператора мутації.

4. Оновлення популяції. Помістити нащадків в популяцію, заздалегідь видаливши з неї батьків. Обчислити пристосованості нащадків і відновити значення середньої пристосованості популяції $f_{ср}(t)$. Якщо формування популяції не завершене, перейти до кроку 2.

Генітор (Genitor).

У моделі генітор використовується специфічний спосіб відбору [31]. Спочатку, як і завжди, популяція ініціалізується, і її особини оцінюються. Потім вибираються випадковим чином дві особини, схрещуються, причому виходить лише один нащадок, який оцінюється і займає місце менш пристосованої особини в популяції (а не одного з батьків). Після цього знову випадковим чином вибираються дві особини, і їх нащадок займає місце батьківської особини з найнижчою пристосованістю. Таким чином, на кожному кроці в популяції оновлюється лише одна особина. Процес продовжується до тих пір, поки придатності хромосом не стануть однаковими. У даний алгоритм можна додати мутацію нащадка після його створення. Критерій закінчення

процесу, як і вигляд кросинговера і мутації, можна вибирати різними способами. Підводячи підсумки, можна виділити наступні характерні особливості.

- Фіксований розмір популяції.
- Фіксована розрядність генів.
- Особини для схрещування вибираються випадковим чином.
- Обмежень на типа кросовера і мутації немає.
- В результаті схрещування особин виходить один нащадок, який займає місце найменш пристосованої особини.

Метод переривистої рівноваги.

Даний метод заснований на палеонтологічній теорії переривистої рівноваги, яка описує швидку еволюцію за рахунок вулканічних і інших змін земної кори [18]. Для вживання даного методу в задачах інтелектуального аналізу даних пропонується після кожної генерації проміжного покоління випадковим чином перемішувати особині в популяції, а потім застосовувати основний ГА. У даній моделі для відбору батьківських пар використовується панміксія. Нащадки, що вийшли в результаті кросинговера, і найбільш придатні батьки випадковим чином змішуються. Із загальної маси в нове покоління попадуть лише ті особини, придатність яких вище середньою. Тим самим досягається управління розміром популяції залежно від наявності кращих особин. Така модифікація методу переривистої рівноваги може дозволити скоротити неперспективні популяції і розширити популяції, в яких знаходяться кращі індивідуальності.

Гібридний алгоритм (Hybrid Algorithm - L. «Dave» Davis).

Ідея гібридних алгоритмів полягає в поєднанні генетичного алгоритму з деяким іншим класичним методом пошуку, відповідним до даної задачі [34]. У кожному поколінні всі згенеровані нащадки оптимізуються вибраним методом і потім заносяться в нову популяцію. Тим самим виходить, що кожна особина в популяції досягає локального оптимуму, поблизу якого вона знаходиться. Далі виробляються звичайні для ГА дії: відбір батьківських пар, кросинговер і

мутації. На практиці гібридні алгоритми виявляються дуже вдалими. Це пов'язано з тим, що вірогідність попадання однієї з особин в область глобального максимуму досить велика. Після оптимізації така особина буде рішенням задачі. Відомо, що генетичний алгоритм здатний швидко знайти у всій зоні пошуку хороші рішення, але він може зазнавати труднощі в здобутті з них найкращих. Звичайний оптимізаційний метод може швидко досягти локального максимуму, але не може знайти глобальний. Поєднання двох алгоритмів дозволяє використовувати переваги обох.

CHC (Eshelman).

CHC розшифровується як Cross-population selection, Heterogeneous recombination and Cataclysmic mutation [7]. Даний алгоритм досить швидко сходиться через те, що в ньому немає мутацій, наступних за оператором кросинговера, використовуються популяції невеликого розміру, і відбір особин в наступне покоління ведеться і між батьківськими особинами, і між їх нащадками. У даному методі для кросинговера вибирається випадкова пара, але не допускається, аби між батьками була маленька хеммінгова відстань або мала відстань між крайніми бітами. Для схрещування використовується різновид однорідного кросовера HUX (Half Uniform Crossover), при якому нащадкові переходить рівно половина бітів кожного батька. Для нового покоління вибираються N кращих різних особин серед батьків і дітей. При цьому дублювання рядків не допускається. У моделі CHC розмір популяції відносно малий - близько 50 особин. Це виправдовує використання однорідного кросинговера і дозволяє алгоритму зійтися до рішення. Для здобуття більш менш однакових рядків CHC застосовує cataclysmic mutation: всі рядки, окрім самого пристосованого, піддаються сильній мутації (змінюється біля третини бітів). Таким чином, алгоритм перезапускається і далі продовжує роботу, застосовуючи лише кросинговер.

ГА з нефіксованим розміром популяції (Genetic Algorithm with Varying Population Size - GAVaPS).

У генетичному алгоритмі з нефіксованим розміром популяції кожній особини приписується максимальний вік, тобто число поколінь, після яких особина гине [9]. Впровадження в алгоритм нового параметра - віку - дозволяє виключити оператор відбору в нову популяцію. Вік кожної особини індивідуальний і залежить від її пристосованості. У кожному поколінні t на етапі відтворення звичайним способом створюється додаткова популяція з нащадків. Розмір додаткової популяції (AuxPopsize(t)) пропорційний розміру основної популяції (Popsize(t)) і рівний AuxPopsize(t) = [Popsize(t) p_c], p_c - вірогідність відтворення. Для відтворення особині вибираються з основної популяції з однаковою імовірністю незалежно від їх пристосованості. Після вживання мутації і кросинговера нащадкам приписується вік згідно значенню їх пристосованості. Вік є константою впродовж всієї еволюції особини (від народження до загибелі). Потім з основної популяції видаляються ті особини, термін життя яких витік, і додаються нащадки з проміжної популяції. Таким чином, розмір після однієї ітерації алгоритму обчислюється за формулою Popsize(t+1) = Popsize(t)+AuxPopsize(t)-D(t), де D(t) - число особин, які вмирають в поколінні t.

Простий паралельний ГА (Parallel implementations).

Генетичні алгоритми застосовуються і при паралельних обчисленнях [47] При цьому формується декілька популяцій, що живуть окремо. На етапі формування нового покоління за деяким правилом відбираються особини з різних популяцій. Згенерована таким чином популяція, в деяких випадках замінює всі інші. Таким чином, так або інакше, відбувається міграція особин однієї популяції в інші популяції. Тому паралельні ГА називають *міграціями*.

Спершу розглянемо створення простого паралельного ГА з класичної моделі Холланда. Для цього використовуватимемо турнірний відбір. Заведемо $N/2$ процесів (тут і далі процес розглядається як деяка машина, процесор, якої може працювати незалежно). Кожен з них вибиратиме випадково з популяції 4 особини, проводити 2 турніри, і переможців схрещувати. Отримані нащадки

записуватимуться в нове покоління. Таким чином, за один цикл роботи одного процесу змінюється ціле покоління.

Міграція (Migration).

Модель міграції представляє популяцію як множину підпопуляцій. Кожна підпопуляція обробляється окремим процесором. Ці підпопуляції розвиваються незалежно одна від одної протягом однакової кількості поколінь T (час ізоляції). Після закінчення часу ізоляції відбувається обмін особинами між підпопуляціями (міграція). Кількість особин, що піддалися обміну (вірогідність міграції), метод відбору особин для міграції і схема міграції визначає частоту виникнення генетичного різноманіття в підпопуляціях і обмін інформацією між популяціями. Відбір особин для міграції може відбуватися таким чином:

- випадкова одноманітна вибірка з числа особин;
- пропорційний відбір: для міграції беруться найбільш придатні особини.

Окремі підпопуляції в паралельних ГА можна умовно прийняти за вершини деякого графа. У зв'язку з цим можна розглядати топологію графа міграційного ГА. Найбільш поширеною топологією міграції є повний граф (рис. 5.13), при якій особині з будь-якої підпопуляцій можуть мігрувати в будь-яку іншу підпопуляцію. Для кожної підпопуляції повна кількість потенційних іммігрантів будується на основі всіх підпопуляцій. Мігруюча особина випадковим чином вибирається з цього загального числа.

При використанні в необмеженій міграції пропорційного відбору спочатку формується масив з найбільш придатних особин, відібраних по всіх підпопуляціях. Випадковим чином з цього масиву вибирається особина, і нею замінюють найменш придатну особину в підпопуляції 1. Аналогічні дії проробляємо з останніми підпопуляціями. Можливо, що якась популяція отримає дублікат своєї «хорошої» особини.

Інша основна міграційна схема - це топологія кільця (рис. 5.14). Тут особини передаються між сусідніми (по напряму обходу) підпопуляціями.

Таким чином, особини з однієї підпопуляції можуть мігрувати лише в одну - сусідню підпопуляцію.

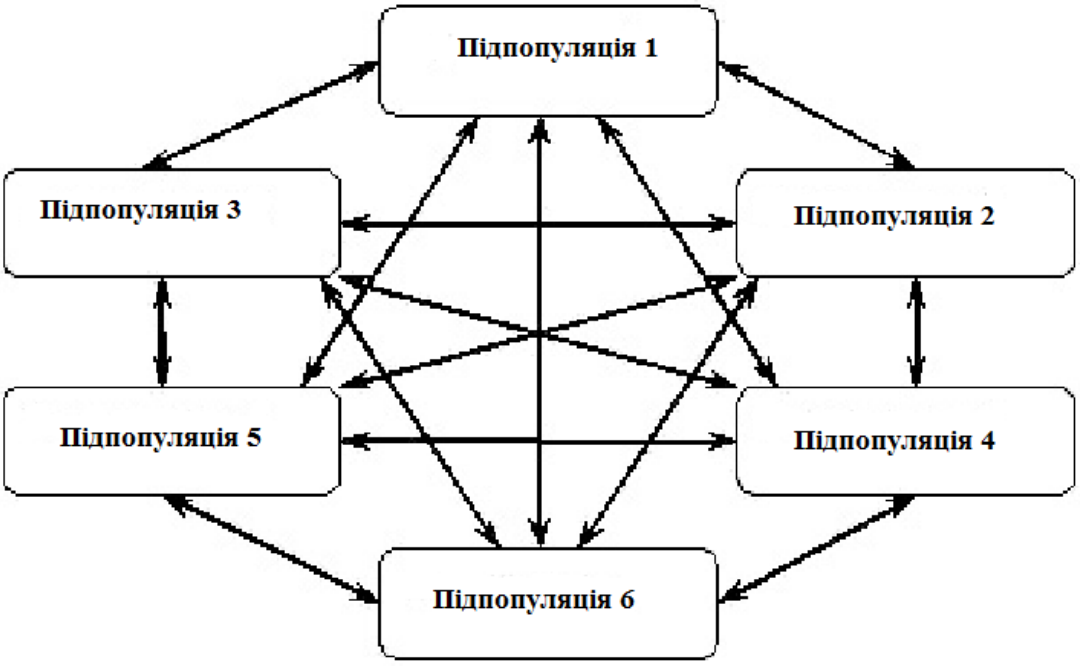


Рис. 5.13. Міграція з топологією повної мережі.

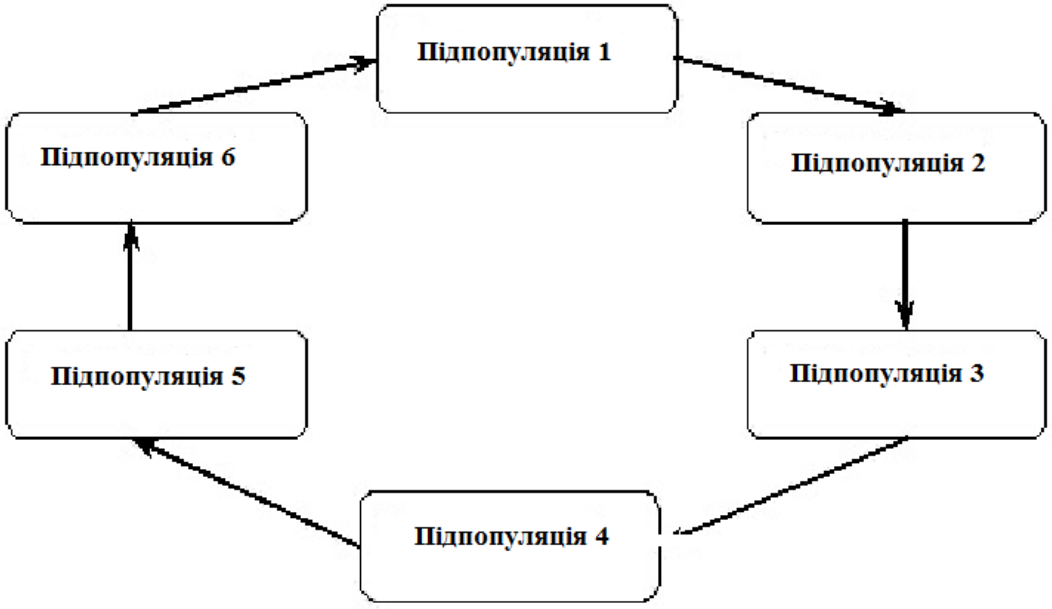


Рис. 5.14. Міграція з топологією кільця.

Модель дифузії, або острівна модель ГА (Island Model GA).

Острівна модель є найбільш поширеною моделлю паралельного ГА. Її суть полягає в тому, що популяція, яка як правило складається з дуже великого

числа особин, розбивається на однакові за розміром підпопуляції. Кожна підпопуляція обробляється окремим процесором за допомогою одного з різновидів непаралельного ГА. Зрідка, наприклад, через п'ять поколінь, підпопуляції обмінюватимуться декількома особинами. Такі міграції дозволяють підпопуляціям спільно використовувати генетичний матеріал.

Хай виконуються 16 незалежних генетичних алгоритмів, використовуючи підпопуляції з 1000 особин на кожному процесорі. Якщо міграцій немає, то відбувається 16 незалежних пошуків рішення. Всі пошуки ведуться на різних початкових популяціях і сходяться до певних особин. Дослідження підтверджують, що генетичний дрейф схильний приводити підпопуляції до різних домінуючих особин. Це пов'язано з тим, що, по-перше, кількість островів, що приймають домінуючих «емігрантів» з острова, обмежена (2 - 5 островів). По-друге, обмін особинами однобічний. Тому у великій популяції з'являться групи островів з різними домінуючими особинами. Якщо популяція невеликого розміру, то можлива швидка міграція помилкових домінуючих особин. Наприклад, дійсне рішення знаходиться лише на одному острові, а декілька помилкових домінант - на інших островах. Тоді при міграції кількість помилкових особин на островах зросте (на кожен острів міграції відбуваються з не менш 2 островів), генетичним алгоритмом вірне рішення буде зруйновано. Тим самим в маленькій популяції при генетичному дрейфі можлива поява помилкових домінуючих особин і сходження алгоритму до помилкового оптимуму.

Введення міграцій в острівній моделі дозволяє знаходити різні особини-домінанти в підпопуляціях, що сприяє підтримці різноманіття в популяції. Кожну підпопуляції можна прийняти за острів. Під час міграції підпопуляції обмінюються своїм домінуючим генетичним матеріалом. При частій міграції великої кількості особин відбувається перемішування генетичного матеріалу. Тим самим усуваються локальні відмінності між островами. Дуже рідкі міграції не дозволяють запобігти передчасній збіжності алгоритму в маленьких підпопуляціях. У даній моделі з кожного острова міграції можуть відбуватися

лише на певну відстань: 2 - 5 островів залежно від кількості підпопуляцій. Таким чином, кожен острів виявляється майже ізольованим. Кількість островів, на які можуть мігрувати особини однієї підпопуляції, називають відстанню ізоляції. Слід зауважити, що взаємоміграції виключені (рис. 5.15).

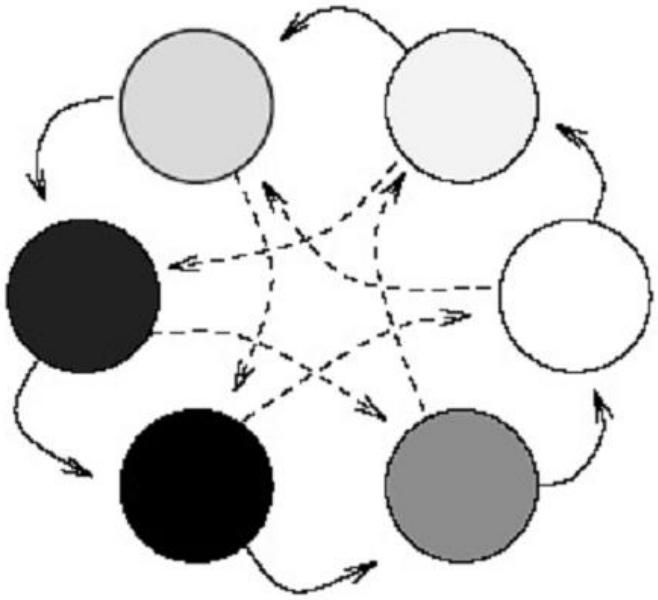


Рис. 5.15. Острівна модель.

Шима (Schema).

Хоча зовні здається, що ГА обробляє рядки, насправді при цьому неявно відбувається обробка шим, які представляють шаблони подібності між рядками. ГА практично не може займатися повним перебором всіх представлень в просторі пошуку. Проте він може виробляти вибірку значного числа гіперплощин в зонах пошуку з високою пристосованістю. Кожна така гіперплощина відповідає множині схожих рядків з високою пристосованістю. Шима - це рядок довжини l (як і довжина будь-якого рядка популяції), що складається із знаків алфавіту {0; 1; *}, де {*} - невизначений символ. Кожна шима визначає множину всіх бінарних рядків довжини l, що мають у відповідних позиціях або 0, або 1, залежно від того, який біт знаходиться у відповідній позиції самої шими. Шима, що не містить жодного невизначеного символу, є деяким рядком. Шима з одним невизначеним символом описує два бінарні рядки, а з двома — чотири рядки. Наприклад, шима, 10 * * 1, визначає

собою множину з чотирьох п'ятибітових рядків {10001; 10011; 10101; 10111}. Неважко відмітити, що шима з r невизначеними символами описує 2^r бінарних рядків. З іншого боку, кожен рядок довжини m описується 2^m шимами. Отже, в популяції з n таких рядків число можливих шим може досягати $n2^m$! При цьому велика частина шим ймовірно буде менш пристосованою чим інші, що може привести до епістазу. Тому рекомендується створювати початкову популяцію з шим з високою пристосованістю. Всі шими різні між собою. Основними характеристиками шин є порядок і довжина [7, 31].

Порядок шими $o(S)$ (order) - це число фіксованих бітів (0 або 1) в шимі S.

Визначальна довжина $\delta(S)$ (defining length) - це відстань між першим і останнім фіксованими бітами в шимі S. Довжина шими визначає концентрацію інформації в шимі. Вважається, що шима з однією фіксованою позицією має нульову довжину. Наприклад, шима $S = (***001*110)$ має порядок $o(S) = 6$ і довжину $\delta(S) = 10 - 4 = 6$. Порядок і довжина шим використовуються для визначення вірогідності мутації і кросинговера відповідно.

У зв'язку з тим, що більш пристосовані особини (хромосоми) описуються шимою з більшою пристосованістю, сенс роботи ГА полягає в пошуку двійкового рядка певного вигляду зі всієї множини бінарних рядків довжини m. Тоді простір пошуку складає 2^m рядків, а його розмірність рівна m. Шима відповідає деякій гіперплощині в цьому просторі. Дане ствердження можна проілюструвати таким чином. Хай розрядність хромосоми дорівнює 3, тоді всього можна закодувати $2^3 = 8$ рядків. Представимо куб в тривимірному просторі. Позначимо вершини цього куба трьохрозрядними бінарними рядками так, щоб мітки сусідніх вершин відрізнялися рівно на один розряд, причому вершина з міткою «000» знаходилася б на початку координат (рис. 5.16). Якщо узяти шиму вигляду «**0», то вона опише ліву грань куба, а шима «*10» - верхнє ребро цієї грані. Вочевидь, що шима «***» відповідає всьому простору.

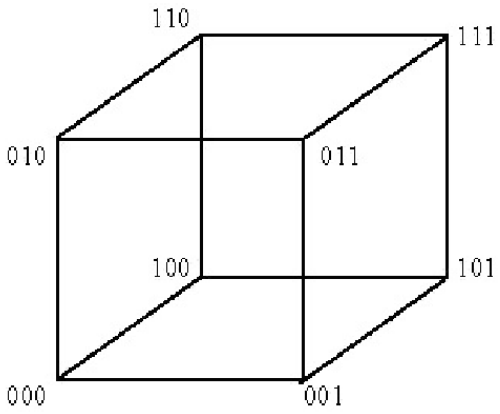


Рис. 5.16. Тривимірний куб.

Тут шимі «*1**» відповідає гіперплощина, що включає задні грані зовнішнього і внутрішнього куба, а шимі «**10» — гіперплощина з верхніми ребрами лівих граней обох кубів. Таким чином терміни «гіперплощина» і «шима» взаємозамінні.

Будівельні блоки - це шими, що володіють:

- високою придатністю;
- низьким порядком;
- короткою певною довжиною.

Придатність шими визначається як середнє придатності рядків-особин, які її містять. Після процедури відбору залишаються лише рядки з вищою придатністю. Отже, рядки, які є прикладами шим з високою придатністю, вибираються частіше. Кросинговер рідше руйнує шими з коротшою певною довжиною, а мутація рідше руйнує шими з низьким порядком. Тому, такі шими мають більше шансів переходити з покоління в покоління. Холланд показав, що в той час, як ГА явним чином обробляє n рядків на кожному поколінні, неявно обробляються порядка n^3 таких коротких шим низького порядку і з високою пристосованістю (корисних шим - useful schemata). Він називав це явище *неявним паралелізмом* (implicit parallelism). Для рішення реальних задач, присутність неявного паралелізму означає, що велика популяція має більше можливостей локалізувати рішення експоненціально швидше за популяцію з меншим числом особин.

Теорема шим (The schema theorem).

Теорема шим показує, яким чином простий ГА експоненціально збільшує число прикладів корисних шим або будівельних блоків, що приводить до знаходження рішення вихідної задачі.

Хай $m(H,t)$ - число прикладів шими H в t поколінні. Обчислимо очікуване число прикладів H в наступному поколінні або $m(H,t+1)$ в термінах $m(H,t)$. Простий ГА кожному рядку при відборі ставить у відповідність імовірність її «виживання» пропорційно її пристосованості (наприклад, як в методі рулетки). Очікується, що шима H може бути вибрана $m(H,t)(f(H)/f_{ср})$ разів, де $f_{ср}$ - середня придатність популяції, а $f(H)$ - середня придатність тих рядків в популяції, які являються прикладами H. Ймовірність того, що одноточечний кросинговер зруйнує шиму дорівнює ймовірності того, що точка розриву попаде між певними бітами. Ймовірність же того, що H «переживає» кросинговер не менше $1-p_c(\delta(H)/l-1)$, де p_c - ймовірність кросинговера. Ця ймовірність - нерівність, оскільки шима зможе вижити, якщо в кросинговері також брав участь приклад подібної шими.

Ймовірність того, що H переживе точкову мутацію - $(1-p_m)^{o(H)}$, де p_m - ймовірність мутації. Цей вираз можна апроксимувати як $(1-o(H))$ для малих p_m і $o(H)$. Множення очікуваного число відборів і ймовірності виживання відоме як *теорема шим*

$$\langle m(H,t+1)\rangle \geq m(H,t)\frac{f(H,t)}{\overline{f(t)}}[1-p_c\frac{\delta(H)}{l-1}](1-p_m)^{o(H)}.$$

Теорема шим показує, що будівельні блоки зростають по експоненті, у той же час шими з пристосованістю нижче середньою розпадаються з тією ж швидкістю. Голдберг в своїх дослідженнях теореми шим висуває гіпотезу будівельних блоків, яка полягає в тому, що «будівельні блоки об'єднуються, аби сформувати кращі рядки». Тобто рекомбінація і експоненціальне зростання будівельних блоків веде до формування кращих будівельних блоків.

Тоді як теорема шим передбачає зростання прикладів хороших шим, сама теорема вельми спрощено описує поведінку ГА. Перш за все, $f(H)$ і $f_{ср}$ не залишаються постійними від покоління до покоління. По-друге, теорема шим пояснює втрати шим, але не появу нових. Нові шими часто створюються кросинговером і мутацією. Крім того, в результаті еволюції члени популяції стають все більш і більш схожими один на одного так, що зруйновані шими будуть відразу ж відновлені. Нарешті, доведення теореми шим побудоване на елементах теорії вірогідності і, отже, не враховує розкид значень. У багатьох задачах розкид значень придатності шими може бути досить великий, роблячи процес формування шим дуже складним.

Істотна різниця придатності шими може привести до збіжності та неоптимального рішення. Не дивлячись на простоту, теорема шим описує декілька важливих аспектів поведінки ГА. Мутації з більшою вірогідністю руйнують шими високого порядку, тоді як кросинговер з більшою вірогідність руйнує шими з більшою певною довжиною. Коли відбувається відбір, популяція сходиться пропорційно відношенню пристосованості кращої особини, до середньої пристосованості в популяції: це відношення - *міра тиску відбору («selection pressure»)*. Збільшення p_c чи p_m, або зменшення тиску відбору веде до збільшеного здійснення вибірки або дослідження простору пошуку, але не дозволяє використовувати все хороші шими, які має в своєму розпорядженні ГА. Зменшення p_c, чи p_m, або збільшення тиску вибору веде до поліпшення використання знайдених шим, але гальмує дослідження простору у пошуках нових хороших шим. Моделювання ГА передбачає збереження рівноваги ГА між тим і іншим, що зазвичай відоме як проблема «балансу дослідження і використання».

Деякі дослідники критикують зазвичай швидку збіжність ГА, заявляючи, що випробування величезних кількостей шим, що перекриваються, вимагає більшої вибірки і повільнішої, більш керованій збіжності. Методологія управління збіжністю простого ГА до цих пір не вироблена.

Недоліками теореми шим є те, що вона:

- застосовується лише до канонічного ГА;
- не враховує ту обставину, що кросинговер і мутація можуть не лише руйнувати шиму, але створювати її з інших шим. Тому в теоремі шим присутній знак нерівності;
- дозволяє розрахувати долю шим в популяції лише для наступного покоління, тобто при спробі підрахувати число рядків, відповідних даній шимі, через декілька поколінь з використанням теореми шим до успіху не приведе.

Неперервні генетичні алгоритми.

При роботі з оптимізаційними задачами в неперервних просторах цілком природно представляти гени безпосередньо дійсними числами. В цьому випадку хромосома є вектор дійсних чисел. Довжина хромосоми збігатиметься з довжиною вектора-рішення оптимізаційної задачі, інакше кажучи, кожен ген відповідатиме за одну змінну. Генотип об'єкту стає ідентичним його фенотипу. Вищесказане визначає список основних переваг неперервних алгоритмів.

1. Використання неперервних генів робить можливим пошук у великих просторах (навіть у невідомих), що важко робити в разі двійкових генів, коли збільшення простору пошуку скорочує точність вирішення при незмінній довжині хромосоми.

2. Однією з важливих рис неперервних ГА є їх здібність до локального налаштування рішень.

3. Використання неперервних алгоритмів для представлення рішень зручно, оскільки близько до постановки більшості прикладних задач. Крім того, відсутність операцій кодування/декодування, які необхідні в класичному ГА, підвищує швидкість роботи алгоритму.

Як відомо, появу нових особин в популяції канонічного ГА забезпечують декілька біологічних операторів: відбір, схрещування і мутація. Як оператори відбору особин в батьківську пару тут підходять будь-які відомі : пропорційний, турнірний, відбір усіканням. Проте оператори схрещування і мутації не годяться: у класичних реалізаціях вони працюють з бітовими рядками. Потрібні власні реалізації, що зважають на специфіку неперервних

алгоритмів. Оператор схрещування неперервного ГА породжує одного або декількох нащадків від двох хромосом. Власне кажучи, потрібно з двох векторів дійсних чисел отримати нові вектори за якими-небудь законами. Більшість неперервних алгоритмів генерують нові вектори в околиці батьківських пар. Спершу розглянемо прості і популярні кросовери.

Хай $C_1 = (c_1^1, c_2^1, ..., c_n^1)$ та $C_2 = (c_1^2, c_2^2, ..., c_n^2)$ - дві хромосоми, вибрані оператором селекції для схрещування.

Плоский кросовер (flat crossover). Створюється нащадок $H = (h_1, ..., h_k, ..., h_n)$, де h_k - випадкове число з інтервалу $[c_k^{\min}, c_k^{\max}]$, $c_k^{\min} = \min(c_k^1, c_k^2)$, $c_k^{\max} = \max(c_k^1, c_k^2)$.

Простий кросовер (simple crossover). Випадковим чином вибирається число k з інтервалу $\{1,2,\ldots,n\text{-}1\}$ і генеруються два нащадки $H_1 = (c_1^1, c_2^1, ..., c_k^1, c_{k+1}^2, ..., c_n^2)$ та $H_2 = (c_2^2, c_2^2, ..., c_k^2, c_{k+1}^1, ..., c_n^1)$.

Арифметичний кросовер (arithmetical crossover). Створюються два нащадки $H_1 = (h_1^1, ..., h_n^1)$, $H_2 = (h_1^2, ..., h_n^2)$, де $h_k^1 = w^* c_k^1 + (1-w) * c_k^2$, $h_k^2 = w * c_k^2 + (1-w) * c_k^1$. Величина w або константа (рівномірний арифметичний кросовер) з інтервалу [0;1], або змінюється із збільшенням епох (нерівномірний арифметичний кросовер).

Геометричний кросовер (geometrical crossover). Створюються два нащадки $H_1 = (h_1^1, ..., h_n^1)$, $H_2 = (h_1^2, ..., h_n^2)$, де $h_k^1 = (c_k^1)^w (c_k^2)^{1-w}$, $h_k^2 = (c_k^2)^w (c_k^1)^{1-w}$. Величина w - випадкове число з інтервалу [0;1]. (Застосовується в тому випадку, якщо вектори C_1 і C_2 не негативні)

Дискретний кросовер (discrete crossover). Кожен ген h_k вибирається випадково по рівномірному закону з кінцевої множини $\{c_k^1, c_k^2\}$.

Евристичний кросовер (Wright's heuristic crossover). Хай C_1 - один з двох батьків з кращою пристосованістю. Тоді $h_k = w * \left| c_k^1 - c_k^2 \right| + c_k^1$, де w - випадкове число з інтервалу [0;1].

В якості оператора мутації найбільшого поширення набула *випадкова мутація*. При випадковій мутації ген, що підлягає зміні, набуває випадкового значення з інтервалу своєї зміни. Розглянуті кросовери історично були запропоновані першими, проте в багатьох задачах їх ефективність виявляється невисокою. Пізніше були розроблені покращені оператори схрещування, аналітична формула яких і ефективність обґрунтовані теоретично. Розглянемо детальніше один з таких кросоверів - SBX.

SBX кросовер (Simulated Binary Crossover). SBX – кросовер, імітуючий двійковий. Він був розроблений в 1995 році дослідницькою групою під керівництвом K. Deb'a. Як випливає з його назви, цей кросовер моделює принципи роботи двійкового оператора схрещування.

SBX кросовер був отриманий наступним способом. Автором було введено поняття *сили пошуку кросовера* (search power). Це кількісна величина, що характеризує розподіл вірогідності появи будь-якого нащадка від двох довільних батьків. Спочатку була розрахована сила пошуку для одноточкового двійкового кросовера, а потім був розроблений вещественний кросовер SBX з такою ж силою пошуку. У ньому сила пошуку характеризується розподілом вірогідності випадкової величини β

$$p(\beta) = \begin{cases} 0.5(n+1)\beta^n, \beta \le 1 \\ 0.5(n+1)\beta^{-(n+2)}, \beta > 1 \end{cases}.$$

Для генерації нащадків використовується наступний алгоритм, що використовує наступний вираз для $p(\beta)$. Створюються два нащадки $H_k = (h_1^k, ..., h_j^k, ..., h_n^k)$, $k = 1,2$, де $h_1^1 = 0.5[(1-\beta)c_1^1 + (1+\beta)c_1^2]$, $h_1^2 = 0.5[(1+\beta)c_1^1 + (1-\beta)c_1^2]$, β - число, отримане по формулі

$$\beta(u) = \begin{cases} (2u)^{\frac{1}{n+1}}, u(0,1) \le 0.5 \\ (\frac{1}{2(1-u)})^{\frac{1}{n+1}}, u(0,1) > 0.5 \end{cases}.$$

У формулі $u(0,1)$ - випадкове число, розподілене по рівномірному закону, $n \in [2,5]$ - параметр кросовера. На рисунку 5.17 приведена геометрична

інтерпретація роботи SBX кросовера при схрещуванні двох хромосом, відповідних дійсним числам 2 і 5. Видно, як параметр n впливає на кінцевий результат: збільшення n спричиняє за собою збільшення ймовірності появи нащадка в околиці батька і навпаки.

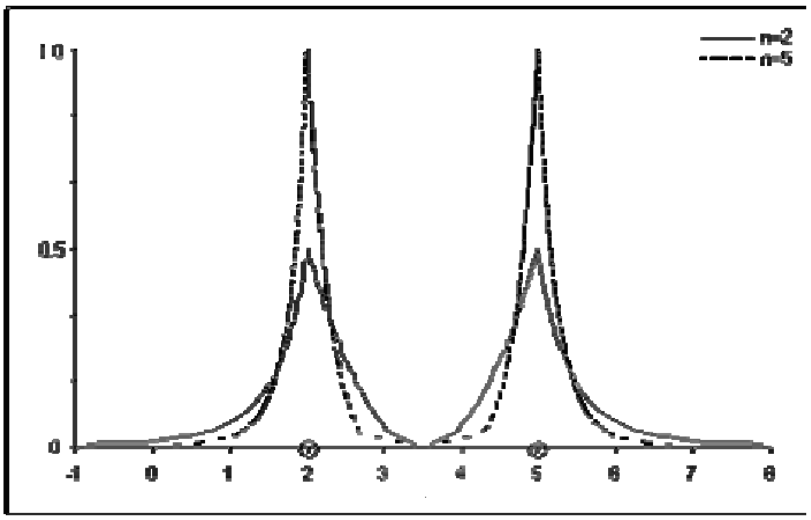


Рис. 5.17. Геометрична інтерпретація роботи SBX кросовера.

Експерименти автора SBX кросовера показали, що він у багатьох випадках ефективніше змішаного кросовера, хоча не існує жодного кросовера, ефективного у всіх випадках. Дослідження показують, що використання декількох операторів кросовера дозволяє зменшити ймовірність передчасної збіжності, тобто поліпшити ефективність алгоритму оптимізації в цілому. Для цього можуть використовуватися спеціальні стратегії, що змінюють ймовірність вживання окремого еволюційного оператора залежно від його «успішності», або використання гібридних кросоверів. В будь-якому разі, якщо стоїть задача оптимізації в безперервних просторах з використанням еволюційних стратегій, то слід зробити вибір на користь безперервного генетичного алгоритму.

5.4. Програмне забезпечення та сфери застосування генетичних алгоритмів.

Доля нових наукових напрямів - при всій їх революційності, незвичності і парадоксальності - зазвичай розвивається за канонами класичної мелодрами, повторюючи вічну казку про Попелюшку. Цей нехитрий сюжетний хід прослідкується в розвитку практично всіх нових наук - і теорії нейронних мереж, і нечіткої логіки, і системної динаміки, і інших. Всіх, окрім однієї. Цього не сталося з напрямом artificial life (або A-life, як охрестили нову науку американські журналісти), основу якого складають генетичні алгоритми [2, 25].

Отже, що таке artificial life? Найзагальніше визначення дає журнал MIDRANGE Systems, трактуючи A-life як «комп'ютерне моделювання живих об'єктів». Проте на практиці до A-life прийнято відносити комп'ютерні моделі, що володіють рядом конкретних особливостей. По-перше, центральна модель системи - будь то самохідний робот або інтелектуальний агент Internet - володіє здатністю адаптуватися до умов зовнішнього світу, поповнюючи знання про нього шляхом взаємодії з іншими об'єктами і середовищем. По-друге, компоненти системи, розвиваючись в процесі еволюції, здатні передавати свої характерні риси по спадку. Відповідно, присутній механізм породження нових поколінь - шляхом ділення, схрещування або дублювання існуючих об'єктів. По-третє, навколишній світ досить жорстокий і зводить до мінімуму шанси на виживання і появу потомства у слабких і погано пристосованих особин. І, нарешті, присутній механізм породження нових форм (аналог мутацій на реальному світі), що зазвичай містить елемент випадковості. Іншими словами, аби вирішити реальну задачу методами artificial life (наприклад, створити винищувач ідеальної форми, або розробити оптимальну стратегію біржової гри), треба побудувати динамічну модель середовища, в якому належить існувати проектованому об'єкту, населити її множиною різновидів цього об'єкту і дати їм «пожити» декілька поколінь. Слабкі особини відмиратимуть, сильні - схрещуватися, закріплюючи в нових поколіннях свої кращі риси. Через

декілька десятків (інколи - сотень і навіть тисяч) циклів така селекція породить «цивілізацію» практично невразливих особин, ідеально пристосованих до заданої моделі світу. Можна з упевненістю сказати, що «життєва сила» рішення, що пройшло жорстокий відбір, буде достатньою, аби успішно протистояти будь-яким діям конкурентів.

Багато представників великого бізнесу (у тому числі такі гіганти, як Ford) заявляють, що вже давно використовують елементи A-life в ситуаційному моделюванні і ділових іграх для керівників. Наприклад, моделюється ринок США, населений сонмом акул-конкурентів і працюють фірми, віддані в управління кожному з учасників. За один такт гри треба оцінити позиції своєї фірми на різних ринках декількох міст, закрити збиткові філії, реорганізувати неефективні і розширити прибуткові, тобто прийняти в цілому близько п'ятдесяти управлінських рішень. Через півгодини центральний комп'ютер підводить загальні підсумки такту (рівного кварталу роботи фірми) і переходив до наступного «покоління».

Промисловці застосовують методи A-life в першу чергу при створенні всіляких роботів, маніпуляторів і автоматизованих виробництв. Одне з визначень A-life навіть трактує її як теорію управління співтовариством роботів, що вирішують навігаційні завдання шляхом адаптації до зовнішніх умов. Розробники БІС серйозно захоплені розробкою нового покоління мікросхем, що реалізовують базові алгоритми A-life «в кремнії». Піонером тут виступає легендарний професор Mead (який створив п'ятнадцять років назад перший «кремнієвий компілятор», - повністю автоматичну САПР замовлених БІС). Він розробив модель асинхронного нейроподібного кристала, на якому збирається спершу реалізувати моделі ока і вуха, і вже отримав перший патент на однотранзисторну модель синапсу. А на конференції ECAL була продемонстрована перша програмована мікросхема (ППМ), здібна до самоудосконалення. У схему «зашитий» генетичний алгоритм, що оптимізує програмний код обробки зовнішніх дій.

Бізнесменам добре знайома фірма Flavors Technology, яка вирішує велику кількість складноформалізуємих фінансових і управлінських задач, використовуючи методи A-life і теорії хаосу, які реалізовані на спеціалізованій багатопроцесорній системі. Менш відомо, що фірма Thinking Machines, знаменитий розробник суперкомп'ютерів, також має власні програмні напрацювання по втіленню A-life для вирішення ряду оборонних завдань. Осібно коштує така екзотична область, як комп'ютерна вірусологія. Дослідники, що використовують методи A-life, підійшли до цієї проблеми, трактуючи комп'ютерні віруси як специфічний різновид штучного життя, здібний до мутацій, розмноження, інфікування місця існування і самоудосконалення. При такому трактуванні стає реальністю давня ідея про універсальний антивірус, який виявлятиме не конкретні види вірусів (як більшість існуючих програм) а прояви нових форм комп'ютерного життя - зміна дисципліни обробки переривань, нетипова поведінка відомих програм, незнайомий «почерк» спілкування програм з файловою системою і тому подібне Фірма IBM вже анонсувала перший комерційний антивірус, побудований на подібних принципах. Що ж до таких областей, як моделювання катастроф, надзвичайних ситуацій і військових конфліктів, то тут історія ситуаційних центрів, що фактично використовують елементи A-life, налічує вже декілька десятиліть. Багаточисельні приклади таких застосувань можна знайти в «NASA Technology», з якого недавно був знятий гриф секретності.

Скільки завгодно красива і цікава теорія опановує маси лише тоді, коли на ринок виходять прості і зручні інструментальні засоби, що роблять експерименти з новими концепціями загальнодоступними. На щастя, для A-life такий інструмент вже існує. Основним компонентом систем A-life є пакети, що реалізовують генетичні алгоритми. На ринку програмних засобів представлено декілька таких пакетів:

- Arlequin - аналіз генетичних для популяції даних;
- Genepop - генетичний для популяції аналіз;

- Genetix - генетичний для популяції аналіз (програма доступна лише французькою мовою);
- MacClade - комерційна програма для інтерактивного еволюційного аналізу даних;
- MEGA - молекулярно-еволюційний генетичний аналіз;
- Populations - генетичний для популяції аналіз.

Найбільш поширеними вважаються два програмних пакета: пакет Evolver (перший з масових пакетів GA), а також пізніший і потужніший пакет GeneHunter фірми Ward Systems Group. Останній особливо популярний, оскільки входить до складу нейромережевого пакету Ward, активно вживаного банкірами для портфельної гри і валютного ділінга.

GeneHunter. Потужний програмний комплекс під назвою GeneHunter призначений для рішення оптимізаційних задач будь-якої складності за допомогою генетичних алгоритмів і складається з наступних трьох частин: надбудови для MS Excel, динамічної бібліотеки функцій GALIB і прикладів задач, вирішених на пакеті.

Використання надбудови GeneHunter для Excel. Для того, щоб створити модель задачі в GeneHunter, необхідно внести дані, що підлягають обробці, в робочий аркуш Excel і визначити параметри рішення задачі в діалоговому вікні GeneHunter (рис. 5.18).

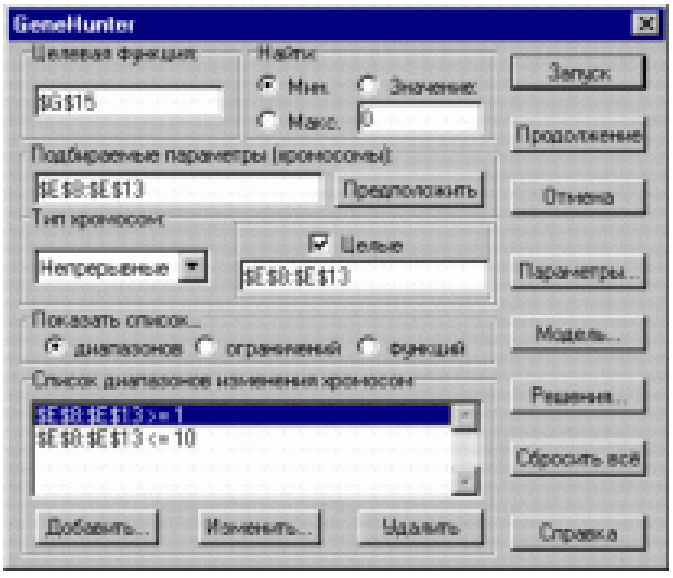


Рис. 5.18. Діалогове вікно GeneHunter.

Діалогове вікно GeneHunter дає можливість вказати ячейки в робочому аркуші, які використовуються при рішенні задачі. Тут також визначається метод рішення і створюється список обмежень, які мають бути дотримані в процесі рішення.

Вікно «Цільова функція» передає в GeneHunter повідомлення про положення ячейки з формулою, по якій визначається, наскільки хороше рішення задачі знайшов GeneHunter. Формула може бути створена за допомогою будь-якої функції Excel, доступної з меню «Вставка». Для створення формул також можна використовувати макроси Excel або функції Excel Visual Basic, які дозволяють вирішувати дуже складні задачі.

Діалогове вікно GeneHunter «Хромосоми» використовується для задання змінних, значення яких треба підібрати для того, щоб вирішити задачу. Їх значення зрештою визначають величину цільової функції. GeneHunter використовує два типи хромосом:

1. Безперервні хромосоми використовуються у тому випадку, коли параметр може набувати значень з деякої безперервної області, наприклад, значення 1,5 усередині діапазону від 0 до 2. Безперервні хромосоми можуть також бути цілими числами, якщо необхідно обмежити простір пошуку.

2. Перерахуємі хромосоми використовуються в задачах пошуку оптимальних комбінацій типа вибору маршруту, складання розкладу занять або послідовності процесів і тому подібне

Діалогове вікно GeneHunter «Діапазони, обмеження і додаткові цільові функції» дозволяє зробити наступне.

- Вказати діапазони значень кожної хромосоми, в яких GeneHunter шукатиме рішення.
- Додати до первинної цільової функції додаткові умови. Такі умови називаються обмеженнями. GeneHunter намагатиметься шукати рішення, які задовольняють обмеженням, але і одночасно оптимізують цільову функцію.
- Вказати ячейки додаткових цільових функцій, значення яких оптимізуватимуться одночасно із значенням головної цільової функції.

Динамічні бібліотеки функцій GALIB. Багато користувачів хочуть використовувати потужність генетичних алгоритмів в своїх додатках, проте вважають за краще розробити власний інтерфейс або скоротити час обчислень складної цільової функції в порівнянні з часом, який ця процедура займає в Excel. Для задоволення цих потреб до складу GeneHunter включена повна динамічна бібліотека функцій генетичних алгоритмів - GALIB.DLL. У бібліотеку входять функції створення популяції, визначення параметрів еволюції (таких як вірогідність схрещування, мутації, різноманітності), визначення значень цільової функції індивідуумів, оновлення популяції і переходу в наступне покоління. Користувач має можливість створювати індивідуумів з безперервними або перерахуємими хромосомами (рис. 5.19).

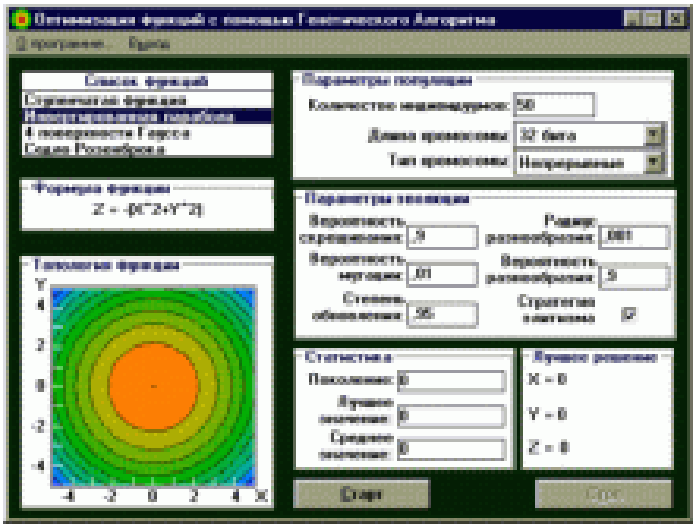


Рис. 5.19. Діалогове вікно динамічної бібліотеки.

Динамічна бібліотека GALIB.DLL дозволяє створювати додатки, в яких можуть розвиватися одночасно до 128 популяцій. Наприклад, функція MakeChromosomePool дає можливість швидко створювати множину схожих між собою хромосом, що буває зручно для таких застосувань, як оптимізація вагів нейронної мережі. GeneHunter дозволяє використовувати навіть суміш безперервних та перерахуємих хромосом в одній популяції.

Приклади використання GeneHunter. До складу пакету GeneHunter входять ряд прикладів використання пакету з робочих листів Microsoft Excel і із зовнішніх програм. Зупинимося коротко на деяких з них.

1. Аналіз портфеля акцій.

«Портфель акцій» є характерним прикладом множини задач, які вимагають групування різних по своїм властивостях об'єктів оптимальним способом. У даному прикладі фінансовий менеджер хоче розподілити наявні акції різної вартості на шість груп так, щоб вартість кожної групи складала заданий відсоток від вартості всього портфеля.

2. Оптимізація інвестиційного портфеля.

Цей приклад має більше відношення до задач, що реально виникають в житті. У реальних задачах оптимізації портфеля трейдер або керівник фондами старається мінімізувати ризик, одночасно намагаючись максимізувати дохід. Він може зробити це, намагаючись отримати портфель, вартість якого моделює поведінку вартості існуючого оптимального портфеля, такого, як S&P 500. Трейдер вибирає деяку кількість акцій, наприклад, ті, які він вважає найбільш прибутковими. ГА використовується для мінімізації ризику, допомагаючи скласти портфель так, щоб його вартість поводилася подібно до вартості більш диверсифікованого портфеля S&P 500.

3. Прогнозування індексу NYSE.

В даному прикладі GeneHunter створює правила для прогнозування зростання індексу NYSE. Формулювання цього завдання для GeneHunter декілька відрізняється від звичайного формулювання завдання для ГА. Кожен індивідуум в популяції представляє правило наступного вигляду: «якщо індекс close 4 дні тому назад менше, ніж індекс high 9 днів тому назад і якщо індекс high 3 дні тому назад більше, ніж індекс close 4 дні тому назад, тоді індекс close завтра зросте» (рис. 5.20). Цільова функція в цьому випадку має бути доходом, який можна отримати в разі використання даного правила при торгівлі. Концепція використання генетичних алгоритмів для пошуку правил може бути

поширена на задачі пошуку оптимальних правил для обробки даних, складання розкладів руху літаків або доставки вантажів і так далі.

Рис. 5.20. Прогнозування індексу NYSE.

4. Задача комівояжера.

Задача комівояжера - це широко відома задача, що стало тестом для перевірки і порівняння різних алгоритмів рішення комбінаторних задач оптимізації. Комівояжер повинен зробити замкнутий маршрут через задану кількість міст. Всі міста зв'язані між собою дорогами, і кожне місто комівояжер повинен відвідати лише один раз. GeneHunter вирішує цю задачу, вибираючи порядок відвідин міст і мінімізуючи довжину маршруту.

5. Створення оптимального графіка робіт.

Одним з найбільш важливих вживань генетичних алгоритмів є їх використання для складання оптимального розкладу для задач практично будь-яких розумних розмірів. ГА дають прекрасну можливість підприємцям планувати доручення для своїх співробітників, а технологам складати розклад технологічних процесів в конкретних умовах виробництва.

У даному прикладі ми представимо себе в ролі керуючого невеликим заводом по виробництву печатних плат, який щодня здійснює збірку, монтаж, паяння і тестування деякої кількості печатних плат різних типів. Кожна з плат повинна пройти через п'ять робочих станцій, проте на кожній з них можуть працювати по декілька фахівців (або верстатів), кожен з яких в змозі незалежно від інших повністю виконувати всі роботи, що входять в обов'язки даної

робочої станції. На заводі виготовляються декілька типів печатних плат, і кожна з них вимагає свого часу завантаження для кожної робочої станції. Тому є небезпека простою фахівців або верстатів, оскільки плати можуть зажадати довгого часу обробки на попередніх робочих станціях. GeneHunter створює розклад, що дозволяє виробити всі плати за мінімальний час.

6. Генетичне тренування нейронної мережі.

Штучні нейроні мережі спочатку були розроблені для моделювання здатності мозку розпізнавати образи. В даний час вони широко застосовуються для вирішення багатьох практичних задач передбачення і класифікації. Існують багато способів тренування нейромереж, і одним з них є використання генетичних алгоритмів. Приклад «Нейрона мережа» демонструє, як це можна зробити. У даному прикладі показаний один із способів, як можна використовувати GeneHunter для створення і тренування нейронної мережі, не застосовуючи жодних інших нейромережевих програм і алгоритмів (рис. 5.21).

Рис. 5.21. Генетичне тренування нейронної мережі.

7. Поліноміальна апроксимація функціональних залежностей.

GeneHunter можна використовувати для пошуку математичних моделей даних, багато в чому подібних до моделей, які можуть бути побудовані за допомогою нелінійного регресійного аналізу. В даному прикладі GeneHunter використовується для знаходження коефіцієнтів при п'яти незалежних змінних, а також показника міри при кожній змінній. Приклад легко доопрацювати для побудови складніших математичних моделей.

Auto2Fit 3. Програмний комплекс є потужним і в той же час легким у використанні інструментом, призначеним для вирішення оптимізаційних задач і проведення складних розрахунків. Програма Auto2Fit має в своєму розпорядженні вісім оптимізаційних алгоритмів, включаючи генетичні, і їх багаточисельні модифікації. Якщо говорити про ГА, то пакет дозволяє кодувати рішення для більшої ефективності, має шість типів операторів схрещування (кросинговера) і сім можливих варіантів відбору. Програма може працювати в одному з наступних режимів: швидкому і програмному. З додаткових можливостей Auto2Fit можна відзначити зручну навігацію по файлах, що реалізовується за допомогою дерева каталогів, робочу область, представлену у вигляді таблиць Excel, побудова 3D-графиков і інше. У комплект також включені багаточисельні приклади, у тому числі і класичні приклади задач оптимізації.

GeneBase. Бібліотека компонент для Delphi, що реалізовує генетичні алгоритми - потужний засіб рішення задач багатовимірної непараметричної оптимізації. У наборі є компонент, що реалізовує генетичний алгоритм, який може бути використаний для рішення задач багатовимірної непараметричної оптимізації. Зокрема він дозволяє знаходити субоптимальне рішення NP-повних задач (рис. 5.22).

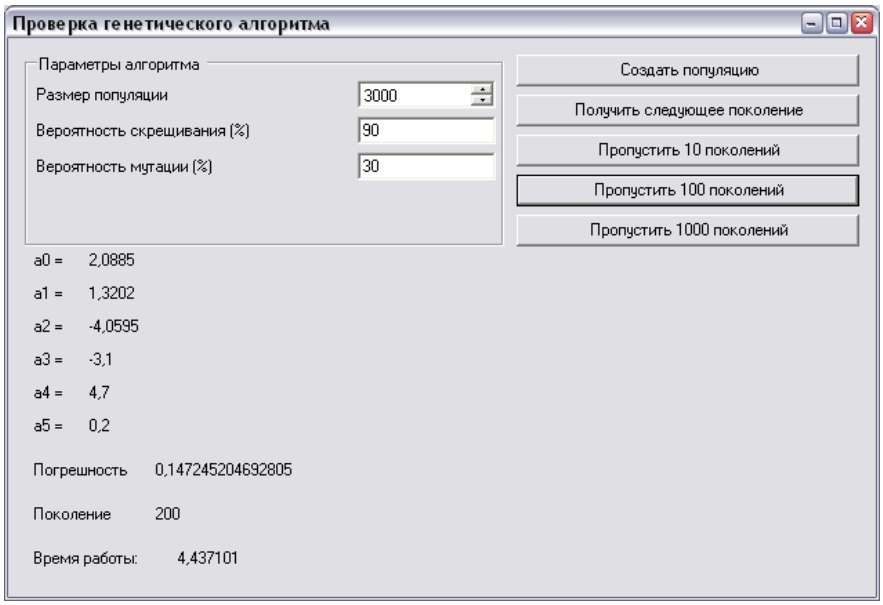


Рис. 5.22. Реалізація генетичного алгоритму на основі GeneBase.

Genetic Algorithm and Direct Search Toolbox. Genetic Algorithm and Direct Search Toolbox призначений для розширення функціональних можливостей пакету MATLAB і, зокрема, Optimization Toolbox новим виглядом алгоритмів оптимізації. У даному інструментарії містяться нові можливості по вживанню відомих алгоритмів оптимізації для такого класу задач, який представляє певні труднощі при вирішенні звичайними методами оптимізації. Подібні методи і алгоритми найчастіше використовуються у разі, коли шукана цільова функція є переривистою, істотно нелінійною, стохастичною і не має похідних або ці похідні є недостатньо визначеними. Цей модуль може розглядатися і як деяке доповнення до інших методів оптимізації як деякий засіб для пошуку прийнятної початкової точки розрахунку по основному алгоритму оптимізації. Алгоритм може служити і як засіб для подальшого уточнення раніше використаних основних алгоритмів (рис. 5.23).

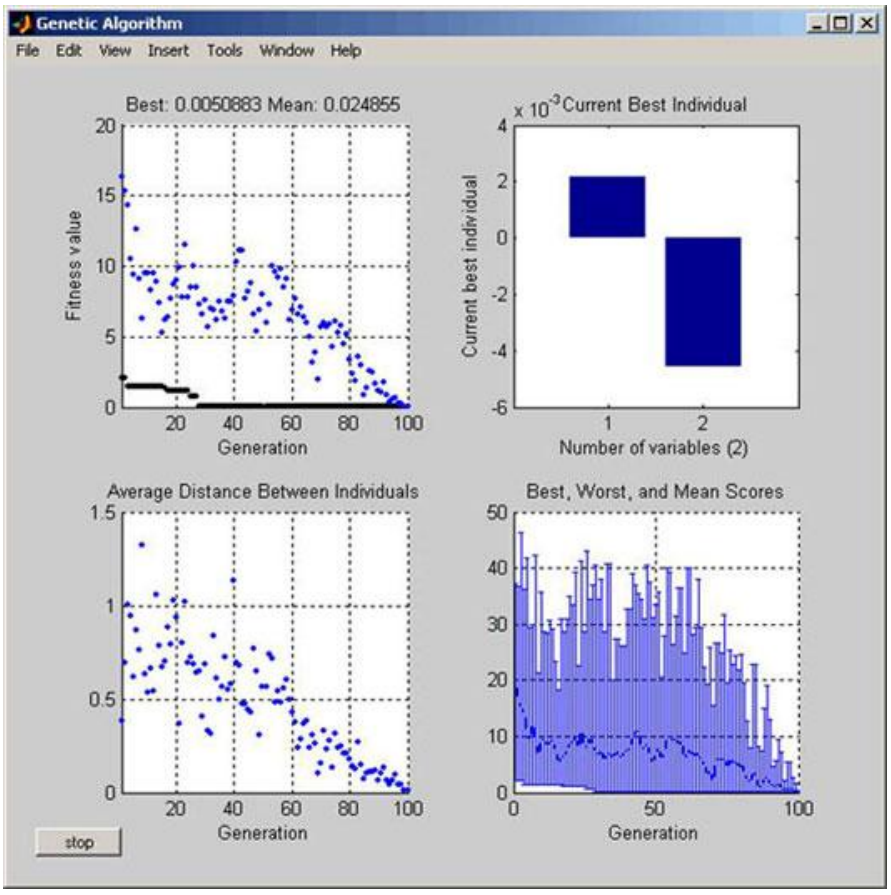


Рис. 5.23. Реалізація генетичного алгоритму засобами MATLAB.

Звернення до функцій Genetic Algorithm and Direct Search Toolbox можливо за допомогою основного графічного інтерфейсу або через командний рядок MATLAB з використанням відкритої алгоритмічної мови. Також є відповідні інструментарії для управління процесом оптимізації і контролю ефективності виконання і введення критеріїв останову виконання програми. Під час виконання процесу оптимізації з метою уточнення рішення і коректування отримуваних результатів є можливість оперативної зміни опцій виконуваного завдання. Такий підхід означає, що будь-який користувач має можливість відстежування алгоритму, модифікації вихідних кодів і створення власних оригінальних програм.

Алгоритм допоможе вирішити задачі, які або неможливо вирішити звичайними методами або виникають нестійкі рішення при вживанні стандартних математичних методів. Графічний інтерфейс користувача допоможе швидко встановити тип вирішуваної задачі і отримати її рішення (рис. 5.24).

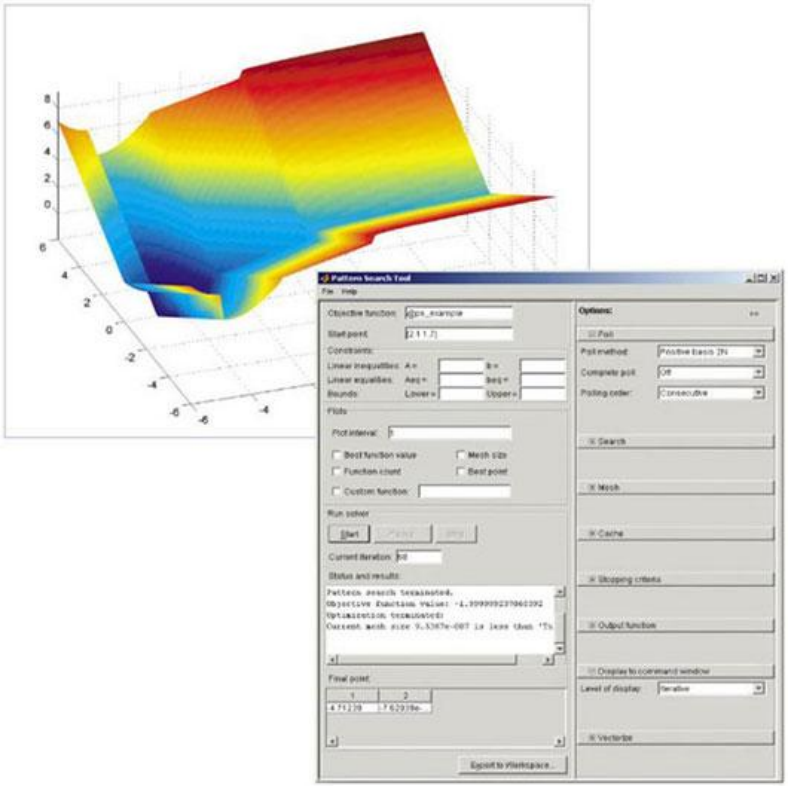


Рис. 5.24. Графічний інтерфейс Genetic Algorithm.

Genetic Algorithm and Direct Search Toolbox тісно інтегрований з пакетом MATLAB і Optimization Toolbox. Така інтеграція дозволяє використовувати генетичний алгоритм і методи безпосереднього пошуку для визначення найкращої стартової точки і, відповідно, більш повно використовувати можливості вирішувачів Optimization Toolbox або програм пакету MATLAB для підвищення точності вирішення задач оптимізації (рис. 5.25).

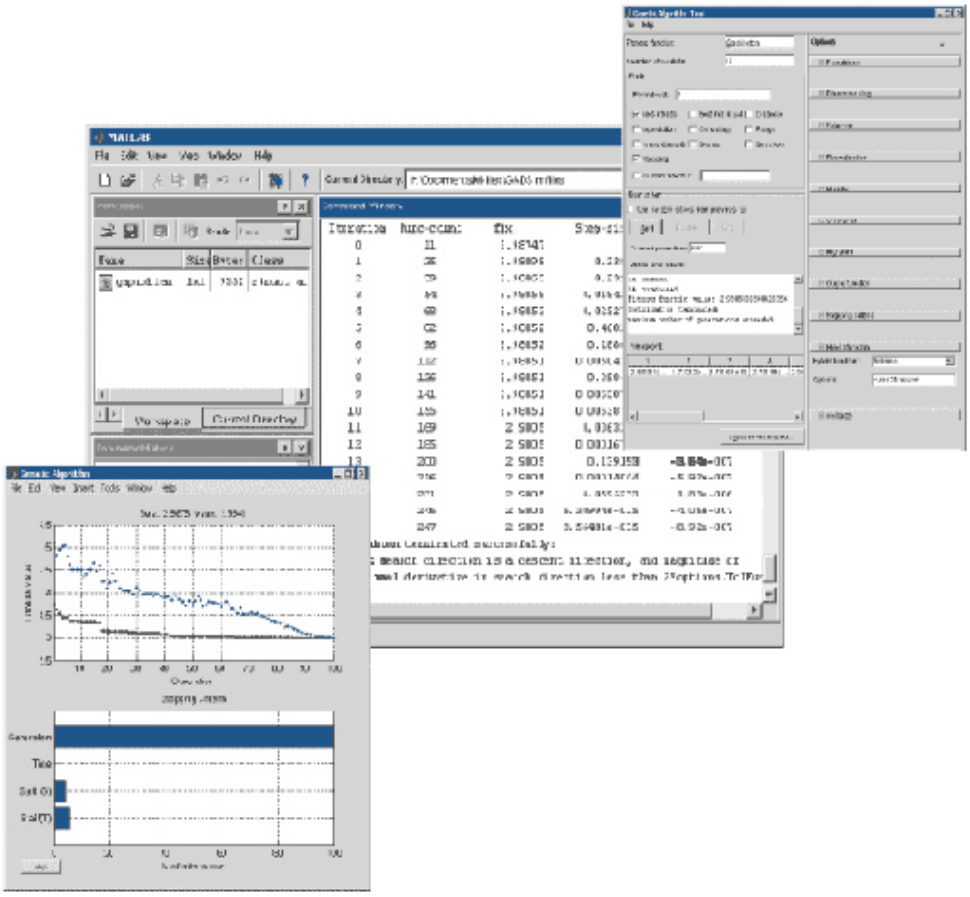


Рис. 5.25. Інтеграція генетичного алгоритму з іншими програмами MATLAB.

У даний інструмент включений ряд графічних функцій для відображення результатів оптимізації. Подібна візуалізація дає можливість встановити динамічний зворотній зв'язок з процесом оптимізації і проводити необхідні модифікації під час виконання програм. Спеціалізовані графічні функції є як для генетичного алгоритму, так і для методу прямого пошуку. Крім того, є можливість поєднання графіків, виділення окремих графіків для

ретельнішого аналізу і введення власних графічних відображень користувача. Так само є можливість експорту алгоритмічних опцій виконаних задач з подальшим їх імпортом в графічний інтерфейс.

Апаратна реалізація генетичних алгоритмів. В даний час спостерігається зростаючий інтерес до вживання генетичних алгоритмів в автономних системах і розвитку нового напряму в області проектування апаратних засобів, який отримав назву еволюційні апаратні засоби. Вперше ця ідея була запропонована С. Луїсом [Louis] і Д. Раулінсом [Rawlins] в 1991 році та експериментально перевірена в області цифрових схем. Надалі, запропоновані методи були розвинені і доопрацьовані, і отримали вживання в багатьох автоматизованих системах проектування апаратури. Зараз бурхливо розвивається один з найбільш перспективних напрямів в даної області, пов'язаний з проектуванням самореконфігуруємих апаратних засобів і автоматичного проектування схем, реалізації динамічної реконфігурації в мобільних системах, побудови реконфігуруємих апаратних засобів на одному кристалі БІС. Використання генетичних алгоритмів як механізму для автоматичного проектування схем на реконфігуруємих платформах, отримало назву еволюційні апаратні засоби (Evolvable Hardware), яке також використовується синонімом для загального напряму, відомого як еволюційна електроніка (Evolutionary Electronics).

Новий напрям використання генетичних алгоритмів - побудова динамічно реконфігуруємих апаратних засобів, в яких виробляється еволюційна зміна архітектури системи в режимі реального часу відповідно до зміни зовнішніх чинників. При реалізації подібних систем, ГА як правило виступає як зовнішній апаратний модуль або вбудовується в один кристал з реконфігурованою апаратною системою.

Апаратна реалізація компактного генетичного алгоритму передбачає перехід від програмної реалізації компактного ГА до апаратного виконання на прикладі реалізації алгоритму на FPGA фірми Xilinx. В результаті, апаратно реалізований компактний ГА виконує одну генерацію за три тактові цикли для

задачі one-max. Даний алгоритм був також розглянутий Джоном Галлагхером і доопрацьований для вживання в еволюційних апаратних засобах, на прикладі використання алгоритму в схемі управління, в якому реконфігурована аналогова нейронна мережа змінюється в інтерактивному режимі, для управління фізичними процесами. У структуру компактного ГА були додані деякі генетичні оператори, наприклад стратегія елітизма, оператор мутації і схема перевиборки кращого рішення (champion resampling), а також були внесені зміни в структуру алгоритму, що дозволило добитися здобуття якісно нових результатів при тестуванні алгоритмами ДеДжонга [DeJong]. Основний ухил при апаратній реалізації робився в області зменшення споживаної потужності і необхідного простору кристала для розміщення алгоритму.

Іншим рішенням підвищення ефективності генетичних алгоритмів і сфери їх застосування є апаратна реалізація генетичного алгоритму UMDA (Univariate Marginal Distributional Algorithm) (рис. 5.26).

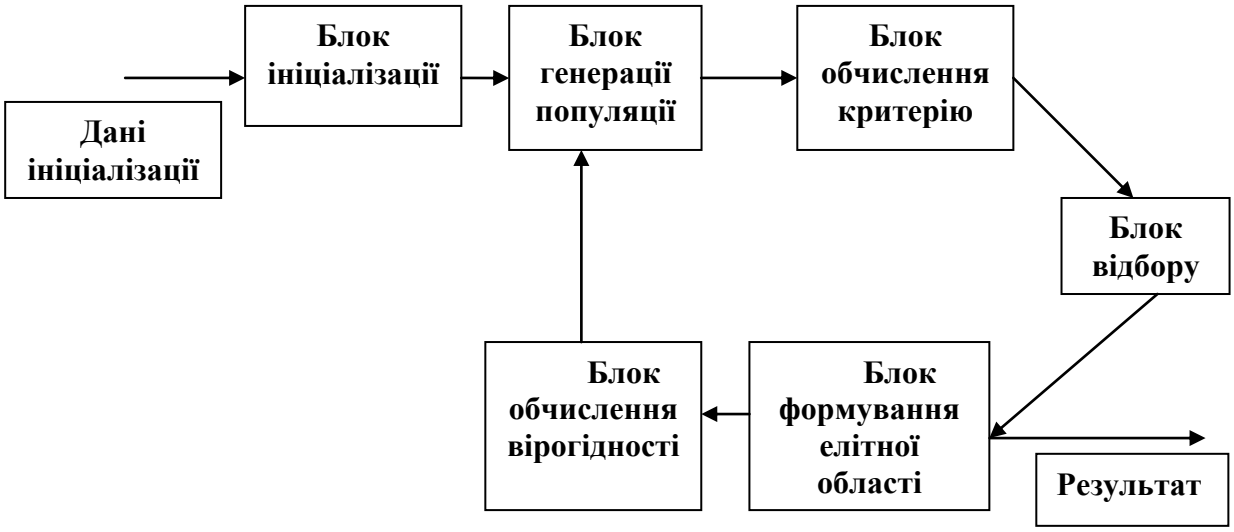


Рис.5.26. Структурна схема функціонування імовірнісного генетичного алгоритму UMDA.

Функціонування алгоритму полягає в послідовному виконанні генетичних операторів. При ініціалізації виробляється налаштування параметрів функціонування алгоритму. Блок генерації популяції відповідає за формування

популяції за допомогою послідовної генерації особин (хромосом). Для кожної хромосоми виробляється обчислення критерію відбору (функції придатності), на основі якого виконується відбір кращих особин і формування елітної області. На основі сформованої елітної області виробляється обчислення вірогідності, необхідної для генерації наступного покоління. Ознакою знаходження рішення є рішення задачі one-max або виконання заданої кількості ітераційних циклів.

Розглянемо деякі найбільш значущі сфери практичного застосування генетичних алгоритмів.

Задача комівояжера. Задача комівояжера в теорії дискретної оптимізації вважається класичною задачею. Вперше вона була сформульована ще в 1759 році. Суть задачі полягає в тому, аби знайти найкоротшу замкнуту дорогу обходу декількох міст, заданих своїми координатами (або за допомогою матриці відстаней між ними). Міста можуть відвідуватися лише один раз. Відомо, що вже для 50 міст пошук оптимальної дороги є складним завданням, що спонукало розвиток різних нових методів (у тому числі нейромережевих і генетичних алгоритмів). Це задача NP-повна (задача з експоненціальною оцінкою числа ітерацій, необхідних для відшукання точного рішення) і мультимодальна (має локальні екстремуми). ГА використовується для знаходження околооптимального шляху за лінійний час.

Функція пристосованості. Значення функції пристосованості повинне відповідати відстані, яку проходить комівояжер згідно шляху, що представляється хромосомою. Оскільки це значення має бути мінімальне, то кінцева формула функції пристосованості j хромосоми часто виглядає таким чином: $f_j = 1.1 \cdot d_{\max} - d_j$, де $d_{\max}$ - довжина максимального маршруту в поточній популяції, d_j - довжина маршруту, що представляється j хромосомою. Значення цієї функції чим більше, тим краще.

Кодування рішень. В даний час існує чотири основні представлення маршруту комівояжера у вигляді хромосоми: сусідське, порядкове, шляхове і матричне. Оскільки класичні оператори схрещування і мутації для них, як

правило, непридатні, кожне з цих представлень має власні «генетичні» оператори, всі вони дуже сильно розрізняються. Деякі з розглянутих нижче кросоверів можуть створювати потомство, що не має рішення (невалідні рішення). Невалідні рішення можуть бути корисні для створення і внесення різноманітності в популяцію, проте цим вони можуть уповільнити або навіть запобігти збіжності ГА. Одним з рішень цієї проблеми може стати ідея відновлення невалідних рішень.

Сусідське представлення. Сусідське представлення представляє маршрут як список з n міст. Місто j знаходиться на позиції i лише в тому випадку, якщо маршрут проходить з міста i в місто j. Наприклад, маршрут з наступним порядком обходу міст: $1 \to 2 \to 4 \to 3 \to 8 \to 5 \to 9 \to 6 \to 7 \to 1$ представляється як хромосома $P = [2\ 4\ 8\ 3\ 9\ 7\ 1\ 5\ 6]$. Кожен маршрут має лише одне сусідське представлення, але не кожен вектор в сусідському представленні відповідає якому-небудь маршруту. Наприклад, вектор [2 4 8 1 9 3 5 7 6] позначає послідовність $1 \to 2 \to 4 \to 1...$, тобто частина маршруту - це замкнутий цикл, що недопустимо.

Сусідське представлення не підтримує класичну операцію кросовера (він практично завжди породжує недопустимі маршрути). Для нього розроблено три власні оператора.

Кросовер альтернативних ребер. Кросовер alternating edges будує нащадків, вибираючи по черзі перше ребро від першого батька, потім друге ребро від другого, потім знову наступне від першого і так далі. Якщо нове ребро представляє замкнутий цикл, беруть ребро з того ж батька (порушуючи чередування). Якщо і воно утворює цикл, беруть випадкове ребро, яке ще не вибиралося і не утворює замкнутого циклу. Другий нащадок будується аналогічно, але перше ребро беруть від другого батька. Для прикладу розглянемо побудову одного з нащадків двох хромосом:

$$P_1 = [2\ 3\ 8\ 7\ 9\ 1\ 4\ 5\ 6],$$

$$P_2 = [7\ 5\ 1\ 6\ 9\ 2\ 8\ 4\ 3].$$

Крок 1. П1 = [2 _ _ _ _ _ _ _ _]. (У маршрут входить ребро $1 \to 2$).

Крок 2. П1 = [2 5 _ _ _ _ _ _ _]. (Маршрут містить фрагмент $1 \to 2 \to 5$).

Крок 3. П1 = [2 5 8 _ _ _ _ _ _]. (Маршрут містить фрагменти $1 \to 2 \to 5,3 \to 8$).

Крок 4. П1 = [2 5 8 6 _ _ _ _ _]. (Маршрут містить фрагменти $1 \to 2 \to 5,3 \to 8,4 \to 6$).

Крок 4. П1 = [2 5 8 6 9 _ _ _ _]. (Маршрут містить фрагменти $1 \to 2 \to 5 \to 9,3 \to 8,4 \to 6$).

Крок 5. П1 = [2 5 8 6 9 1 _ _ _]. (Маршрут містить фрагменти $4 \to 6 \to 1 \to 2 \to 5 \to 9,3 \to 8$).

Крок 6. П1 = [2 5 8 6 9 1 4 _ _]. (Маршрут містить фрагменти $7 \to 4 \to 6 \to 1 \to 2 \to 5 \to 9,3 \to 8$).

Крок 7. П1 = [2 5 8 6 9 1 4 7 _]. (Маршрут містить фрагмент $3 \to 8 \to 7 \to 4 \to 6 \to 1 \to 2 \to 5 \to 9$). На цьому етапі нащадок отримав ребро $8 \to 7$, не присутнє ні у кого з батьків, оскільки додавання в маршрут 5-го або 4-го батьківських міст привело б до утворення замкнутого підциклу.

Крок 8. П1 = [2 5 8 6 9 1 4 7 3]. (Маршрут має вигляд $3 \to 8 \to 7 \to 4 \to 6 \to 1 \to 2 \to 5 \to 9 \to 3$).

Кросовер фрагментів. Кросовер subtour chunks створює нащадків, вибираючи (випадково) фрагмент від одного з батьків, потім випадкової довжини фрагмент від іншого з батьків, і так далі. І знову, якщо на якомусь рівні утворюється замкнутий цикл, він вирішується аналогічним чином (вибирається інший випадковий фрагмент, який ще не вибирався).

Евристичний кросовер. Heuristic crossovers будує нащадків, вибираючи випадкове місто як стартову точку для маршруту - нащадка. Потім він порівнює два відповідні ребра від кожного з батьків і вибирає більше коротке. Потім кінцеве місто вибирається як початкове для вибору наступного коротшого ребра з цього міста. Якщо на якомусь кроці виходить замкнутий маршрут, він продовжується будь-яким випадковим містом, яке ще не відвідувалося.

Переваги сусідського представлення в тому, що воно дозволяє схематично аналізувати створювані маршрути. Це представлення має в основі

«будівельні блоки» – зв'язки між містами. Наприклад, схема (* * * 3 * 7 * * *) описує множину всіх маршрутів з ребрами $(4 \to 3)$ і $(6 \to 7)$. Основний же недолік даного представлення - в тому, що множина всіх його операцій дуже бідна. Зокрема, для нього не існує простих алгоритмів мутації. Кросовер alternating edges часто руйнує хороші маршрути, які були у обох батьків до вживання цієї операції. Кросовер subtour chunks має кращі характеристики, чим перший, завдяки тому, що його руйнівні властивості менші. Але все одно його експлуатаційні якості все ж досить низькі. Кросовер heuristic crossover, звичайно ж, найкращий оператор для даного представлення. Причина в тому, що попередні операції сліпі, вони не беруть в розрахунок справжню довжину ребер. З іншого боку, heuristic crossover вибирає краще ребро з двох можливих. Може вийти, що подальша дорога буде неможлива і доведеться вибирати ребро, довжина якого невиправдано велика.

Порядкове представлення. Порядкове представлення визначає маршрут як список з n міст; i елемент списку - номер від 1 до $n - i - 1$. Ідея порядкового представлення полягає в наступному. Є впорядкований список міст, який служить для зв'язку маршрутів з їх порядковим представленням. Передбачимо, що такий впорядкований список простий: $C = [1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9]$. Тоді маршрут $1 \to 2 \to 4 \to 3 \to 8 \to 5 \to 9 \to 6 \to 7 \to 1$ буде представлений як список $L = [1\ 1\ 2\ 1\ 4\ 1\ 3\ 1\ 1]$. Він може бути інтерпретований таким чином.

- Оскільки перший номер списку L дорівнює 1, беремо перше місто із списку C як перше місто маршруту (місто номер 1) і виключаємо його із списку. Поточний фрагмент маршруту - це 1.
- Наступний номер в списку L також 1, тому беремо перший номер з списку $C = [\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 9]$, що залишився. Оскільки ми виключили із списку C 1-е місто, наступне місто - 2. Виключаємо і це місто із списку. Маршрут на даному кроці набуває вигляду: $1 \to 2$.
- Наступний номер в списку $L - 2$. Беремо із списку $C = [\ 3\ 4\ 5\ 6\ 7\ 8\ 9]$ 2-ге по порядку місто, що залишилося. Це - 4. Виключаємо його із списку. Маршрут $1 \to 2 \to 4$.

- Наступний номер в списку $L-1$. Беремо із списку $C=[\ 3\ 5\ 6\ 7\ 8\ 9]$ 1-е місто - з номером 3. Маємо маршрут $1\to 2\to 4\to 3$.
- Наступний номер в списку $L-4$, беремо 4-е місто із списку C, що залишився =[5 6 7 8 9] - це 8. Маршрут $1\to 2\to 4\to 3\to 8$.
- Наступний номер в списку $L-1$. Беремо із списку $C=[\ 5\ 6\ 7\ 9]$ 1 -е місто - з номером 5. Маршрут $1\to 2\to 4\to 3\to 8\to 5$.
- Наступний номер в списку $L-3$. Беремо третє місто із списку $C=[\ 6\ 7\ 9]$ - це 9. Видаляємо його з C. Маршрут $1\to 2\to 4\to 3\to 8\to 5\to 9$.
- Наступний номер в списку $L-1$, тому беремо перше місто з поточного списку $C=[\ 6\ 7]$ - місто номер 6, і видаляємо його з C. Частинний маршрут має вигляд $1\to 2\to 4\to 3\to 8\to 5\to 9\to 6$.
- Останнім номером в списку L завжди буде 1, тому беремо останнє місто, що залишилося, з поточного списку $C=[7]$, і видаляємо його з C. Остаточний маршрут має вигляд $1\to 2\to 4\to 3\to 8\to 5\to 9\to 6\to 7\to 1$.

Основна перевага порядкового представлення в тому, що класичний кросовер в даному випадку працює. Будь-які два маршрути в порядковому представленні, що розрізаються в будь-якій позиції і склеєні разом, породять двох нащадків, кожен з яких буде правильним маршрутом. Наприклад, два батька $P_1=[1\ 1\ 2\ 1|\ 4\ 1\ 3\ 1\ 1]$ та $P_2=[5\ 1\ 5\ 5\ |\ 5\ 3\ 3\ 2\ 1]$, які позначають відповідно маршрути $1\to 2\to 4\to 3\to 8\to 5\to 9\to 6\to 7\to 1$ та $5\to 1\to 7\to 8\to 9\to 4\to 6\to 3\to 2\to 5$, з точкою розрізу, позначеною « | » породять наступних нащадків: П1 = [1 1 2 1 5 3 3 2 1] та П2 = [5 1 5 5 4 1 3 1 1]. Ці нащадки позначають маршрути $1\to 2\to 4\to 3\to 9\to 7\to 8\to 6\to 5\to 1$ та $5\to 1\to 7\to 8\to 6\to 2\to 9\to 3\to 4\to 5$. Слід відмітити, що частини маршруту зліва від лінії розрізу не змінилися, тоді як частини маршруту праворуч від лінії розрізу мають досить багато відмінностей від закінчень батьківських хромосом. В якості мутації використовується зміна з вірогідністю p_m i елементу порядкового представлення на випадковий номер від 1 до $n-i-1$.

Шляхове представлення. Шляхове представлення - це найбільш природне представлення маршруту. Наприклад, тур

$5 \to 1 \to 7 \to 8 \to 9 \to 4 \to 6 \to 2 \to 3 \to 5$ буде представлений як рядок [5 1 7 8 9 4 6 2 3].

Частково відображаємий кросовер. Кросовер, що частково відображується, бере спочатку частину дороги одного батька і зберігає послідовність і позиції як можна більшого числа міст іншого батька. Точка кросовера вибирається випадково (рис. 5.27).

$$V_1 = 12 \mid 543$$
$$V_2 = 35 \mid 421$$

Рис. 5.27. Точка перетину частково відображаємого кросовера.

Потім виконується відображення замін перших частин хромосом $\{1 \Leftrightarrow 3, 2 \Leftrightarrow 5\}$, яке застосовується поточково до батьків для здобуття потомства. Перевіряється кожен елемент першого батька: якщо є для нього заміна, то вона виконується і потім копіюється ($\Updownarrow$) в першого нащадка, інакше він просто копіюється без заміни ($\Downarrow$) (рис. 5.28). Подібна процедура може бути використана і для другого нащадка (рис. 5.29, 5.30, 5.31).

$$V_1 = 1\ 2\ 5\ 4\ 3$$
$$\Updownarrow\ \Updownarrow\ \Updownarrow\ \Downarrow\ \Updownarrow$$
$$V_6 = 3\ 5\ 2\ 4\ 1$$

Рис. 5.28. Частково відображаємий кросовер.

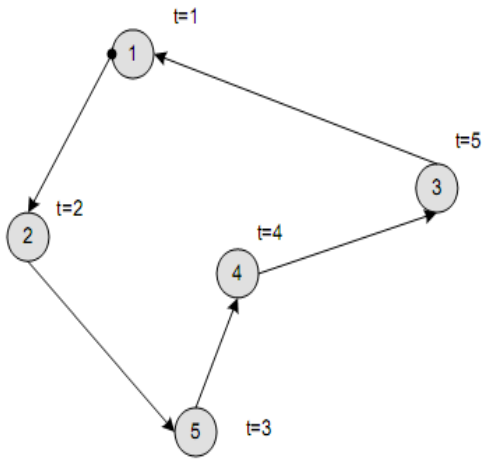


Рис. 5.29. Батько V_1.

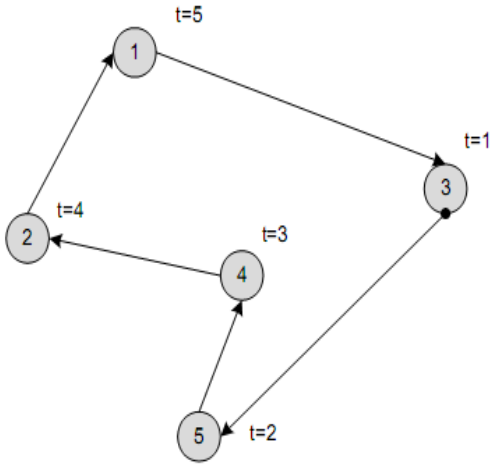


Рис. 5.30. Батько V_2.

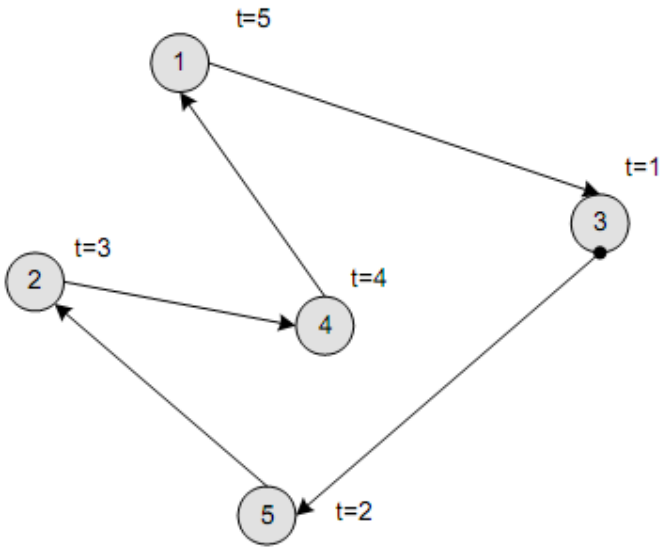


Рис. 5.31. Нащадок V_6.

Впорядкований кросовер. Впорядкований кросовер бере частину шляху одного батька і зберігає родинний порядок міст з іншого батька. Перші дві точки перетину кросовера вибираються випадково (рис. 5.32). Кожен елемент центральної секції першого батька копіюється в нащадка (рис. 5.33).

$V_1 = 1 \mid 2\ 5 \mid 43$

$V_2 = 3 \mid 5\ 4 \mid 21$

Рис. 5.32. Точки перетину кросовера.

$V_1 = 1\ 2\ 5\ 4\ 3$

$V_7 = \quad 2\ 5$

Рис. 5.33. Створення нового нащадка.

Потім елементи другого батька збираються в список (рис. 5.34), починаючи з другої точки кросовера. Нарешті, міста, вже представлені в нащадках, видаляються (рис. 5.35), і елементи, що залишилися, копіюються замість порожніх пропусків нащадка, починаючи з другої точки перетину кросовера (мал. 5.36).

[2 1 3 5 4]

Рис. 5.34. Список міст в 2 батьку.

[2 1 3 5 4] => [1 3 4]

Рис. 5.35. Видалення дубльованих міст.

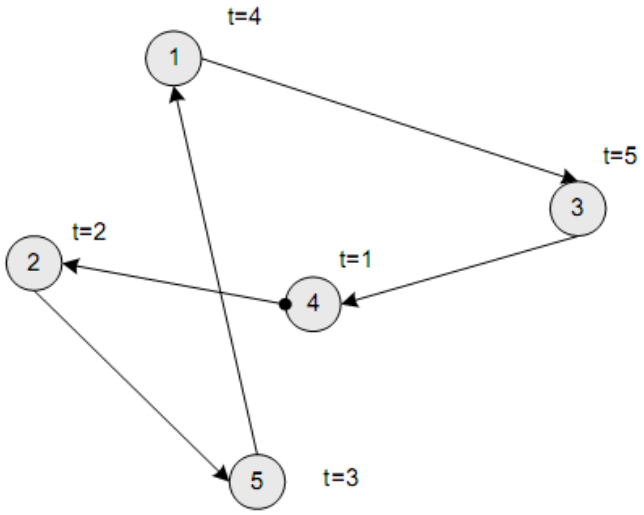


Рис. 5.36. Нащадок $V_7 = [4\ 2\ 5\ 1\ 3]$.

Кросовер рекомбінації ребер. Кросовер рекомбінації ребер робить потомство лише за допомогою ребер, представлених в обох батьках. Спочатку з двох батьків $V_1 = [1\ 2\ 5\ 4\ 3]$ та $V_2 = [3\ 5\ 4\ 2\ 1]$ будується список ребер (табл. 1). Міста в нащадках вибираються поодинці, при цьому вибирається місто з найменшою кількістю ребер. Першим елементом в хромосомі є місто з маленькою кількістю зв'язків. Після того, як місто вибране, воно видаляється з таблиці, і міста, приєднані до нього, розглядаються як кандидати при наступному виборі (рис. 5.37).

Табл. 1.

Список ребер.

Місто	Сполучений з		
1	2	3	
2	1	5	4
3	4	1	5
4	5	3	2
5	2	4	3

Хромосома на першому кроці - $V_9 = [1\ \_\ \_\ \_\ \_\ ]$ (табл. 2).

Табл. 2.

Список ребер після першого кроку.

Місто	Сполучений з		
2	5	4	
3	4	5	
4	5	3	2
5	2	4	3

Хромосома на другому кроці - $V_9 = [1\ 3\ \_\ \_\ \_\ ]$ (табл. 3).

Табл. 3.

Список ребер після другого кроку.

Місто	Сполучений з	
2	5	4
4	5	3
5	2	4

Хромосома на третьому кроці - $V_9 = [1\ 3\ 5\ \_\ \_\ ]$ (табл. 4).

Табл. 4.

Список ребер після третього кроку.

Місто	Сполучений з
2	4
4	2

Хромосома на четвертому кроці - $V_9 = [1\ 3\ 5\ 2\ \_\ ]$ (табл. 5).

Табл. 5.

Список ребер після четвертого кроку.

Місто	Сполучений з
4	

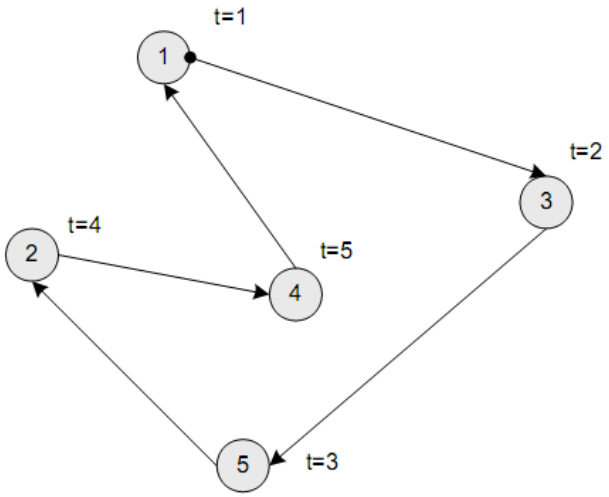


Рис. 5.37. Отриманий нащадок $V_9 = [1\ 3\ 5\ 2\ 4]$.

Мутація. Мутація працює за наступним принципом. Вибираються випадковим чином два міста в хромосомі і міняються місцями.

«Жадібна мутація». «Жадібна мутація» (greedy reconnection) полягає у впорядковуванні послідовності міст. Її вживання допомагає добре шукати локальний оптимум. Спочатку випадковим чином вибираються дві точки перетину в хромосомі так, щоб між ними було не менше чим 2 міста. Потім послідовність міст між точками перетину упорядковується: вони переставляються залежно від близькості один до одного. У нашому випадку серед міст 2, 5 і 4 визначається місто, найближче до міста 1. Хай, наприклад, це місто 5. Він ставиться слідом за містом 1. Потім серед міст 2 і 4 визначається найближчий до міста 5. Хай, наприклад, це місто 4. Ставимо його за містом 5. На останнє місце ставимо місто, що залишилося, 2. В результаті маємо модифіковану хромосому.

Крок 1. $V_1 = [1 \mid 2\ 5\ 4 \mid 3]$, $V_1^* = [1\ \_\ \_\ \_\ 3]$.

Крок 2. $V_1^* = [1\ 5\ \_\ \_\ 3]$.

Крок 3. $V_1^* = [1\ 5\ 4\ \_\ 3]$.

Нащадок, отриманий при мутації: $V_1^* = [1\ 5\ 4\ 2\ 3]$.

Матричне представлення. Для кодування хромосоми також може служити бінарна матриця V, де $V_{ct} = 1$, якщо місто c відвідане у момент часу t

(знаходиться в маршруті у позиції t), інакше $V_{ct} = 0$. Наприклад, шляхи ABEDC (V_1) і CEDBA (V_2) можуть бути представлені у вигляді наступних матриць (рядки-міста, стовпці-час) (рис. 5.38):

$$V_1 = \begin{vmatrix} & 1 & 2 & 3 & 4 & 5 \\ A & 1 & 0 & 0 & 0 & 0 \\ B & 0 & 1 & 0 & 0 & 0 \\ C & 0 & 0 & 0 & 0 & 1 \\ D & 0 & 0 & 0 & 1 & 0 \\ E & 0 & 0 & 1 & 0 & 0 \end{vmatrix} \qquad V_2 = \begin{vmatrix} & 1 & 2 & 3 & 4 & 5 \\ A & 0 & 0 & 0 & 0 & 1 \\ B & 0 & 0 & 0 & 1 & 0 \\ C & 1 & 0 & 0 & 0 & 0 \\ D & 0 & 0 & 1 & 0 & 0 \\ E & 0 & 1 & 0 & 0 & 0 \end{vmatrix}$$

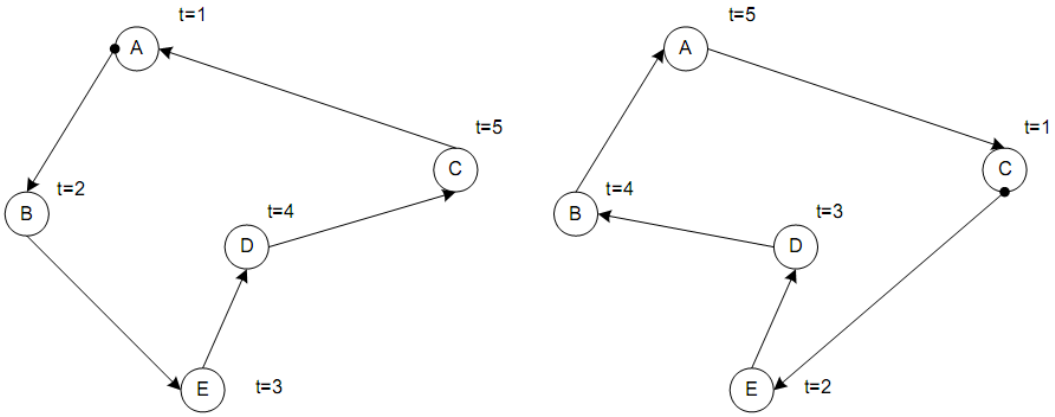


Рис. 5.38. Шляхи ABEDC та CEDBA.

Матриці можуть бути представлені у вигляді наступних хромосом:

$$V_1 = 100000100000010001000100$$

$$V_2 = 000010001010000010001000$$

Необхідно відзначити, що міста в однаковій послідовності, але з різними стартовими точками або з протилежним напрямом кодуються різними матрицями, тобто різними хромосомами. ГА може створити деякі хромосоми, що не мають шляхового представлення (невалідні рішення). Це може статися при створенні початкової популяції і при дії стандартних генетичних операторів.

Фокс (Fox) і Макмахон (McMahon) розглядали маршрут як двійкову матрицю, в якій елемент X_{ij} містить 1, якщо місто i стоїть в маршруті раніше, ніж місто j. Існує також альтернативне шляхове представлення , при якому в

матрицю в елемент X_{ij} записується 1 лише в тому випадку, якщо i місто безпосередньо передує j. Для цих представлень також розроблені способи схрещування і мутації, але вони досить складні, тому в алгоритмах частіше застосовуються розглянуті раніше способи символьного кодування хромосом (рис. 5.39).

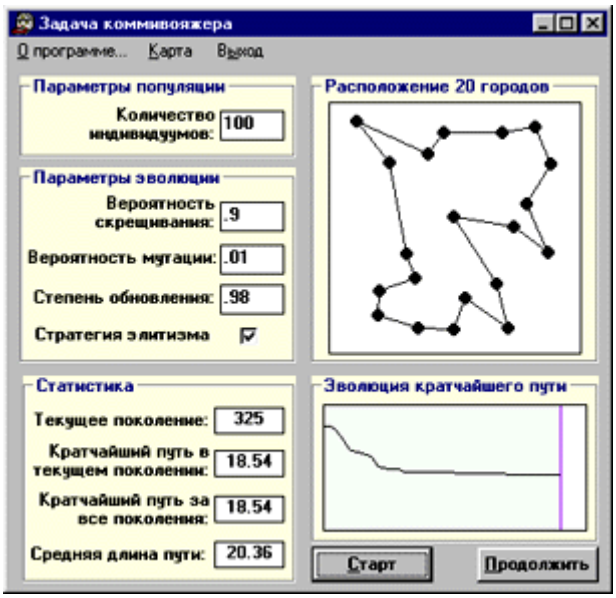


Рис. 5.39. Рішення задачі комівояжера в пакеті GeneHunter.

Аналіз ризиків проектів. В даний час підприємства, що займаються реорганізацією своєї діяльності і одночасно з цим впроваджують інформаційні системи управління, прагнуть скоротити терміни реалізації цих проектів до мінімуму, при цьому постійно корегуючи вимоги до створюваної інформаційної системи. Така ситуація приводить до багатократного зростання ризиків невдалого завершення цих проектів. Як правило, системні інтегратори, що виконують такі проекти, намагаються проводити аналіз можливих ризиків до початку, а також в ході реалізації проекту. Проте, на практиці часом складно, а деколи і неможливо визначити залежність різних процесів проекту від можливих ризиків, що діють на проект, ще складніше привести аналітичний опис такої залежності, тим більше на початкових етапах проекту. Навіть сама задача визначення ризиків, які можуть діяти на проект стає порівнянною по складності з вхідною задачею. Ці обставини значно ускладнюють вживання на

всіх етапах проекту класичних методів аналізу ризиків, оскільки більшість з них ґрунтується на використанні апріорної інформації про реакцію системи управління проектом на виникнення ризикових ситуацій в аналогічних, вже виконаних проектах. У зв'язку з цим встає задача побудови таких методів аналізу ризиків, які були б здатні виявляти ризики проекту практично при повній відсутності припущень про характер поведінки процесів проекту на різних етапах. Одними з таких методів є еволюційні методи пошуку, зокрема, генетичні алгоритми.

Генетичний алгоритм, побудований для такої задачі, володіє досить широкими можливостями. Відстроївши відповідним чином параметри системи можна управляти процесом пошуку залежно від поставленої задачі. Існує як мінімум три класи задач, які можуть бути вирішені представленим алгоритмом аналізу ризиків проекту:

- задачі визначення найбільшої вірогідності виникнення ризикової події. При рішенні цієї задачі встановлюється етап робіт проекту або стадія виконання процесу на якій існує найбільша вірогідність виникнення вказаної ризикової події;
- задачі визначення системи ризиків проекту. При рішенні цієї задачі встановлюються ризики проекту або процесів проекту, які мають найбільшу вірогідність передаватися з етапу в етап або з процесу в процес;
- задачі визначення профілів ризиків проекту або процесів проекту на всіх його етапах. При рішенні цієї задачі визначається взаємозв'язана система ризиків проекту що діє на даному етапі.

Генетичний алгоритм побудови розкладу для багатопроцесорних обчислювальних систем. Багатопроцесорна обробка, як спосіб підвищення загальної ефективності обчислень, є одним з перспективних напрямів розвитку обчислювальних систем. Задачі, які необхідно вирішувати при плануванні паралельних обчислень в загальній постановці є NP-повними. Для скорочення часу вирішення таких задач використовують евристичні підходи. Одним з

евристичних підходів до вирішення задач планування є використання генетичних алгоритмів.

Для однозначного розподілу процесів по процесорах необхідно для кожного процесу знати номер процесора, на якому він виконуватиметься. Інформацію про процес виконання програми зручно представляти у вигляді рядки наступного вигляду:

- кількість чисел в рядку (довжина рядка) відповідає кількості процесів;
- кожна позиція рядка відповідає одному процесу, порядковий номер якого дорівнює номеру позиції;
- число у позиції рядка показує, на якому процесорі виконуватиметься даний процес.

Залежно від вибраного критерію ефективності цільова функція може бути або час виконання програми або завантаження системи. Початкова популяція генерується випадковим чином, якщо передбачається побудова нового розкладу. Якщо є варіант розкладу і його необхідно поліпшити, за основу для початкової популяції береться наявний розклад. Селекція проводиться за пропорційною схемою. Число нащадків прямо пропорційно залежить від значення цільової функції для поточного рядка. Для проведення операції мутації в рядку випадковим чином вибирається позиція, число в якій замінюється вибраним випадковим чином числом з інтервалу.

Розроблений алгоритм представляє практичний інтерес при побудові планувальників багатопроцесорних систем. Дослідження алгоритму показали високу ефективність даного підходу порівняно з іншими методами. Додатковою перевагою даного методу є незалежність від структури обчислювальної системи, кількість процесорів в системі є вхідними даними для алгоритму.

5.5. Мурашині алгоритми та генетичне програмування

У останні два десятиліття при оптимізації складних систем дослідники все частіше застосовують природні механізми пошуку найкращих рішень. Ці механізми забезпечують ефективну адаптацію флори і фауни до довкілля впродовж мільйонів років. Останніми роками інтенсивно розробляється науковий напрям Natural Computing - «Природні обчислення», який об'єднує математичні методи з природними механізмами прийняття рішень, а саме:

- Genetic Algorithms - генетичні алгоритми;
- Evolution Programming - еволюційне програмування;
- Neural Network Computing – еволюційне нейромережеві обчислення;
- DNA Computing - ДНК обчислення;
- Cellular Automata - клітинні автомати;
- Ant Colony Algorithms – мурашині алгоритми.

Імітація самоорганізації мурашиної колонії складає основу мурашиних алгоритмів оптимізації - нового перспективного методу природних обчислень. Колонія мурах може розглядатися як багатоагентна система, в якій кожен агент (мурашка) функціонує автономно по дуже простих правилах. На противагу майже примітивній поведінці агентів, поведінка всієї системи виходить на подив розумною. Мурашині алгоритми серйозно досліджуються європейськими вченими з середини 90-х років. Вперше підхід був запропонований бельгійським дослідником Марко Доріго (Marco Dorigo). На сьогодні вже отримані важливі результати мурашиної оптимізації таких складних комбінаторних задач, як: задача комівояжера, задача оптимізації маршрутів вантажівок, задача розфарбовування графа, квадратичної задачі про призначення, оптимізації мережевих графіків, задача календарного планерування і інших. Особливо ефективні мурашині алгоритми при оптимізації процесів в розподілених нестаціонарних системах, наприклад трафіків в телекомунікаційних мережах [50, 78].

Кожен, хто хоч раз в житті спостерігав за мурахами, обов'язково повинен був відмітити: вся діяльність цих комах має яскраво виражене групове забарвлення. Працюючи разом, група мурах здатна затаскати в мурашник шматок їжі або будівельного матеріалу, в 10 разів більше самих працівників. Вчені давно знають про це, але лише останнім часом задумалися про корисне застосування мурав'їнного досвіду в повсякденному житті.

Сам по собі мурашка - досить примітивна істота. Всі його дії, по суті, зводяться до елементарних інстинктивних реакцій на навколишнє оточення і своїх побратимів. Проте декілька мурах разом утворюють складну систему, яку деякі вчені називають «колективним розумом». Тому алгоритми мурав'їнних колоній часто називають алгоритмами «роєвого інтелекту». Наприклад, група мурах прекрасно здатна знаходити найкоротшу дорогу до їжі. Якщо яка-небудь перешкода - палиця, камінь, нога людини - встає на дорозі, браві добувачі швидко знаходять новий оптимальний маршрут.

Принципи поведінки мурах витримали випробування впродовж 100 мільйонів років - саме стільки часу тому мурашки «колонізували» Землю. Вони відносяться до соціальних комах, що живуть усередині деякого колективу - колонії. На Землі близько двох відсотків комах є соціальними, половину з них складають мурав'ї - невеликі істоти масою від 1 до 5 міліграма. Число мурах в одній колонії вагається від 30 штук до декількох мільйонів. На Землі близько 10^{16} мурах із загальною масою, приблизно рівній масі людства.

Які ж механізми забезпечують настільки складну поведінку мурах, і що можна запозичити у цих істот для вирішення своїх глобальних задач? Основу «соціальної» поведінки мурашок складає самоорганізація - множина динамічних механізмів, що забезпечують досягнення системою глобальної мети в результаті низькорівневої взаємодії її елементів. Принциповою особливістю такої взаємодії є використання елементами системи лише локальній інформації. При цьому виключається будь-яке централізоване управління і звернення до глобального образу, що репрезентує систему на зовнішньому світі.

Мурахи використовують два способи передачі інформації: прямий - обмін їжею, мандибулярний, візуальний і хімічний контакти, і непрямий - стігмержи (stigmergy). Стігмержи - це рознесений в часі тип взаємодії, коли один суб'єкт взаємодії змінює деяку частину довкілля, а останні використовують інформацію про її стан пізніше, коли знаходяться в її околиці. Біологічно стігмержи здійснюється через феромон (pheromone) - спеціальний секрет, що відкладається як слід при переміщенні мурав'їв. Феромон - досить стійка речовина, він може сприйматися мурашками декілька діб. Чим вище концентрація феромону на стежці, тим більше мурах по ній рухатиметься. З часом феромон випаровується, що дозволяє мурашкам адаптувати свою поведінку під зміни зовнішнього середовища. Розподіл феромону по простору пересування мурах є свого роду динамічно змінною глобальною пам'яттю мурашника. Будь-яка мурашка у фіксований момент часу може сприймати і змінювати лише один локальний елемент цієї глобальної пам'яті.

Експерименти з Argentine ants, які проводилися Госсом в 1989 і Денеборгом в 1990 році послужили відправною точкою для дослідження роєвого інтелекту. Дослідження застосування отриманих знань для дискретної математики почалися на початку 90-х років XX століття, автором ідеї є Марко Доріго з Університету Брюсселю, Бельгія. Саме він вперше зумів формалізувати поведінку мурах і застосувати стратегію їх поведінки для вирішення задач про найкоротші шляхи. Пізніше за участю Гамбарделли, Тайлларда і Ді Каро були розроблені і багато інших підходів до вирішення складних оптимізаційних задач при допомогою мурашиних алгоритмів.

Ідея мурашиного алгоритму полягає в наступному: дві мурашки з мурашника повинні дістатися до їжі, яка знаходиться за перешкодою. Під час переміщення кожна мурашка виділяє трохи феромону, використовуючи його як маркер. При інших рівних умовах кожна мурашка вибере свою дорогу. Перша мурашка вибирає верхню дорогу, а друга - нижню. Оскільки нижня дорога в два рази коротше верхньої, друга мурашка досягне мети за час t_1. Перша мурашка у цей момент пройде лише половину дороги. Коли одна мурашка

досягає їжі, вона бере один з об'єктів і повертається до мурашника по тій же дорозі. За час t_2 друга мурашка повернеться в мурашник з їжею, а перша мурашка досягне їжі. При переміщенні кожної мурашки на дорозі залишається трохи феромону. Для першої мурашки за час t_0t_2 дорога була покрита феромонами лише один раз. У той же самий час друга мурашка покрила дорогу феромонами двічі. За час t_4 перша мурашка повернулася в мурашник, а друга мурашка вже встигла ще раз сходити до їжі і повернутися. При цьому концентрація феромону на нижній дорозі буде в два рази вище, ніж на верхній. Тому перша мурашка наступного разу вибере нижню дорогу, оскільки там концентрація феромону вища. У цьому і полягає базова ідея мурашиного алгоритму - оптимізація шляхом непрямого зв'язку між автономними агентами.

Розглянемо покроковий опис алгоритму. Передбачимо, що довкілля для мурашок є повний неорієнтований граф. Кожне ребро має вагу, яка позначається як відстань між двома вершинами, сполученими їм. Граф двонаправлений, тому мурашка може подорожувати по грані в будь-якому напрямку.

Імовірність включення ребра в маршрут окремої мурашки пропорційна кількості феромону на цьому ребрі, а кількість феромону, що відкладається, пропорційна довжині маршруту. Чим коротше маршрут тим більше феромону буде відкладено на його ребрах, отже, більша кількість мурашок включатиме його в синтез власних маршрутів. Моделювання такого підходу, що використовує лише позитивний зворотний зв'язок, приводить до передчасної збіжності - більшість мурашок рухаються по локально-оптимальному маршруту. Уникнути цього можна моделюючи негативний зворотний зв'язок у вигляді випару феромону. Причому, якщо феромон випаровується швидко, то це наводить до втрати пам'яті колонії і забуванню хороших рішень, з іншого боку, великий час випарів може привести до здобуття стійкого локального оптимального рішення.

Мурашка. Мурашка - це програмний агент, який є членом великої колонії і використовується для вирішення якої-небудь проблеми. Агент

забезпечується набором простих правил, які дозволяють йому вибирати дорогу в графі. Він підтримує список вузлів, які вже відвідав. Таким чином, мурашка повинна проходити через кожен вузол лише один раз. Дорога між двома вузлами графа, по якому мурашка відвідав кожен вузол лише один раз, називається шляхом Гамільтона.

Вузли в списку «поточної подорожі» розташовуються в тому порядку, в якому мурашка відвідував їх. Пізніше список використовується для визначення протяжності дороги між вузлами. Справжня мурашка під час переміщення по дорозі залишатиме за собою феромон. У алгоритмі агент залишає феромон на ребрах графа після завершення подорожі. Стартова точка, куди поміщається мурашка, залежить від обмежень, що накладаються умовами задачі, оскільки для кожної задачі спосіб розміщення мурашок є визначальним. Або всі вони поміщаються в одну точку, або в різні з повтореннями, або без повторень. На цьому ж етапі задається початковий рівень феромону. Він ініціалізується невеликим позитивним числом для того, щоб на початковому кроці ймовірності переходу в наступну вершину не були нульовими.

Рух мурашок. Рух мурашок ґрунтується на одному простому ймовірнісному рівнянні. Якщо мурашка ще не закінчила дорогу, тобто не відвідала всі вузли мережі, для визначення наступного ребра дороги використовується рівняння

$$P = \frac{\tau(r,u)^{\alpha}\eta(r,u)^{\beta}}{\sum_{k}\tau(r,u)^{\alpha}\eta(r,u)^{\beta}}.$$

Тут $\tau(r,u)$ - інтенсивність ферменту на ребрі між вузлами r та u, $\eta(r,u)$ - функція, яка представляє вимір зворотної відстані для грані, α - вага ферменту, а β - коефіцієнт евристики. Параметри α та β визначають відносну значущість двох параметрів, а також їх вплив на рівняння. Оскільки мурашка подорожує лише по вузлах, які ще не були відвідані (як вказано списком табу), ймовірність розраховується лише для ребер, які ведуть до ще не відвіданих вузлів. Змінна k представляє ці ребра.

Подорож мурашок. Пройдений мурашкою шлях відображується, коли вона відвідає всі вузли графа. Цикли заборонені, оскільки в алгоритм включений список табу. Після завершення довжина дороги може бути підрахована - вона дорівнює сумі довжин всіх ребер, по яких подорожувала мурашка. Рівняння показує кількість феромону, який був залишений на кожному ребрі шляху для мурашки k. Змінна Q є константою.

$$\Delta\tau_{ij}^{k}(t) = \frac{Q}{L^{k}(t)}.$$

Результат рівняння є засобом виміру шляху, - коротка дорога характеризується високою концентрацією феромону, а довша дорога - нижчою. Потім отриманий результат використовується в іншому рівнянні, аби збільшити кількість феромону уздовж кожного ребра пройденої мурашкою дороги.

$$\tau_{ij}(t) = \Delta\tau_{ij}(t) + (\tau_{ij}^{k}(t)\rho).$$

Важливо, що дане рівняння застосовується до всього шляху, при цьому кожне ребро позначається феромоном пропорційно довжині дороги. Тому слід діждатися, поки мурашка закінчить подорож і тільки потім відновити рівні феромону, інакше дійсна довжина дороги залишиться невідомою. Константа ρ - значення між 0 та 1.

Випар феромону. На початку шляху в кожного ребра є шанс бути обраним. Аби поступово видалити ребра, які входять в гірші шляхи графа, до всіх ребер застосовується процедура випару феромону. Використовуючи константу ρ з попереднього рівняння, ми отримуємо нове рівняння

$$\tau_{ij}(t) = \tau_{ij}(t)(1-\rho).$$

Тому для випару феромону використовується зворотний коефіцієнт оновлення шляху.

Повторний запуск. Після того, як дорога мурашки завершена, ребра оновлені відповідно до довжини шляху і стався випар феромону на всіх ребрах, алгоритм запускається повторно. Список табу очищається, і довжина дороги обнуляється. Мурашкам дозволяється переміщатися по графові, засновуючи

вибір ребра на першому рівнянні. Цей процес може виконуватися для постійної кількості шляхів або до моменту, коли впродовж декількох запусків не було відмічено повторних змін. Потім визначається кращий шлях, який і є рішенням.

Розглянемо функціонування алгоритму на прикладі сценарію з двома мурашками. На рисунку 5.40 показаний приклад з двома ребрами між двома вузлами (V_0 та V_1). Кожне ребро ініціалізується і має однакові шанси на те, аби бути обраним.

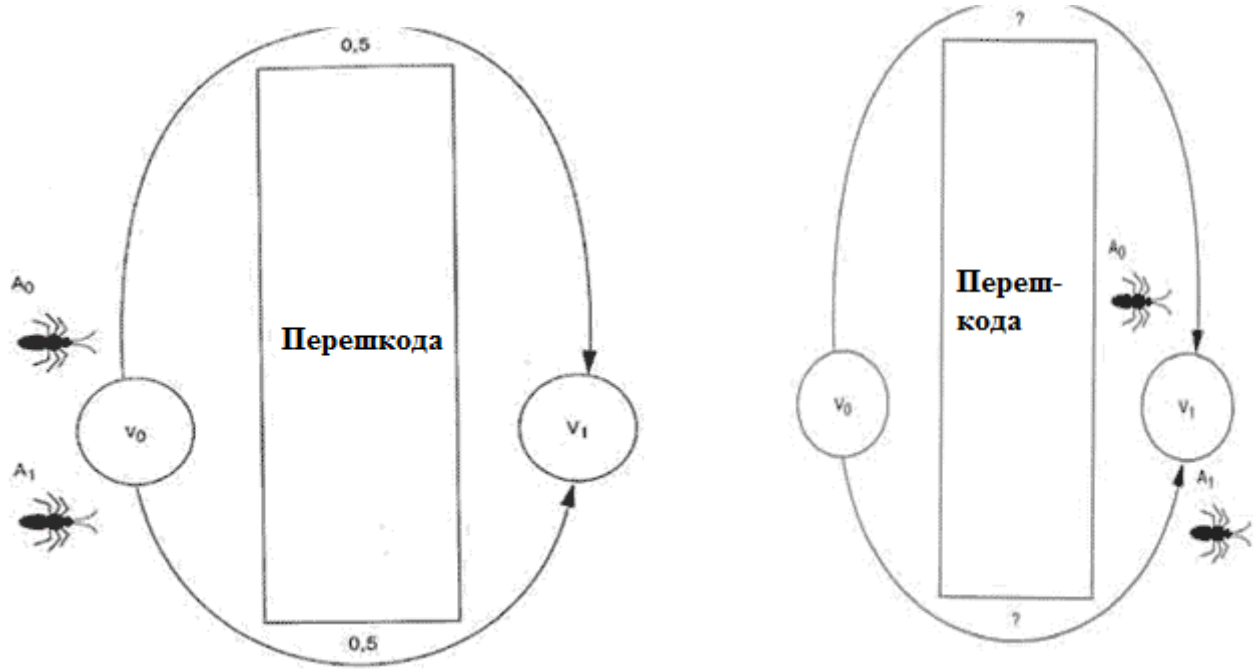


Рис. 5.40. Сценарій з двома мурашками.

Дві мурашки знаходяться у вузлі V_0 і позначаються як A_0 та A_1. Оскільки ймовірність вибору будь-якої дороги однакова, то в цьому циклі ігнорується рівняння вибору дороги. Дані для задачі:

- число пройдених кроків: для A_0 - 20, для A_1 - 10;
- рівень феромону (Q /пройдену відстань): для A_0 - 0.5, A_1 - 1.0;
- $\rho = 0.5,\ \alpha = 0.3,\ \beta = 1.0.$

Кожна мурашка обирає свій шлях (мурашка A_0 йде по верхній дорозі, а мурашка A_1 - по нижній). Мурашка A_0 зробив 20 кроків, а мурашка A_1, - лише 10. Згідно другого рівняння розраховуємо кількість феромону, яке має бути

«нанесене». Далі по третьому рівнянню розраховується кількість феромону, яка буде застосовано. Для мурашки A_0 результат складає $0.1+(0.5*0.6)=0.4$ Для мурашки A1 результат складає $0.1+(0.1*0.6)=0.7$. За допомогою четвертого рівняння визначається, яка частина феромону випарується і, відповідно, скільки залишиться. Результати (для кожного шляху відповідно) складають

$$0.4*(1.0-0.6)=0.16$$
$$0.7*(1.0-0.6)=0.28.$$

Ці значення представляють нову кількість феромону для кожного шляху (верхнього і нижнього, відповідно). Тепер перемістимо мурашку назад у вузол V_0 і скористаємося першим імовірнісним рівнянням вибору шляху, аби визначити, яку дорогу повинні обрати мурашки. Вірогідність того, що мурашка обере верхню дорогу (представлену кількістю феромону 0.16), складає

$$\frac{(0.16)^{3.0}*(0.5)^{1.0}}{(0.16)^{3.0}*(0.5)^{1.0}+(0.28)^{3.0}*(1.0)^{1.0}}=\frac{0.002048}{0.024}=P(0.085).$$

Вірогідність того, що мурашка обере нижню дорогу (представлену кількістю феромону 0.28) складає

$$\frac{(0.28)^{3.0}*(1.0)^{1.0}}{(0.16)^{3.0}*(0.5)^{1.0}+(0.28)^{3.0}*(1.0)^{1.0}}=\frac{0.021952}{0.024}=P(0.915).$$

При зіставленні двох імовірностей обидві мурашки виберуть нижню дорогу, яка є оптимальною.

Слід зазначити, що для алгоритму мурашиної колонії необхідно вказати: закон виділення феромону, закон випару феромону, кількість агентів, місця розміщення. Всі ці характеристики вибираються з врахуванням особливості задачі на основі експериментальних досліджень.

За останній час було запропоновано декілька метаевристичних моделей мурашиної колонії. Серед них три найбільш успішні:

- Ant system (Dorigo, 1996);
- Ant colony system (ACS) (Dorigo & Gambardella, 1997);
- MAX-MIN Ant system (MMAS) (Stutzle & Hoos, 2000).

Розглянемо застосування мурашиних алгоритмів до рішення вже відомої задачі комівояжера. Дослідимо реалізацію чотирьох складових самоорганізації мурашок при оптимізації маршруту комівояжера. Багатократність взаємодії реалізується ітераційним пошуком маршруту комівояжера одночасно декількома мурашками. При цьому кожна мурашка розглядається як окремий, незалежний комівояжер, вирішуючий свою задачу. За одну ітерацію алгоритму кожна мурашка здійснює повний маршрут комівояжера. Позитивний зворотний зв'язок реалізується як імітація поведінки мурашок типа «залишення слідів - переміщення по слідах». Чим більше слідів залишено на тропі - ребрі графа в задачі комівояжера, тим більше мурашок пересуватиметься по ній. Для задачі комівояжера позитивний зворотний зв'язок реалізується наступним стохастичним правилом: ймовірність включення ребра графа в маршрут мурашки пропорційна кількості феромону на ньому. Вживання такого імовірнісного правила забезпечує реалізацію і іншої складовій самоорганізації - випадковості. Кількість феромону, що відкладається мурашкою, на ребрі графа обратно пропорційно до довжини маршруту. Чим коротше маршрут, тим більше феромону буде відкладено на відповідних ребрах графа і тим більше мурашок використовуватиме їх при синтезі своїх маршрутів. Відкладений на ребрах феромон виступає як підсилювач, він дозволяє хорошим маршрутам зберігатися в глобальній пам'яті мурашника. Ці маршрути можуть бути покращені на подальших ітераціях алгоритму.

Використання лише позитивного зворотного зв'язку приводить до передчасної збіжності рішень - до випадку, коли всі мурашки рухаються одним і тим же субоптимальним маршрутом. Для уникнення цього використовується негативний зворотний зв’язок - випар феромону.

Для кожної мурашки перехід з міста i в місто j залежить від трьох складових: пам'яті мурашки (tabu list), видимості і віртуального сліду феромону. Tabu list - це список відвіданих мурашкою міст, заходити в які ще раз не можна. Використовуючи цей список, мурашка гарантовано не попаде в одне і те ж місто двічі. Позначимо через, J_{ik} список міст, які ще необхідно

відвідати мурашці k, що знаходиться в місті i. Видимість - величина, зворотна відстані: $\eta_{ij} = 1/D_{ij}$, де D_{ij} - відстань між містами i та j. Видимість - це локальна статична інформація, що висловлює евристичне бажання відвідати місто j з міста i, - чим ближче місто, тим більше бажання відвідати його. Віртуальний слід феромону на ребрі (i, j) представляє підтверджене мурашиним досвідом бажання відвідати місто j з міста i. На відміну від видимості слід феромону є глобальнішою і динамічнішою інформацією - вона змінюється після кожної ітерації алгоритму, відображаючи придбаний мурашками досвід. Кількість віртуального феромону на ребрі (i, j) на ітерації t позначимо через $\tau_{ij}(t)$. Подальша робота алгоритму аналогічна наведеному вище.

Загальна кількість мурашок в колонії залишається постійною впродовж виконання алгоритму. Звичайне число мурашок призначають рівним кількості міст - кожна мурашка починає маршрут зі свого міста. Для поліпшення часових характеристик мурашиного алгоритму вводять так званих елітних мурашок. Елітна мурашка підсилює ребра найкращого маршруту, знайденого з початку роботи алгоритму. Кількість феромону, що відкладається на ребрах найкращого поточного маршруту T^+ приймається рівним Q/L^+, де L^+ - довжина маршруту T^+. Цей феромон спонукає мурашок до дослідження рішень, що містять декілька ребер найкращого на даний момент маршруту T^+. Якщо в мурашнику є e елітних мурах, то ребра маршруту T^+ отримуватимуть загальне посилення $\Delta\tau_e = eQ/L^+$.

Практична реалізація вказаного алгоритму може бути виконана різними програмними засобами. Зокрема, українським вченим С. Д. Штовбою запропонован варіант його застосування засобами пакету MATLAB. Для задачі з 29 населеними пунктами в Баварії «Bays - 29» алгоритм без елітних мурашок після 100 ітерацій знайшов оптимальний маршрут завдовжки 2020 лише в одному випадку з п'яти. Рішення можна поліпшити простим збільшенням кількості ітерацій до 1 - 2 тисяч. Довжини маршрутів мурашок на одній ітерації

відрізняються трохи, тому для прискорення знаходження оптимуму необхідно штучно підсилювати найкращі поточні рішення за допомогою елітних мурашок.

Еволюція маршрутів комівояжера, знайдених алгоритмом з трьома елітними мурашками, показана на рис. 5.41.

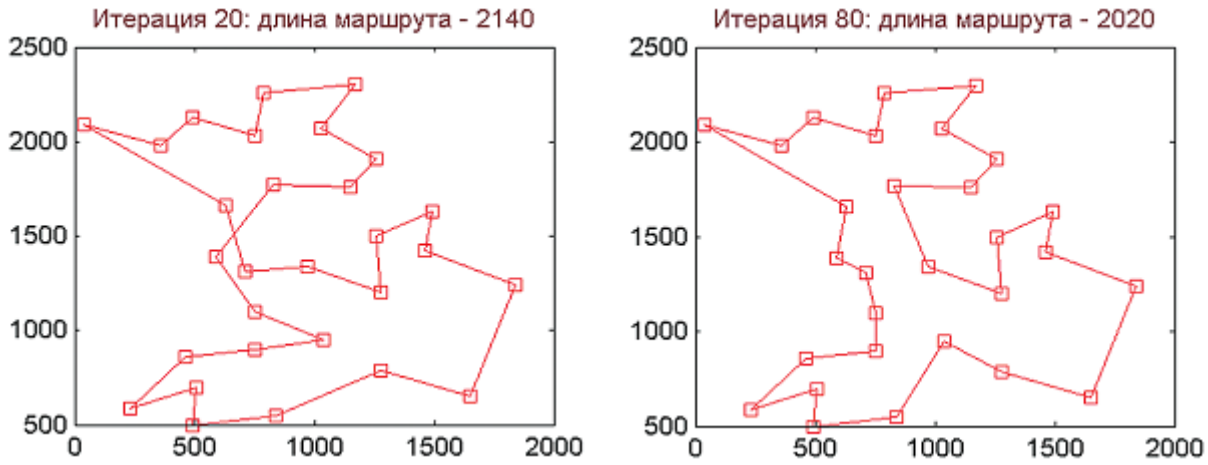


Рис.5.41. Еволюція маршрутів комівояжера.

В порівнянні з точними методами, наприклад динамічним програмуванням або методом гілок і границь, мурашиний алгоритм знаходить близькі до оптимуму рішення за значно менший час навіть для задач невеликої розмірності. Час оптимізації мурашиним алгоритмом є поліноміальною функцією від розмірності $F(t,n^2,m)$, тоді як для точних методів залежність експоненціальна.

Мурашиний алгоритм оптимізації маршруту комівояжера після незначних модифікацій може використовуватися для вирішення різних комбінаторних задач: квадратичної задачі про призначення (Quadratic Assignment Problem), задачі про оптимізацію маршрутів вантажівок (Vehicle Routing Problem), задачі календарного планування (Job-Shop Schedule Planing), задачі розфарбовування графа (Graph Coloring Problem) та ін. Важливі результати були отримані для нестаціонарних систем із змінними в часі параметрами, наприклад, для розрахунків телекомунікаційних і комп'ютерних

мереж. Мурашиний алгоритм застосовувався для розробки оптимальної структури знімальних мереж GPS в рамках створення високоточних геодезичних і знімальних технологій.

Фахівці з американської дослідницької компанії Pacific Northwest National Laboratory знайшли новий підхід до вивчення безпеки комп'ютерних мереж. Для боротьби з вірусами, троянами і комп'ютерними черв'яками вони вирішили використовувати мурашині алгоритми. За допомогою спеціалізованої програми, дослідники намагаються знайти «мережеві аномалії». Їх програма використовує розподілені по комп'ютерних мережах сенсори, що безперервно збирають дані. Немов мурахи, які передають своїм родичам інформацію про їду або небезпеку за допомогою запахів, ці сенсори діляться зібраною інформацією один з одним. Таким чином, програма може визначити своєрідні мережеві аномалії, що сигналізують про можливу небезпеку, наприклад про масштабне зараження мережі. Сенсори бувають різної спрямованості - одні можуть збирати дані про надмірне завантаження центрального процесора комп'ютерів, а інші - перевіряти мережевий трафік. Також є «вартові» - спеціальні блоки програми, що аналізують інформацію, отриману від всіх сенсорів - мурашок. Хоча інноваційний антивірусний комплекс знаходиться на ранній стадії розробки, вже зараз він здатний виявляти деяких комп'ютерних черв'яків.

Перспективними напрямами поліпшення мурашиних алгоритмів є різні адаптації параметрів з використанням бази нечітких правил і їх гібридизація, наприклад, з генетичними алгоритмами. Як варіант, така гібридизація може полягати в обміні через певні проміжки часу поточними найкращими рішеннями.

Достоїнствами мурашиних алгоритмів є:

- в порівняння з GAs (Genetic Algorithms) мурав’їнні алгоритми спираються на пам'ять всієї колонії замість пам'яті лише про попереднє покоління і менше схильні до неоптимальних початкових рішень (із-за випадкового вибору шляху);

- можуть використовуватися в динамічних додатках (швидко адаптуються до змін);
- мають важливу практику застосовування до множини різних задач.

Недоліки таких алгоритмів вважають:

- ускладнений теоретичний аналіз в результаті послідовності випадкових (незалежних) рішень і зміни розподілу вірогідності при ітераціях;
- збіжність алгоритму гарантується, але час збіжності не визначений;
- зазвичай необхідне вживання додаткових методів таких, як локальний пошук;
- сильно залежать від параметрів настройки, які підбираються лише виходячи з експериментів.

Мурашині алгоритми засновані на імітації самоорганізації соціальних комах за основі використання динамічних механізмів, за допомогою яких система досягає глобальної мети в результаті локальної низькорівневої взаємодії елементів. Мурашині алгоритми забезпечують рішення також інших комбінаторних задач не гірше за загальні метаевристичні технології оптимізації і деякі проблемно-орієнтовані методи. Важливою властивістю мурашиних алгоритмів є неконвергентність: навіть після великого числа ітерацій одночасно досліджується множина варіантів рішення, внаслідок чого не відбувається тривалих часових задержок в локальних екстремумах. Все це дозволяє успішно застосовувати такі алгоритмі для вирішення складних задач інтелектуального аналізу даних.

Генетичне програмування (genetic programming, GP) - одна з найзручніших і універсальних методик вирішення задач, що встають перед аналітиками. Воно застосовується для вирішення широкого круга проблем: символьної регресії (symbolic regression), аналізу даних (data mining), оптимізації і дослідження поведінки потомства (emergent behavior), що з'являється, в біологічних співтовариствах.

Ідею генетичного програмування (ГП) вперше запропонував Дж. Коза в 1992 році, спираючись на концепцію генетичних алгоритмів. Генетичне

програмування - це розширення генетичної моделі навчання в область програмного забезпечення. Його об'єктом на відміну від генетичних алгоритмів є не символьні рядки фіксованої довжини, що кодують можливі рішення проблеми, а власне програми, виконуючи які і отримують різні варіанти рішення задачі. У генетичному програмуванні програми представляються у вигляді дерева граматичного розбору, а не у вигляді рядків коду, тому в ГП всі операції виконуються не над рядками, а над *деревами*. При цьому використовуються такі ж оператори, як і в генетичному алгоритмі: селекція, схрещування і мутація [31, 66].

У ГП хромосомами є *програми*. Програми представлені як дерева з *функціональними* (проміжними) і *термінальними* (кінцевими) елементами. Термінальними елементами є константи, дії і функції без аргументів, функціональними - функції, що використовують аргументи. Для приклада можна розглянути функцію $y = 2 + (4/7)x$, представлену на рис. 5.42. Термінальні елементи $T = \{2, x, 4, 7\}$, функціональні $F = \{+, *, /\}$.

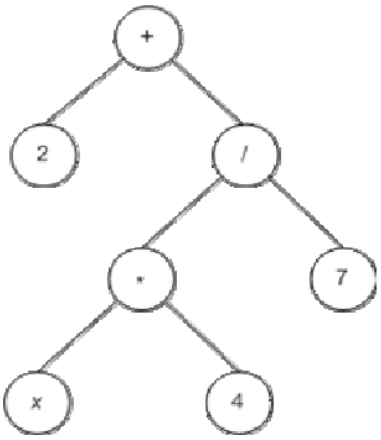


Рис. 5.42. Деревовидне представлення функції.

Для того, щоб застосувати ГП до якої-небудь проблеми, необхідно визначити:

- множину термінальних елементів;
- множину функціональних елементів;
- міру пристосованості;

- параметри, контролюючі еволюцію;
- критерій останову еволюції.

Генетичні програми не пишуться програмістами, а створюються в результаті наступного ітераційного покрокового процесу.

1. Генерується початкова популяція програм, що є випадковими композиціями функцій (арифметичних, логічних та ін. операцій) і терміналів (змінних і констант), узятих з множини функціональних і термінальних елементів, що відносяться до вирішуваної проблеми. Якщо ми збираємося вивести програму, яка управляє транспортною логістикою, то до набору терміналів увійдуть наступні змінні - відстань до об’єкта, швидкість і вантажопідйомність, тип транспортних засобів і так далі. Набор функцій в цьому випадку включатиме різні математичні операції, як прості (множення, ділення, віднімання, складання), так і складніші.

2. Кожна програма виконується і їй привласнюється значення пристосованості, відповідне тому, наскільки добре вона вирішує задачу.

3. Створюється нова популяція комп'ютерних програм за рахунок:

- копіювання в неї найкращих програм старої популяції;
- створення нових програм за допомогою мутацій (випадкової зміни функцій і терміналів або навіть цілих піддерев);
- створення нових програм за допомогою схрещування тих, що існують. Операція схрещування реалізується за рахунок обміну піддерев двох хромосом, вибраних випадково (відповідно до їх пристосованості).

4. Всі програми знову виконуються і цикл повторюється до тих пір, поки не буде отриманий необхідний результат.

Слід зазначити, що алгоритм роботи ГП такий же, як і ГА: селекція, схрещування і мутація. Проте оскільки ГП оперує над деревами, а не над рядками, то оператори схрещування і мутації мають відмінності.

Оператор схрещування працює наступним образом: вибираються випадкові частини батьківських дерев, і ці частини міняються місцями (рис. 5.43).

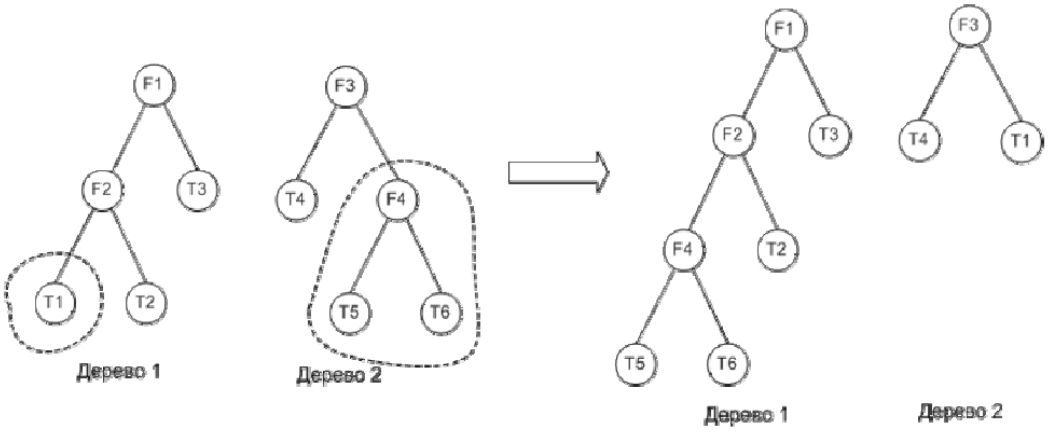


Рис. 5.43. Схрещування двох дерев.

В якості особливості необхідно відзначити, що в ГП розмір хромосоми міняється. Аби запобігти надмірному розростанню дерева, вводять максимальну кількість функціональних елементів в дереві або максимальну глибину дерева. Проте при операції схрещування можлива ситуація, коли при схрещуванні двох дерев вийде одне з дерев, що перевершує заданий ліміт. В цьому випадку замість конфліктного дерева копіюється батьківське дерево (рис. 5.44).

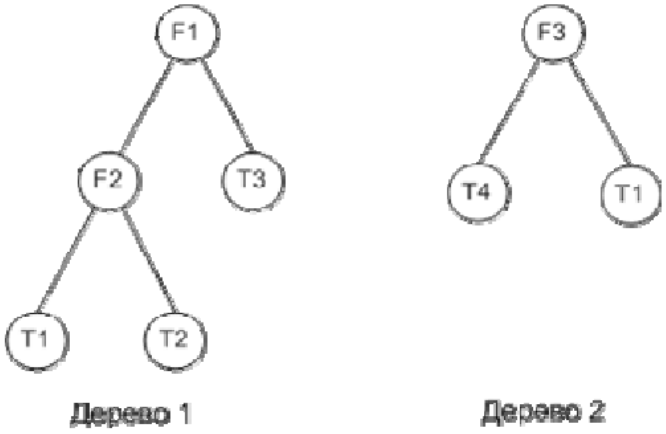


Рис. 5.44. Розв’язання конфліктної ситуації попереднього оператора схрещування при максимальній глибині дерева, рівній трьом.

Також слід зазначити, що часто при вживанні оператора схрещування з'являються *інтрони* - ділянки коди, що нічого не роблять (наприклад, обчислюють вирази вигляду $x = x * 1$). На перший погляд, час, який ГП витрачає на збільшення і розвиток інтронів, проходить даремно. З іншого боку,

інтрони допомагають зберігати від руйнування хороші блоки для майбутніх поколінь, що підвищує шанси на здобуття ще ефективніших особин.

Мутація. Оператор мутації випадково видаляє частину дерева і замінює її новим деревом (рис. 5.45).

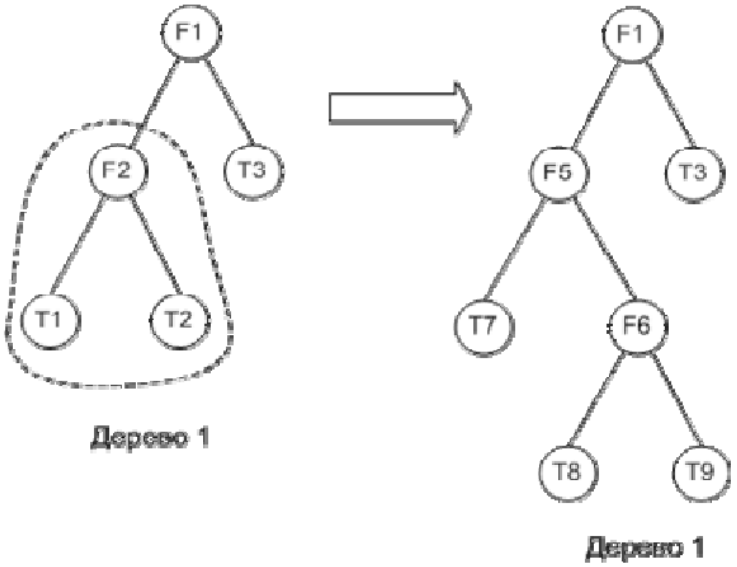


Рис. 5.45. Дія оператора мутації.

Зупинимося на деяких напрямах застосування генетичного програмування.

Класифікуючі системи (classifier systems) - дуже цікавий напрям, що вивчає питання створення самонавчальних машин. Один з його творців Дж. Холланд запропонував для цих цілей використовувати когнітивну систему, здатну класифікувати стан довкілля і відповідно реагувати на цей стан. Що необхідне для побудови такої системи? Вочевидь (1) - середовище; (2) - рецептори, які повідомлятимуть систему про те, що відбувається; (3) - ефектори, які дозволять нашій системі маніпулювати середовищем; і (4) - власне систему, «чорний ящик», який має (2) і (3) і «живе» усередині (1). Такі системи часто називають «аніматами» (animal + robot = animat). Основна ідея класифікуючих систем - почавши з нульового знання, використовуючи випадково згенеровану класифікуючу популяцію, дозволити системі створити свою програму за допомогою індукції. Вона приводить вхідний потік до

деякого шаблону, який дозволяє анімату класифікувати свій поточний стан/контекст і реагувати відповідним чином [62].

Системи зі складною поведінкою. Програмні і апаратні системи, а також їх окремі елементи, часто мають складну поведінку (наприклад, пристрої управління і мережеві протоколи). Останнім часом для опису об’єктів із складною поведінкою в програмуванні пропонується використовувати автоматний підхід. Еволюційним моделям обчислень у вигляді кінцевих автоматів було присвячено багато досліджень. При цьому більшість авторів займалися оптимізацією автоматів - розпізнавателей (parsers) і перетворювачів (transducers). Відповідно до автоматного підходу об'єкт представляється у вигляді автоматизованого об'єкту - сукупності управляючого автомата і об'єкту управління. При цьому вся «логіка поведінки» виявляється зосередженою в автоматі, що управляє. Побудова автомата, що управляє, зазвичай вимагає від розробника набагато більше зусиль і породжує більше помилок, чим реалізація об'єкту управління. Для деяких задач евристична побудова автомата або неможлива, або досить складна. Крім того, навіть для простих задач автомат, побудований вручну, часто є неоптимальним. Цю проблему можна вирішувати методами генетичного програмування. При цьому автомати генеруються автоматично, що істотно спрощує побудову автоматних програм і дозволяє підвищити рівень автоматизації їх проектування.

До перспективних напрями розвитку ГП слід віднести роботи по так званих автоматично визначаємих функціях (ADF), ідея яких полягає в підвищенні ефективності ГП за рахунок модульної побудови програм, що складаються з головної програми і ADF-модулів, які генеруються в ході моделювання еволюції. До початку еволюції визначається структура програми, число ADF-модулів і параметри кожної ADF. Всі вершини даної структури мають свій номер, список аргументів включає окремі локальні змінні, які визначаються при виклику ADF. Задання функції головної програми завершує установку загальної програми. Налаштування ADF (їх число, аргументи і тому

подібне) залежить від вирішуваної задачі, наявних обчислювальних ресурсів і попереднього досвіду.

Іншим перспективним напрямом є реалізація ГП на трансп'ютерних обчислювальних системах.

Тест

1. Застосування еволюційних методів в інтелектуальному аналізі даних передбачає, що …

а) моделювання складного об'єкту замінюється моделюванням його генетичних параметрів;

б) моделювання складного об'єкту замінюється моделюванням його популяції;

в) моделювання складного об'єкту замінюється моделюванням його біологічних характеристик;

г) моделювання складного об'єкту замінюється моделюванням його еволюції.

2. Під поняттям «еволюційні обчислення» Ви розумієте:

а) сукупність алгоритмів пошуку, оптимізації або навчання, заснованих на формалізованих принципах штучного інтелекту;

б) сукупність алгоритмів пошуку, оптимізації або навчання, заснованих на формалізованих принципах природного біологічного процесу;

в) сукупність алгоритмів пошуку, оптимізації або навчання, заснованих на формалізованих принципах природного еволюційного процесу.

3. Якщо для аналізу даних Ви вирішили застосувати еволюційний алгоритм на основі методу групового обліку аргументів, то розумієте, що його сутністю є …

а) розробка сімейства імітаційних алгоритмів для моделювання мультипараметричних даних;

б) розробка сімейства індуктивних алгоритмів для моделювання мультипараметричних даних;

в) розробка сімейства індуктивних алгоритмів для моделювання біологічних даних.

4. Генетичний алгоритм – це …

а) модель еволюції в природі;

б) модель революції в природі;

в) модель прогресу в природі.

5. Хромосомою при застосуванні генетичних алгоритмів називають:

а) властивості об'єктів представлені значеннями параметрів, що об'єднуються в запис;

б) властивості об'єктів представлені значеннями параметрів, що об'єднуються в формулу;

в) властивості об'єктів представлені значеннями функцій, що об'єднуються в запис.

6. Популяція – це …

а) множина об'єктів, до яких відносяться гени;

б) множина об'єктів, до яких відносяться хромосоми;

в) множина об'єктів, до яких відносяться алелі.

7. Під імовірнісним вибором батьків Ви розумієте …

а) імітацію динамічних принципів;

б) імітацію імовірнісних принципів;

в) імітацію генетичних принципів.

8. При використанні генетичних алгоритмів використовується термін «генотип», який означає …

а) часткову генетичну модель особини і відповідає підструктурі в ГА;

б) повну генетичну модель особини і відповідає структурі в ГА;

в) початкову генетичну модель особини і відповідає популяції в ГА.

9. При використанні генетичних алгоритмів використовується термін «фенотип», який означає …

а) сукупність зовнішніх спостережуваних ознак і відповідає вектору в просторі параметрів;

б) сукупність внутрішніх спостережуваних ознак і відповідає вектору в просторі параметрів.

в) сукупність зовнішніх спостережуваних ознак і відповідає вектору в просторі цілей.

10. У генетичному алгоритмі етап формування початкової популяції передбачає

а) отримання скінченого набору будь-яких рішень задачі;

б) отримання скінченого набору оптимальних рішень задачі;

в) отримання скінченого набору допустимих рішень задачі.

11. З якою метою в генетичному алгоритмі використовується функція пристосованості?

а) щоб задати міру якості для кожного індивіда в просторі пошуку;

б) щоб задати міру кількості для кожного індивіда в просторі пошуку;

в) задати міру ймовірності для кожного індивіда в просторі пошуку.

12. На якому етапі виконання генетичного алгоритму застосовується функція пристосованості?

а) на етапі формування початкової популяції;

б) на етапі оцінки особин популяції;

в) на етапі схрещування;

г) на етапі мутації.

13. На етапі вибору генетичного алгоритму Ви виконуєте …

а) вибір шляху пошуку рішень;

б) вибір оптимальної популяції;

в) вибір оптимальної хромосоми;

г) вибір батьківської пари.

14. На Ваш погляд селекція в генетичному алгоритмі полягає в тому, що …

а) батьками можуть стати лише ті особини, значення пристосованості яких не менше нуля;

б) батьками можуть стати лише ті особини, значення пристосованості яких значно більше нуля;

в) батьками можуть стати лише ті особини, значення пристосованості яких не менше порогової величини.

15. В генетичному алгоритмі оператори схрещування Ви використовуєте для …

а) здобуття оптимальних особин-нащадків;

б) здобуття нових особин-нащадків;

в) здобуття будь-яких особин-нащадків.

16. В генетичному алгоритмі мутація є …

а) процес випадкових модифікацій нащадків;

б) процес випадкових модифікацій батьків;

в) процес випадкових модифікацій популяції.

17. На етапі формування нового покоління Ви вирішуєте проблему …

а) які з нових особин увійдуть до наступного покоління;

б) що робити з предками нових особин;

в) які з нових особин увійдуть до наступного покоління і що робити з їх предками.

18. Генетичний алгоритм завершує свою роботу в випадку …

а) коли пройде задане число поколінь;

б) коли виконається критерій останову;

в) коли отримане оптимальне рішення.

19. Якщо для аналізу даних Ви застосовуєте мурашині алгоритми, то розумієте, що в їх основі лежить …

а) імітація руху мурашиної колонії;

б) біологічна діяльність мурашиної колонії;

в) імітація самоорганізації мурашиної колонії.

20. На Ваш погляд основу генетичного програмування складає …

а) еволюція генетичних алгоритмів, виконуючи які і отримують різні варіанти рішення задачі;

б) еволюція програм, виконуючи які і отримують різні варіанти рішення задачі;

в) еволюція мурашиних алгоритмів, виконуючи які і отримують різні варіанти рішення задачі.

Розділ 6.

НЕЧІТКІ МЕТОДИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

6.1. Концепція нечітких обчислень

Нечіткі обчислення є основою нового наукового напряму, названого «м'які обчислення» або «обчислювальний інтелект», який сформувався в останні 15 років, і включає також нейрообчислення, еволюційні і генетичні алгоритми, міркування на основі свідоцтв, мережі довіри та інші [20, 34, 56].

Поняття «м'які обчислення» було введене основоположником нечіткої логіки Л. Заде на семінарі в 1994 році, як консорціум обчислювальних методологій, які колективно забезпечують основи для розуміння, конструювання і розвитку інтелектуальних систем, зокрема, систем інтелектуального аналіза даних. Основоположна відмінність м'яких обчислень від традиційних жорстких обчислень - пристосованість до роботи з неточними, невизначеними або частково істинними даними, що виражається в «допустимому ставленні до неточності, невизначеності і часткової істинності для досягнення зручності маніпулювання, робастності, низькій вартості рішення і кращої згоди з реальністю» [14, 49, 52].

М'які обчислення не гарантують, що знайдене рішення буде оптимальним або буде досягнутий глобальний екстремум за прийнятний час. Проте вони можуть застосовуватися для пошуку допустимого рішення задачі за «достатньо короткий час». В рамках м'яких обчислень кожна з методологій має відмінності у використанні. Зокрема, нечітка логіка працює з неточністю, зернистою структурою (гранульованою) інформації, наближеними міркуваннями і обчисленнями із словами. Слід відмітити, що поняття нечіткості цілком узгоджується з нашими інтуїтивними уявленнями про навколишній світ. Велика частина понять, які ми використовуємо, за своєю природою нечіткі і

розмиті і спроба загнати їх в рамки загальної логіки приводить до неприпустимих спотворень.

Незважаючи на зовнішню простоту і природність базових понять нечітких обчислень, знадобилося більше п'яти років, щоб побудувати і довести комплекс постулатів і теорем, які роблять логіку логікою, а алгебру - алгеброю. Паралельно з розробкою теоретичних основ нової науки опрацьовувалися різні можливості її практичного застосування. І в 1973 р. ці зусилля увінчалися успіхом - вдалося показати, що нечіткі обчислення можуть бути покладені в основу нового покоління інтелектуальних систем управління. Практично відразу після виходу в світ фундаментальних робіт по нечітким обчисленням одна невелика фірма з Данії застосувала викладені в них принципи для удосконалення системи управління складним виробничим процесом. Результат, що називається, перевершив всі очікування - через чотири роки прибутки від впровадження нової системи обчислювалися сотнями тисяч доларів.

Цілком природно, що військові зацікавились таким перспективним інструментом - і на початку 80-х років в Японії, а потім і в США в були розгорнені комплексні роботи по використанню нечітких обчислень в різних оборонних проектах. Одним з найбільш вражаючих результатів стало створення мікропроцесора на основі нечіткої логіки (fuzzy-chip), здатного автоматично вирішувати відому «задачу про собаку, що наздоганяє кота». Зрозуміло, в ролі кота виступала міжконтинентальна ракета супротивника, а в ролі собаки - мобільна зенітна ракета, дуже легка для установки на неї громіздкої традиційної системи управління. Між іншим, згодом ті ж методи нечіткої логіки дозволили вирішити і обернену задачу - розробити маневри для ефективного відходу від антиракет. Перший успіх окрилив військових і нечіткі обчислення упевнено зайняли своє місце серед стратегічно важливих наукових дисциплін. Виникла парадоксальна ситуація - офіційно не визнана американською академічною наукою нечітка логіка в той же час увійшла до переліку передових технологій, заборонених комітетом COCOM (Комітет з контролю над експортом) до експорту із США.

Проте основні результати використання нечітких обчислень в прикладних задачах були отримані не військовими, а промисловцями, і не в США, а в Японії. До 1990 року з'явилося близько 40 патентів, що відносяться до нечітких обчислень (з них 30 японських). Сорок вісім японських компаній утворили спільну лабораторію LIFE (Laboratory for International Fuzzy Engineering). Японський уряд фінансував 5-річну програму по «нечіткій логіці». Вона включила 19 проектів: від систем оцінки глобального забруднення атмосфери і передбачення землетрусів до АСУ заводських цехів і складів. В результаті з'явилися ряд нових масових мікрочіпів, заснованих на «нечітких обчисленнях». Японці довели практичне втілення нечітких обчислень до досконалості. Вони є основою автоматичних прокатних станів, інтелектуальних складів і «безлюдних виробництв». Проте, мабуть, більш вражаючим виглядає використання нечітких обчислень в дешевих виробах масового ринку - пилососах, відеокамерах, мікрохвильових печах і тому подібне. Піонером у застосуванні нечіткої логіки в побутових виробах виступила фірма Matsuhita. У лютому 2001 р. вона анонсувала першу «інтелектуальну» пральну машину, в системі управління якої поєднувалися нечітка логіка і нейронна мережа. Автоматично визначаючи нечіткі вхідні чинники (об'єм і якість білизни, рівень забрудненості, тип порошку і так далі), пральна машина безпомилково вибирала оптимальний режим прання з 3800 можливих. А через пару років використання нечіткої логіки в японській побутовій техніці стало повсюдним.

Паралельно з використанням нечітких обчислень в системах управління робилися енергійні зусилля по створенню на їх основі нового покоління експертних систем. В той час М. Земанковою (Zemankova) було закладено основу теорії нечітких баз даних, а нечітку експертну систему Фудзи-банка, що приносить до 700000 доларів США на місяць на короткостроковій біржовій грі, створила Сизуко Ясунобу (Chizuko Yasunobu). При цьому така експертна система, що управляє діями «електронного трейдера» Fuji Bank, складається всього з 200 правил. Нечіткі експертні системи, окрім своєї основної переваги - кращою адаптацією до умов реального світу, володіють ще двома

достоїнствами в порівнянні з традиційними. По-перше, вони вільні від т.з. «циклічних блокувань» при побудові висновків. По-друге, різні бази нечітких правил можна з легкістю об'єднувати, що рідко удається в звичайних експертних системах. Існують багаточисельні приклади експертних систем (переважно з області промислової діагностики і медицини), які засновані на концепції нечітких обчислень.

Заслуговує на увагу досвід використання нечітких обчислень у фінансовій сфері. Для вирішення складних задач прогнозування різних фінансових індикаторів банкіри і фінансисти використовують дорогі комплексні системи, до складу яких входять і нечіткі обчислення. Початок цьому процесу поклала японська фінансова корпорація Yamaichi Securuties. Задавшись метою автоматизувати гру на ринку коштовних паперів, ця компанія залучила до роботи близько 30 фахівців з штучного інтелекту. У першу версію системи, завершену на початку 1990 р., увійшли 600 нечітких правил - втілення досвіду десяти провідних брокерів корпорації. Перш ніж зважитися на використання нової системи в реальних умовах, її протестували на дворічній вибірці фінансових даних (1987 - 1989 рр.). Система з блиском витримала випробування. Особливий подив екзаменаторів викликало те, що за тиждень до настання біржового краху (знаменитого «Чорного понеділка» на токійській біржі в 1988 р.) система розпродала весь пакет акцій, що звело збиток практично до нуля. Чи треба говорити, що після цього питання про доцільність використання нечіткої логіки у фінансовій сфері вже не піднімалося.

Природно, що успіх фінансистів не міг не зачепити промислові корпорації США. Вони вельми занепокоїлися втратою стратегічної ініціативи і явними успіхами японців. «Нечітка логіка» привернула їх пильну увагу. Такі корпорації як «Motorola», «General Electric», «Otis Elevator», «Pacific Gas & Electric», «Ford» та інші почали інвестувати програми подальших розробок в цьому напрямі. Це не забарилося позначитися на результатах. Отримавши солідну фінансову підтримку, вчені змогли швидко реалізувати свої розробки

для широкого круга додатків. Таким чином, інструмент нечітких обчислень вийшов на масовий ринок.

Можна навести і інші приклади вживання нечітких обчислень в бізнесі. Вдалий досвід Ганса Циммермана (Hans Zimmermann) по використанню експертної системи з нечіткими правилами для аналізу інвестиційної активності в місті Аахене (Германія) привів до створення комерційного пакету ASK для оцінки кредитних і інвестиційних ризиків. А система управління складськими запасами, описана як приклад в пакеті CubiCalc, настільки проста, що може бути з легкістю використана мало підготовленим оптовим торговцем.

Основними «споживачами» нечітких обчислень на ринку України є банкіри і фінансисти, а також фахівці в області політичного і економічного аналізу. Вони використовують нечіткі обчислення для створення моделей різних економічних, політичних, біржових ситуацій. Таким чином, лише чітке розуміння основних аспектів розвитку і вживання нечітких методів може забезпечити швидке вирішення багатьох наукових задач, а також всіляких проблем в різних областях ділової діяльності людини.

Умовно період від моменту зародження даної науки до наших днів можна розділити на три етапи:

- перший (1965 р. - початок 70-х рр.) - етап формування основних теоретичних постулатів;
- другий (1973 р. - початок 90-х рр.) - етап практичних розробок в різних сферах життя, заснованих на нечіткій логіці; народження нового наукового напряму в рамках нечіткої логіки «Fuzzy Economics»;
- третій (1995 р. - наш час) - етап масового використання продукції, в основі роботи якої лежить нечітка логіка. Проте таке ділення достатньо умовно, оскільки теоретичні дослідження в цій галузі знань не припиняються і до цих пір, з кожним роком розширюючи сферу застосування даного математичного апарату.

Першим серйозним кроком у напрямі моделювання неоднозначних тверджень виявилася теорія нечітких, або пухнастих, множин (Fuzzy Sets).

Батьком нечітких множин став американський професор Каліфорнійського університету в Берклі директор Ініціативи Берклі по м'яких обчисленнях (Berkeley Initiative in Soft Computing, BISC) Лотфі Аскер Заде (Lotfi Asker Zadeh), академік Американської Інженерної Академії і іноземний член Російської Академії Природних Наук, почесний доктор півтора десятків університетів в різних країнах.

Поняття нечіткої множини, розробленої Заде і його послідовниками, має наступні цікаві асоціативні зв'язки з деякими концепціями світової культури. Свого часу Аристотель прагнув створити такий інструмент мислення, який би правильно відображав реальність. Відомі його слова: «Істина - це коли ми говоримо про те, що є, що воно є, а хибність - це коли про те, чого немає, говорять, що воно є, або коли про те, що є, говорять, що його немає». Проте у Аристотеля разом з двома значеннями - є та ні - було ще третє - «сімбібекус». На українській мові це перекладається як «перехідний». Можна про щось сказати, що воно є, а про інше - що його немає. А буває і третє - іноді є, іноді немає. Або якоюсь мірою є, а якоюсь - немає. Ще Аристотель розумів це. Ось прямий шлях від Аристотеля до нечітких множин.

Існує легенда, що підставою для створення нової теорії послужила суперечка професора зі своїм колегою про те, чия з дружин привабливіша. Згідно історії, до єдиної думки вони так і не прийшли. А це, у свою чергу, змусило вченого сформулювати концепцію, яка виражає нечіткі поняття типу «привабливість» у числовій формі.

Наступним досягненням теорії нечітких множин є введення так званих нечітких чисел - нечітких підмножин спеціалізованого вигляду, відповідних висловам типу «значення змінної приблизно рівне а». Як приклад можна використовувати так зване трикутне нечітке число, де виділяються три точки: мінімально можливе, найбільш очікуване і максимально можливе значення чинника. Трикутні числа - це найчастіше використовуваний на практиці тип нечітких чисел, причому найчастіше їх використовують як прогнозні значення параметра.

Черговим історичним кроком в даній науці є введення набору операцій над нечіткими числами, які зводяться до операцій алгебри із звичайними числами при завданні певного інтервалу достовірності (рівня приналежності), що отримали згодом назву м'які обчислення. Фундаментальні дослідження в цій області зроблені Д. Дюбуа (Dubois D.) і Х. Прадом (Prade H.). Паралельно з розробкою теоретичних основ нової науки, Лотфі А. Заде опрацьовував різні можливості її практичного застосування. У 1973 році ці зусилля увінчалися успіхом - йому вдалося показати, що нечітка логіка може бути покладена в основу нового покоління інтелектуальних систем управління. Саме тому цю дату логічно вважати за початок другого етапу в розвитку даної науки. Пройшли ще декілька років і теоретична алгебра Заді, завдячуючи Ібрагиму Мамдані (Ebrahim Mamdani) з лондонського коледжу королеви Марії (Queen Mary College), запрацювала «в залізі». Саме Мамдані в 1975 р. спроектував перший контролер, що функціонує на основі алгебри Заде, і керує паровою турбіною (варто відмітити, що принципи його побудови алгоритмики стали канонічними і увічнені загальноприйнятою серед фахівців назвою Mamdani-type controller).

Розробка теорії нечітких множин забезпечило основу для розвитку закладеною Заде в 1970 - 1973 рр. нечіткої логіки (Fuzzy Logic) - гнучкого підходу до аналізу міркувань і моделювання складних гуманістичних систем, поведінка яких описується швидше лінгвістичними, ніж числовими змінними. У вузькому сенсі нечітка логіка - це логічна система, що розширює багатозначну логіку до неперервної, але список основних операцій відрізняється як по духу, так і за змістом від основних операцій систем багатозначних логік. Практичний потенціал теорії нечітких множин і нечіткої логіки, їх здатність моделювати гнучкі і неточні обмеження, частковий прояв властивостей, плавний перехід з однієї ситуації в іншу привернули врешті-решт в цю область цілу армію прикладників. Сьогодні теорія нечітких множин і нечітка логіка отримали справді всесвітнє визнання.

За тридцять років свого розвитку (перші два етапи в приведеній вище класифікації), нечітка логіка зазнала ряд істотних змін і доповнень. Перш за все, завдяки зусиллям Б. Коско (Bart Kosko), був досліджений взаємозв'язок нечіткої логіки і теорії нейронних мереж і доведена основоположна FAT-теорема (Fuzzy Approximation Theorem), що підтвердила повноту нечіткої логіки. Була розроблена нечітка алгебра - незвичайна наука, що дозволяє використовувати при обчисленнях як точні, так і приблизні значення змінних. І нарешті, найширшого поширення набули винайдені Б. Коско так звані нечіткі когнітивні моделі (Fuzzy Cognitive Maps), на яких базуються більшість сучасних систем динамічного моделювання в області фінансів, політики і бізнесу.

Починаючи з кінця 70-х років, методи теорії нечітких множин починають застосовуватися і в економіці. Слід згадати роботи Дж. Баклі «Вирішення нечітких рівнянь в економіці і фінансах» і «Нечітка математика у фінансах», Г. Бояджієва і М. Бояджієва «Нечітка логіка в бізнесі, фінансах і менеджменті», Лафуенте «Фінансовий аналіз в умовах невизначеності», Г. Циммермана «Теорія нечіткої логіки і її застосування» та інші. У 80-х роках почали з'являтися програмні рішення і інформаційні технології, які були здатні вирішувати економічні задачі із застосуванням нечітко - множинних і споріднених ним описів. Так, під керівництвом Ц. Зопоунідіса в Технічному університеті на острові Кріт була розроблена експертна система для детального фінансового аналізу корпорацій. Трохи раніше в Германії групою під керівництвом Г. Циммермана була розроблена система стратегічного планування, в якій реалізується позиціонування бізнесу корпорації на основі нечітких описів конкурентоспроможності і привабливості бізнесу.

Деяка кількість робіт присвячена макроекономічному аналізу фондового ринку на основі нечітких уявлень, наприклад, робота К. Пірей «Нечітко - множинний аналіз інвестиційної діяльності взаємних фондів»; Р. Тріппі «Штучний інтелект у фінансовій і інвестиційній діяльності». На особливу увагу заслуговує макроекономічне дослідження, присвячене

вимірюванню рівня тіньової економіки в Новій Зеландії, виконане Р. Драєсеке і Д. Глісом в 1999 р. Використовуючи статистичні дані з 1963 р. по 1994 р., вчені спробували оцінити динаміку величини тіньової економіки в Новій Зеландії за вказаний інтервал часу. Досить швидко економічні додатки теорії нечітких множин утворили самостійний науковий напрям. Була створена міжнародна асоціація SIGEF (International Association for Fuzzy Set Management & Economy) з штаб квартирою в Барселоні, яка регулярно апробовує нові результати в області нечітко - множинних економічних досліджень, проводячи щорічні конференції і випускаючи журнал «Fuzzy Economic Review».

Розглянемо основні положення теорії нечітких множин. Ця теорія є узагальненням теорії класичних чітких множин, які розглядаються як підмножини деякого універсуму. Поняття нечіткої множини - це спроба математичної формалізації нечіткої інформації для побудови математичних моделей. У основі цього поняття лежить уявлення про те, що складаючи дану множину елементи, які володіють загальною властивістю, можуть володіти цією властивістю в різному ступені і, отже належати до даної множини з різним ступенем. При такому підході висловлювання типу «такий-то елемент належить даній множині втрачають сенс, оскільки необхідно вказати «наскільки сильно» або з яким ступенем конкретний елемент задовольняє властивостям даної множини.

Означення 1. *Нечіткою множиною* A у множині U називається сукупність пар виду $(u, \mu_A(u))$, де $u \in U$, а $\mu_A(u)$ - це функція приналежності нечіткої множини A, $\mu_A : U \to [0,1]$. Тут U - деяка звичайна множина, яка називається *універсальною множиною*.

Для будь-якого елемента U функція приналежності μ_A визначає ступінь належності даного елемента множині A. Нечітку множину можна записати таким чином

$$A = \bigcup_{u \in U} \mu_A(u)/u.$$

Приклади запису нечітких множин:

1. Якщо $U=(a,b,c,d,e,f)$; $M=(0,0.5,1)$, тоді A можна представити у вигляді $A=(0/a,1/b,0.5/c,0/d,0.5/e,0/f)$.

2. Якщо $A=(0.8/a_1,1/a_2,0.4/a_3,0.2/a_4,0.5/a_5,0/a_6)$, то $U=(a_1,a_2,a_3,a_4,a_5,a_6)$; $M=(0,0.2,0.4,0.5,0.8,1)$.

3. Нечітка множина A «Декілька» на дискретному універсумі $U=\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ (рис.6.1)

$$\mu_A(u)=\begin{cases}0.0, u\in[0,1,2,9,10],\\ 0.5, u\in[3,8],\\ 0.8, u\in[4,7],\\ 1.0, u\in[5,6].\end{cases}$$

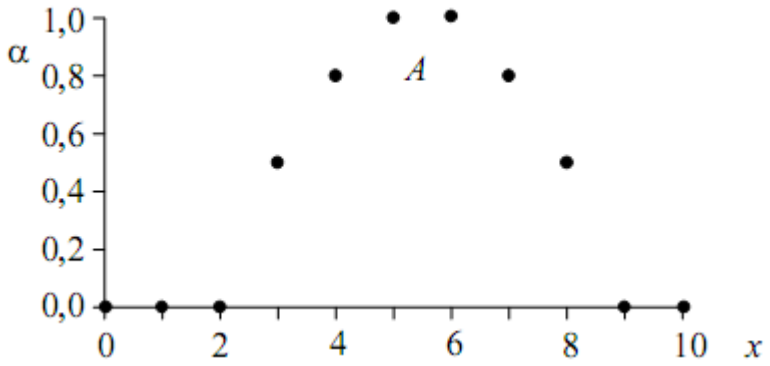


Рис. 6.1. Діаграма Заде нечіткої множини A «Декілька».

4. «Трапецієвидна» нечітка множина A на універсумі [0, 1] (рис. 6.2)

$$\mu_A(u)=\begin{cases}0, u\in[0.0,0.2],\\ 10u-2, u\in[0.2,0.3],\\ 1, u\in[0.3,0.5],\\ (8-10u)/3, u\in[0.5,0.8],\\ 0, u\in[0.8,1.0].\end{cases}$$

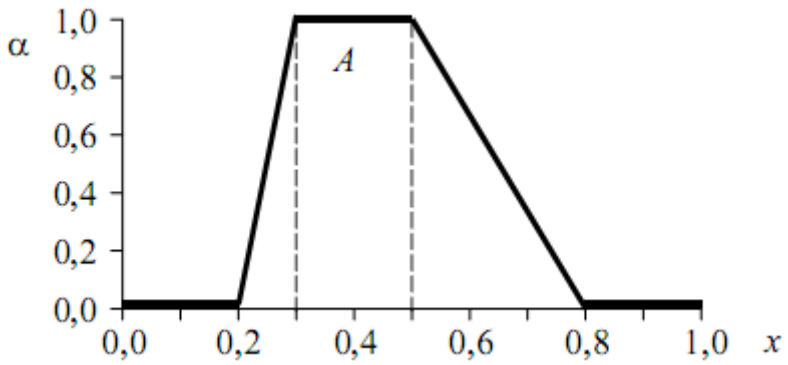


Рис. 6.2. Діаграма Заде «трапецієвидної» нечіткої множини A.

Звичайна множина складає підклас нечітких множин. Функцією приналежності звичайної множини $B \subset U$ є функція

$$\mu_B(u) = \begin{cases} 1, u \in B, \\ 0, u \notin B. \end{cases}$$

Означення 2. *Носієм* (support) нечіткої множини A називається звичайна підмножина таких точок U, для яких величина $\mu_A(u)$ додатна. Носій позначається $S(A)$ або $SuppA$: $S(A) = \{u \mid u \in U, \mu_A(u) > 0\}$.

Оскільки формально нечітка множина - це чітка функція приналежності, то найпростіше нечітку множину A визначити, вказавши її функцію приналежності μ_A і універсум U. Відмітимо також, що ті елементи універсуму, для яких функція приналежності дорівнює нулю, нечіткій множині не належать. Нечітка множина присутня лише там, де її функція приналежності більше нуля; при цьому значення функції приналежності визначає міру нечіткості елементу нечіткої множини.

Означення 3. Висотою (altitude) $h(A)$ нечіткої множини A називається величина $h(A) = \sup_{u \in U} \mu_A(u)$.

Нечітка множина A називається *нормальною*, якщо її висота дорівнює одиниці. В іншому випадку множина A є *субнормальною*. Слід зазначити, що субнормальну нечітку множину завжди можна нормалізувати, поділивши функцію приналежності μ_A на величину $h(A) = \sup_{u \in U} \mu_A(u)$.

Означення 4. Елементи множини U, для яких ступінь приналежності $\mu_A(u) = 0.5$ називаються *точками переходу* (crossover point) нечіткої множини A.

Означення 5. *Ядро* (kernel, core) нечіткої множини A - це чітка множина елементів універсуму, в яких міра нечіткості дорівнює 1: $core(A) = \{u \mid \mu_A(u) = 1\}$.

Означення 6. *Границя* (boundary) нечіткої множини A - це чітка множина елементів універсуму, в яких міра нечіткості відмінна від 0 та 1: $boun(A) = \{u \mid 0 < \mu_A(u) < 1\}$ (рис. 6.3).

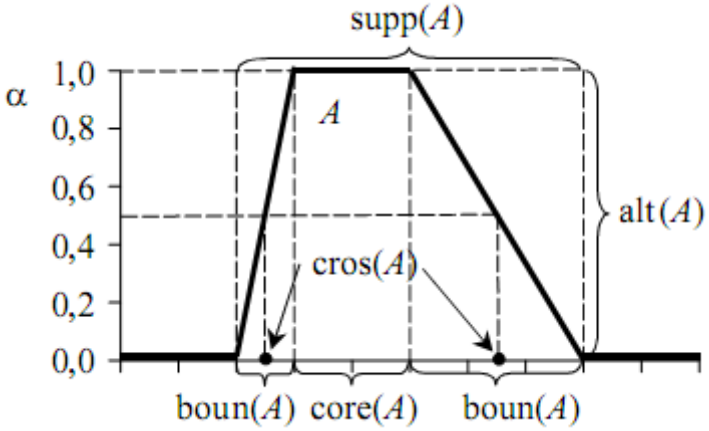


Рис. 6.3. Параметри нечіткої множини.

Приклади нечітких множин:

1. Нехай універсальна множина U представлена у вигляді $\{a,b,c,d,e\}$ і нечітка підмножина A, яка задана на U, має вигляд $A = (0/a, 0.5/b, 0.6/c, 0.7/d, 0.85/e)$. Тоді носієм нечіткої множини A є $S(A) = \{b,c,d,e\}$. Висота нечіткої множини A дорівнює $h(A) = 0.85$, точка переходу - $u = b$, а сама множина є субнормальною. Нормалізована множина буде мати вигляд $A = (0/a, 0.6/b, 0.7/c, 0.8/d, 1/e)$.

2. Нехай універсальна множина U представляє собою інтервал [0, 100], і змінна u, яка приймає значення з цього інтервалу, інтерпретується як «Вік». Тоді нечітку множину A, яка позначається терміном «Старий», можна визначити функцією приналежності вигляду

$$\mu_A(u)=\begin{cases}0, якщо 0\le u\le 50\\ \left(1+\left(\dfrac{u-50}{5}\right)^{-2}\right)^{-1}, якщо 50<u\le 100\end{cases}.$$

Тут носій $S(A)=(50,100)$, висота множини «Старий» близька до одиниці, відповідно множина нормальна. Точкою переходу є значення $u=55$.

3. Існує множина $U=[0,100]$ і змінна u, яка приймає значення з цього інтервалу, та інтерпретується як «Вік». Тоді нечітку множину «Молодий» можна визначити функцією приналежності вигляду

$$\mu_{Молодий}(u)=\begin{cases}1, якщо 1\le u\le 25\\ \dfrac{1}{1+((u-25)/5)^2}, якщо 25<u\le 100\end{cases}.$$

Нечітка множина «Молодий» на універсальній множині $U=\{$Богдан, Іванов, Сілков, …$\}$ задається за допомогою функції приналежності $\mu_{Молодий}(u)$ на $U=[0,100]$, яка по відношенню до U називається *функцією сумісності*. При цьому $\mu_{Молодий}(Богдан)=\mu_{Молодий}(u)$, де u - вік Богдана.

4. Хай $U=\{$Запорожець, Жигулі, Мерседес....$\}$ - множина марок автомобілів, а $U=[0,\infty)$ - універсальна множина «Вартість». Тоді на U можна визначити нечітку множину типа: «Для небагатих», «Для середнього класу», «Престижні», з функціями приналежності наступного вигляду (рис. 6.4).

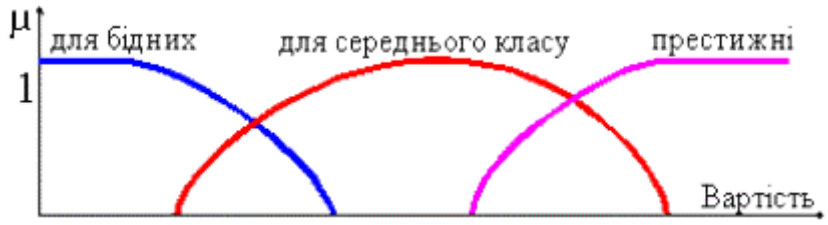


Рис. 6.4. Приклади функцій приналежності.

Маючи ці функції і знаючи ціни автомобілів з U в даний момент часу, визначимо на U нечітку множину з цими ж назвами. Так, наприклад, нечітка

множина «Для небагатих», задана на універсальній множині $U = \{$Запорожець, Жигулі, Мерседес....$\}$ виглядає таким чином (рис. 6.5)

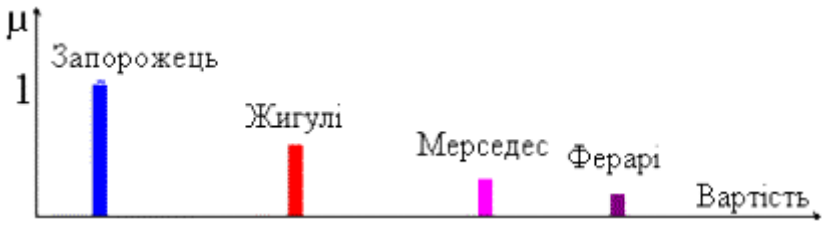


Рис. 6.5. Приклад задання нечіткої множини.

Аналогічно можна визначити нечіткі множини «Швидкісні», «Середні», «Тихохідні» та інше.

Означення 7. *Множиною α-рівня* нечіткої множини A є звичайна множина A_α всіх таких елементів універсальної множини U, ступінь приналежності яких нечіткій множині A більше або рівна α: $A_\alpha = \{u \mid \forall u \in U, \mu_A(u) \geq \alpha\}$. Множину α-рівня іноді називають *перетином* α нечіткої множини A. Причому, якщо $\mu_A(u) \geq \alpha$, то говорять про *сильний перетин,* якщо $\mu_A(u) > \alpha$, то про *слабкий перетин*.

Нечітку множину A можна розкласти по її множинам рівня наступним чином: $A = \bigcup_\alpha \alpha A_\alpha$, де αA_α - множення числа α на множину A_α. Знак $\bigcup_\alpha$ - знак об'єднання множин A_α по α. Розглянемо приклад: якщо нечітка множина $A = \{0.3/a, 0.4/d, 0.7/c, 0.8/f, 0.6/b\}$, то множиною α-рівня при $\alpha = 0.7$ буде множина $A_{07} = \{c, f\}$. Множина A розкладена по її множинам α-рівня має вигляд $A = 0.3\{a,d,c,f,b\} \cup 0.4\{d,c,f,b\} \cup 0.6\{c,f,b\} \cup 0.7\{c,f\} \cup 0.8\{f\}$.

Над нечіткою множиною можна виконувати різні операції, при цьому необхідно визначити їх так, щоб в окремому випадку, коли множина є чіткою, операції переходили в звичайні операції теорії множин, тобто операції над нечіткою множиною повинні узагальнювати відповідні операції над звичайною множиною. При цьому узагальнення може бути реалізоване різними способами,

через що якій-небудь операції над звичайною множиною може відповідати декілька операцій в теорії нечіткої множини.

Спочатку розглянемо виконання логічних операцій над нечітким множинами.

Означення 8. Операція *включення* ($A \subset B$). Нехай A та B є нечіткими множинами на універсальній множині U. Говорять, що A *міститься* в B, якщо $\forall u \in U \quad \mu_A(u) \le \mu_B(u)$. Іноді застосовують термін *домінування*, тобто в випадку, коли $A \subset B$, говорять, що B домінує над A.

Означення 9. *Рівність*. Нехай A та B є нечіткими множинами на універсальній множині U. Говорять, що A і B *рівні* ($A = B$), якщо $\forall u \in U$ $\mu_A(u) = \mu_B(u)$.

Означення 10. Нехай A та B є нечіткими множинами на універсальній множині U. *Об'єднанням* нечітких множин A і B в U називається нечітка підмножина $A \cup B$, яка включає як A, так і B, з функцією приналежності вигляду: $\mu_{A \cup B}(u) = \max(\mu_A(u), \mu_B(u))$, $u \in U$. Об'єднання відповідає сполучнику «АБО». Таким чином, якщо X та Y - символи нечітких множин, то X АБО $Y = X \cup Y$ (рис. 6.6).

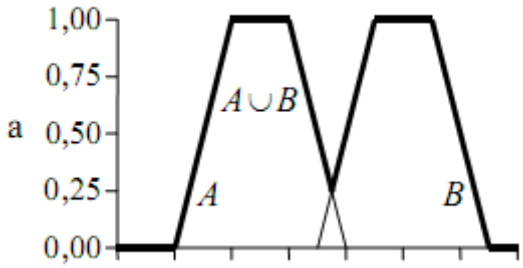


Рис. 6.6. Об'єднання нечітких множин.

Означення 11. *Перетином* нечітких множин A і B в U називається нечітка підмножина $A \cap B$, яка міститься одночасно в A та B, з функцією приналежності вигляду $\mu_{A \cap B}(u) = \min(\mu_A(u), \mu_B(u))$, $u \in U$. Перетин відповідає сполучнику «І». Таким чином, якщо X та Y - символи нечітких множин, то $X \cap Y = X \cap Y$ (рис. 6.7).

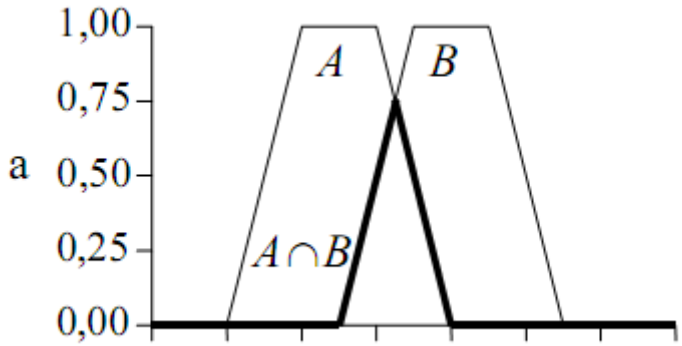


Рис. 6.7. Перетин нечітких множин.

Означення 12. *Доповненням* нечіткої множини A називається нечітка множина $\overline{A}$ з функцією приналежності $\mu_{\overline{A}(u)} = 1 - \mu_A(u)$, $\forall u \in U$. Операція доповнення відповідає операції «НІ», тобто $\overline{X} = \bigcup_{u \in U} (1 - \mu_X(u))/u$ (рис. 6.8).

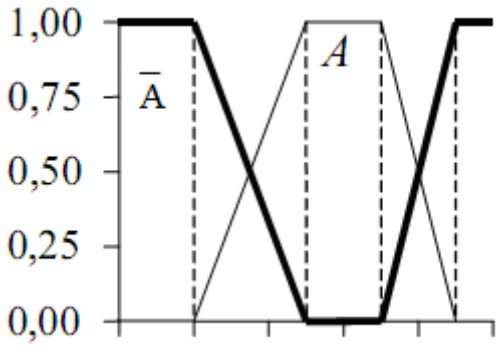


Рис. 6.8. Доповнення нечіткої множини.

Означення 13. *Різниця* нечітких множин A і B в U називається порізному, застосуванням двох незалежних операцій:

$$\mu_{A-B}(u) = \begin{cases} \mu_A(u) - \mu_B(u), \mu_A(u) \geq \mu_B(u) \\ 0, \mu_A(u) < \mu_B(u) \end{cases}$$

або $A - B = A \cap \overline{B}$ з функцією приналежності $\mu_{A-B}(u) = \min(\mu_A(u), 1 - \mu_A(u))$ (рис. 6. 9).

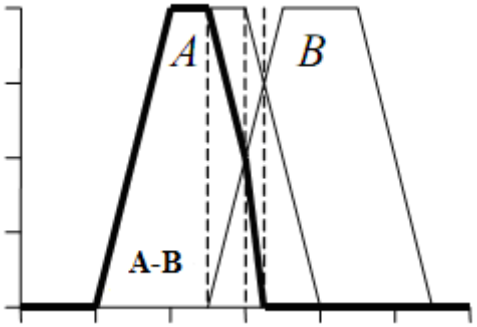


Рис. 6.9. Різниця нечітких множин.

Визначення 14. *Диз'юнктивна сума* $A \oplus B$ визначається виразом вигляду $A \oplus B = (A \cap \overline{B}) \cup (\overline{A} \cap B)$ з функцією приналежності вигляду (рис. 6.10)

$$\mu_{A \oplus B}(u) = \max[\min(\mu_A(u), 1 - \mu_B(u)), \min(1 - \mu_A(u), \mu_B(u))].$$

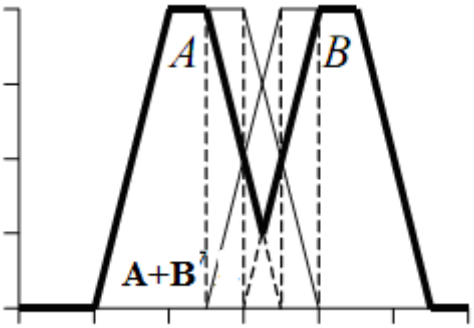


Рис. 6.10. Диз'юнктивна сума нечітких множин.

Наведемо приклади виконання логічних операцій. Вважатимемо, що

$$A = 0.4/u_1 + 0.2/u_2 + 0/u_3 + 1/u_4$$

$$B = 0.7/u_1 + 0.9/u_2 + 0.1/u_3 + 1/u_4, \text{ тоді}$$

$$\overline{A} = 0.6/u_1 + 0.8/u_2 + 1/u_3 + 0/u_4$$

$$A \cap B = 0.4/u_1 + 0.2/u_2 + 0/u_3 + 1/u_4$$

$$A \cup B = 0.7/u_1 + 0.9/u_2 + 0.1/u_3 + 1/u_4$$

$$A - B = A \setminus \overline{B} = 0.3/u_1 + 0.1/u_2 + 0/u_3 + 0/u_4$$

$$A \oplus B = 0.6/u_1 + 0.8/u_2 + 0.1/u_3 + 0/u_4.$$

Вищеприведені операції володіють наступними властивостями:

1. Комутативність: $A \cap B = B \cap A,\ A \cup B = B \cup A.$
2. Асоціативність: $(A \cap B) \cap C = A \cap (B \cap C)$, $(A \cup B) \cup C = A \cup (B \cup C)$.
3. Ідемпотентність: $A \cap A = A,\ A \cup A = A.$
4. Дистрибутивність: $A \cap (B \cup C) = (A \cap B) \cup (A \cap C),$ $A \cup \varnothing = A,$ $A \cup U = U$, $A \cup (B \cap C) = (A \cup B) \cap (B \cup C)$, $A \cap \varnothing = \varnothing$, $A \cap U = A,$.
5. Інволюція: $\overline{\overline{A}} = A.$
6. Теореми де Моргана: $\overline{A \cap B} = \overline{A} \cup \overline{B}$, $\overline{A \cup B} = \overline{A} \cap \overline{B}$.

Розглянемо виконання деяких алгебраїчних операцій над нечітким множинами.

Означення 15. Алгебраїчний добуток A і B $(A \cdot B)$ визначається функцією приналежності вигляду $\mu_{AB}(u) = \mu_A(u)\mu_B(u)$, $\forall u \in U$.

Означення 16. Алгебраїчна сума цих множин $(A + B)$ визначається функцією приналежності $\mu_{A+B}(u) = \mu_A(u) + \mu_B(u) - \mu_A(u)\mu_B(u)$, $\forall u \in U$.

Означення 16. Ступенем нечіткої множини A називається нечітка множина A^α з функцією приналежності $\mu_{A^\alpha}(u) = \mu_A^\alpha(u)$, $\forall u \in U$, $\alpha > 0$.

При $\alpha = 2$ отримуємо операцію *концентрації* (ущільнення) (CON): $\mathrm{CON}(A) = A^2$. В результаті застосування цієї операції до множини A знижується ступінь нечіткості опису, причому для елементів з високим ступенем приналежності це зменшення відносно мале, а для елементів з малим ступенем приналежності відносно велике (рис. 6.11).

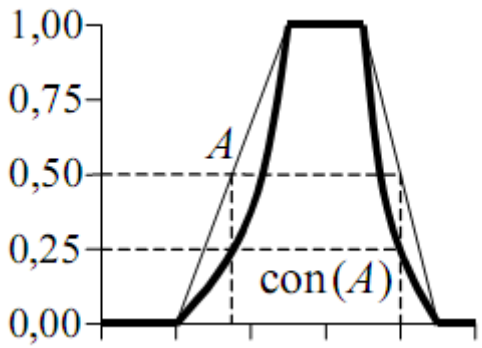


Рис. 6.11. Концентрація нечіткої множини.

При $\alpha = 0.5$ отримуємо операцію *розтягування* (DIL): $\mathrm{DIL}(A) = A^{0.5}$. Ця операція збільшує ступінь нечіткості деякої нечіткої множини (рис. 6.12).

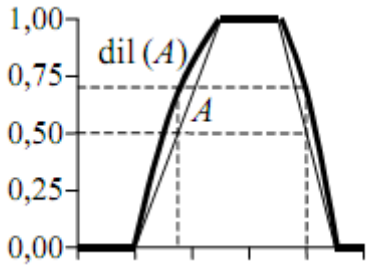


Рис. 6.12. Розтягування нечіткої множини.

Означення 17. *Множення на число.* Якщо α - позитивне число, таке, що $\alpha \max \mu_A(u) \le 1$, то нечітка множина αA має функцію приналежності $\mu_{\alpha A}(u) = \alpha \mu_A(u)$ (рис. 6.13).

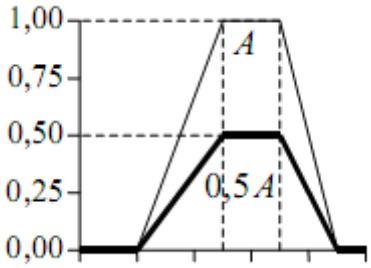


Рис. 6.13. Множення на число нечіткої множини.

Важливою частиною теорії нечітких множин є задачі нечіткої класифікації. Нехай є набір X фотографічних портретів всіх членів декількох сімей. Потрібно розділити цей набір на групи так, щоб в кожній виявилися портрети членів лише однієї сім'ї. Нехай $f_1(x, y)$ - функція приналежності нечіткого бінарного відношення схожості на заданому наборі фотографій. Для кожної пари фотографій x та y значення $f_1(x, y)$ є суб'єктивна оцінка людиною міри схожості x і y. Це нечітке відношення можна розглядати як свого роду «експериментальні дані», що відображають розуміння людиною поняття «схожості» в даній задачі. Наступний етап - використання цих «даних» для потрібної класифікації фотографій.

Відмітимо, що нечітке відношення $f_1(x,y)$ володіє природними властивостями рефлексивності і симетричності. Воно називається однокроковим відношенням, в тому сенсі, що описує результати лише попарного порівняння портретів один з одним. Для $f_1(x,y)$ вводиться n - крокове відношення $f_n(x,y)$ таким чином:

$$f_n(x,y) = \sup_{x \in X} \min\{ f_1(x,x_1),..., f_1(x_{n-1},y)\}.$$

Це відношення є n композицією вихідного «експериментального» відношення $f_1(x,y)$ і є в деякому розумінні його уточненням. Неважко показати, що для будь-яких $x, y \in X$ виконується ланцюжок нерівностей

$$0 \le f_1(x,y) \le f_2(x,y) \le ... \le f_n(x,y) \le ... \le 1,$$

з якого виходить, зокрема, що для будь-яких $x, y \in X$ послідовність $\{f_k(x,y)\}$ має межу при $k \to \infty$. Таким чином, існує граничне відношення схожості, визначуване рівністю $f(x,y) = \lim_{k \to \infty} f_k(x,y)$, для всіх $x, y \in X$. Це граничне відношення є кінцевим результатом обробки результатів нечітких вимірів $f_1(x,y)$ і таким чином використовується для класифікації.

Для довільного числа λ $(0 < \lambda < 1)$ вводиться звичайне відношення R_λ: $R_\lambda(x,y) \Leftrightarrow f(x,y) \ge \lambda$. Нескладно показати, що для будь-якого λ $(0 < \lambda < 1)$ R_λ є відношення еквівалентності в X, тобто для будь-яких $x, y \in X$ виконуються звичайні аксіоми еквівалентності:

- рефлективність - $R_\lambda(x,x)$;
- симетричність - $R_\lambda(x,y) \Rightarrow R_\lambda(y,x)$;
- транзитивність - $R_\lambda(x,y) \,\&\, R_\lambda(y,z) \Rightarrow R_\lambda(x,z)$.

Відмітимо, що остання аксіома є наслідком того, що граничне нечітке відношення $f(x,y)$ володіє властивістю нечіткої транзитивності $f(x,z) \ge \min\{ f(x,y), f(y,z)\}$, для всіх $x, y, z \in X$.

Остаточний етап алгоритму класифікації - розбиття множини X на класи еквівалентності по отриманому відношенню R_λ. Вибір величини порогу

λ в цьому алгоритмі здійснюється, виходячи з умов початкової задачі. У наведеному прикладі з фотографіями цей вибір здійснювали таким чином. Хай є набір з 20 фотографій представників 3 сімей. Тоді величину λ вибирають так, щоб в результаті реалізації алгоритму класифікації вийшли 3 класи еквівалентності по відношенню R_λ.

Представлена теорія дозволяє виявити наступні переваги fuzzy-систем в порівнянні з іншими:

- можливість оперувати нечіткими вхідними даними: наприклад, значеннями, що безперервно змінюються в часі (динамічні задачі), значеннями, які неможливо задати однозначно (результати статистичних опитів, рекламні компанії і так далі);
- можливість нечіткої формалізації критеріїв оцінки і порівняння: оперування критеріями «більшість», «можливо», «переважно» і т.д.;
- можливість проведення якісних оцінок як вхідних даних, так і вихідних результатів: маємо можливість оперуєте не лише значеннями даних, але і їх мірою достовірності і її розподілом;
- можливість проведення швидкого моделювання складних динамічних систем і їх порівняльного аналізу із заданою мірою точності: оперуючи принципами поведінки системи, описаними fuzzy-методами, ми по-перше, не витрачаєте багато часу на з'ясування точних значень змінних і складання рівнянь, що їх описують, по-друге, можна оцінити різні варіанти вихідних значень.

6.2. Нечітка логіка в системах Data Mining

Нечітка логіка «Fuzzy Logic» - це узагальнення традиційної аристотелевої логіки на випадок, коли істинність розглядається як лінгвістична змінна, що набуває значень типа: «дуже істинно», «більш-менш істинно», «не

дуже помилково» і тому подібне Вказані лінгвістичні значення представляються нечіткою множиною.

У поєднанні слів «нечіткий» і «логіка» є щось незвичайне. Логіка в звичайному сенсі слова є представлення механізмів мислення, те, що ніколи не може бути нечітким, але завжди строгим і формальним. Проте математики, що досліджували ці механізми мислення, відмітили, що насправді існує не одна логіка (наприклад, булева), а стільки, скільки ми побажаємо, тому що все визначається вибором відповідної системи аксіом. Звичайно, як тільки аксіоми вибрані, всі твердження, побудовані на їх основі, мають бути строгими, без протиріч зв'язаними один з одним згідно правилам, встановленим в цій системі аксіом.

Людське мислення - це поєднання інтуїції і строгості, яке, з одного боку, розглядає світ в цілому або по аналогії, а з іншого боку - логічно і послідовно і, значить, є нечітким механізмом. Закони мислення, які хотілося б включити в програми комп'ютерів, мають бути обов'язково формальними; закони мислення, що проявляються в діалозі людини з людиною, - нечіткі. Чи можна тоді стверджувати, що нечітка логіка може бути добре пристосована до людського діалогу? Так - якщо програмне забезпечення, розроблене з врахуванням нечіткої логіки, стане операційним і зможе бути технічно реалізоване, то людино-машинне спілкування стане набагато зручнішим, швидшим і краще пристосованим до вирішення проблем.

Вважається, що нечітка логіка виникла як найбільш зручний спосіб побудови систем управління складними технологічними процесами, а вже потім знайшла широке вживання в різного роду комп’ютерних аналітичних системах [5, 23, 65, 71].

Важливим елементом нечіткої логіки є поняття нечіткої і лінгвістичної змінних, що використовується при описі об'єктів і явищ за допомогою нечіткої множини.

Означення 18. *Нечіткою змінною* називається сукупність (кортеж) вигляду $< X, U, x >$, де X - найменування нечіткої змінної, $U = \{u\}$ - область її

визначення (універсальна множина), $x = \bigcup_{u \in U} \mu_x(u)/u$ - нечітка множина на U, що описує обмеження на значення нечіткої змінної X.

Наприклад, якщо провести аналогію з саквояжем, то нечітку змінну можна уподібнити саквояжу з ярликом. Тоді X - напис на ярлику (назва саквояжа), U - список предметів, які в принципі можна помістити в саквояж, а x - частина цього списку, де для кожного предмету u вказано число $\mu_x(u)$, що характеризує міру легкості, з якою предмет можна помістити в саквояж X.

Лінгвістична змінна відрізняється від числової змінної тим, що її значеннями є не числа, а слова або словосполучення в природній або формальній мові. Оскільки слова загалом менш точні, чим числа, поняття лінгвістичної змінної дає можливість приблизно описувати явища, які настільки складні, що не піддаються опису в загальноприйнятих кількісних термінах. Зокрема, нечітку множину, яка є обмеженням, пов'язаним із значеннями лінгвістичної змінної, можна розглядати як сукупну характеристику різних підкласів елементів універсальної множини. У цьому сенсі роль нечіткої множини аналогічна тій ролі, яку грають слова і словосполучення в природній мові.

Важливий аспект поняття лінгвістичної змінної полягає в тому, що ця змінна вищого порядку, ніж нечітка змінна, в тому сенсі, що значеннями лінгвістичної змінної є нечіткі змінні. Наприклад, значеннями лінгвістичної змінної «ВІК» можуть бути: «МОЛОДИЙ, НЕМОЛОДИЙ, СТАРИЙ, ДУЖЕ СТАРИЙ, НЕ МОЛОДИЙ І НЕ СТАРИЙ» і тому подібне. Інший важливий аспект поняття лінгвістичної змінної полягає в тому, що лінгвістичній змінній властиві два правила:

1. Синтаксичне, яке може бути задане у формі граматики, що породжує назву значень змінної.

2. Семантичне, яке визначає алгоритмічну процедуру для обчислення сенсу кожного значення.

Означення 19. *Лінгвістичною змінною* (ЛЗ) називається кортеж вигляду $<\beta,T,U,G,M>$, де

β - найменування ЛЗ;

T - множина її значень (терм-множина), яка є найменуванням нечітких змінних, областю визначення кожної з яких є множина U. Множина T називається базовою терм-множиною лінгвістичної змінної. Терм, який складається з одного слова або з декількох слів, що завжди фігурують разом, називається атомарним термом. Терм, який складається із понад одного атомарного терма, називається складеним термом;

G - синтаксична процедура, яка описує процес утворення з елементів множини T нових, осмислених для даної задачі значень лінгвістичної змінної (терм);

M - семантична процедура, яка дозволяє перетворювати кожне нове значення ЛЗ, що створюється процедурою G, в нечітку змінну, тобто сформувати відповідну нечітку множину.

Розглянемо таке нечітке поняття як «Ціна акції». Це і є назва лінгвістичної змінної. Сформуємо для неї базову терм-множину, яка складатиметься з трьох нечітких змінних: «Низька», «Помірна», «Висока» і задамо область міркувань у вигляді $U=[100,200]$ (одиниць). Останнє, що залишилося зробити - побудувати функції приналежності для кожного лінгвістичного терма з базової терм-множини T. Існує понад десяток типових форм кривих для задання функцій приналежності. Найбільшого поширення набули: трикутна, трапецеїдальна і гаусова функції приналежності.

Трикутна функція приналежності визначається трійкою чисел (a,b,c), і її значення в точці x обчислюється згідно виразу:

$$MF(x)=\begin{cases}1-\dfrac{b-x}{b-a}, a\le x\le b\\ 1-\dfrac{x-c}{c-b}, b\le x\le c\\ 0, інші\_випадки.\end{cases}$$

При $(b-a)=(c-b)$ маємо випадок симетричної трикутної функції приналежності, яка може бути однозначно задана двома параметрами з трійки (a,b,c) (рис. 6.14).

Аналогічно для задання трапецеїдальній функції приналежності необхідна четвірка чисел (a,b,c,d):

$$MF(x)=\begin{cases}1-\dfrac{b-x}{b-a}, a\le x\le b\\ 1, b\le x\le c\\ 1-\dfrac{x-c}{d-c}, c\le x\le d\\ 0, інші\_випадки.\end{cases}$$

При $(b-a)=(d-c)$ трапецеїдальна функція приналежності набирає симетричного вигляду (рис. 6.14).

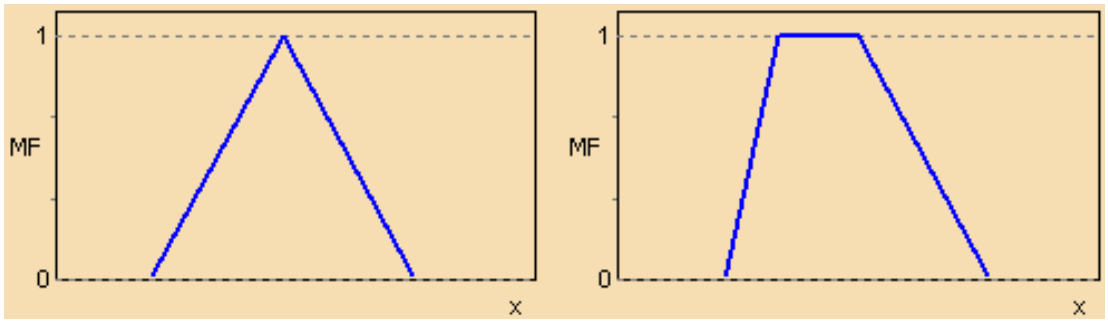


Рис. 6.14. Кусочно–лінійні функції приналежності.

Функція приналежності гаусова типа описується формулою

$$MF(x)=\exp\left[-\left(\frac{x-c}{\sigma}\right)^2\right]$$

і оперує двома параметрами. Параметр c позначає центр нечіткої множини, а параметр σ відповідає за крутість функції (рис. 6.15).

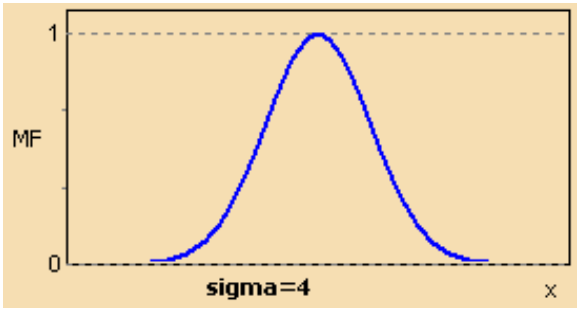


Рис. 6.15. Гаусова функція приналежності.

Сукупність функцій приналежності для кожного терма з базової терм-множини T зазвичай зображаються разом на одному графіку. На рисунку 6.16 наведений приклад описаної вище лінгвістичної змінної «Ціна акції».

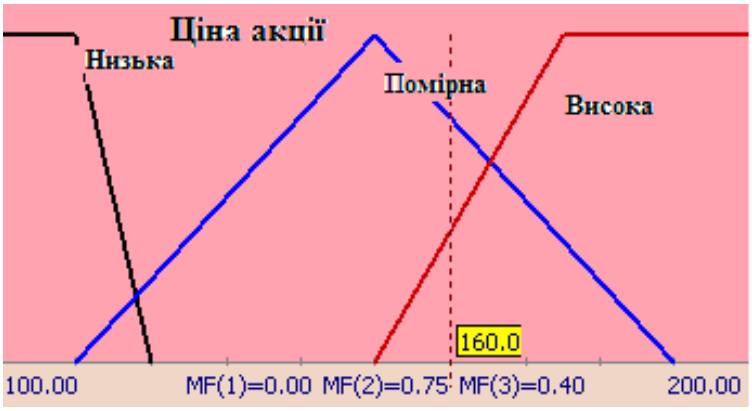


Рис. 6.16. Лінгвістична змінна «Ціна акції».

Розглянемо декілька інших прикладів. Як проста ілюстрація тієї ролі, яку грають синтаксичні і семантичні правила в разі структурованої лінгвістичної змінної, розглянемо змінну «ЗРІСТ», терм-множину якої можна записати у вигляді:

Т(ЗРІСТ)={ВИСОКИЙ, ДУЖЕ ВИСОКИЙ, ДУЖЕ-ДУЖЕ ВИСОКИЙ...}.

$$\text{М(ВИСОКИЙ)} = \begin{cases} \left(1+\left(\dfrac{u-60}{5}\right)^{-2}\right)^{-1}, u \geq 60 \\ 0, інші\_випадки. \end{cases}$$

М(ДУЖЕ ВИСОКИЙ)=(М(ВИСОКИЙ))2, і так далі.

Лінгвістичну змінну називатимемо булевою, якщо її терми є булевими комбінаціями змінних вигляду X_p і hX, де h - лінгвістична невизначеність, X_p - атомарний терм.

Хай «ВІК» - булева лінгвістична змінна з терм-множиною вигляду Т(ВІК)={МОЛОДИЙ, НЕМОЛОДИЙ, СТАРИЙ, НЕСТАРИЙ, ДУЖЕ МОЛОДИЙ, НЕ МОЛОДИЙ І НЕ СТАРИЙ, МОЛОДИЙ АБО НЕ ДУЖЕ СТАРИЙ ...}.

В даному прикладі є два атомарні терми – «МОЛОДИЙ» і «СТАРИЙ» і одна невизначеність – «ДУЖЕ». Якщо ототожнювати союз «І» з операцією пересічення нечіткої множини, «АБО» - з операцією об'єднання нечітких множин, заперечення «НЕ» - з операцією взяття доповнення і модифікатор «ДУЖЕ» - з операцією концентрації, то дана змінна буде повністю структурована.

В залежності від характеру U лінгвістичні змінні можуть бути поділені на числові та нечислові.

Означення 20. Числовою називають лінгвістичну змінну, у якої $U \to R^1$, $R^1 = (-\infty, \infty)$ і яка має вимірювану базову змінну.

Залежність між двома звичайними числовими змінними X та Y найчастіше описується набором висловів, наприклад: «якщо x дорівнює 5, то y дорівнює 12» і т. д. Застосуємо такий же спосіб і для нечітких змінних. Зокрема, якщо X та Y - лінгвістичні змінні, то висловлювання, що описують залежність Y від X, могли б виглядати так: «якщо X мале, то Y велике»; «якщо X не дуже мале, то Y не дуже велике»; «якщо X не дуже мале і не дуже велике, то Y не дуже велике» і таке інше.

Нечіткі висловлювання типу «з A слідує B», де A та B мають невизначене значення, наприклад, «Якщо людина добре до тебе відноситься, то ти повинен бути уважним до неї», звичайні в повсякденній мові. Такі відношення є простими. Для опису більш складних залежностей можуть бути потрібні нечіткі алгоритми.

Якщо звернути увагу на структуру лінгвістичної змінної, то можна відзначити, що в загальному випадку значення лінгвістичної змінної є складеним терміном, що представляє поєднання деяких елементарних термінів. Ці елементарні терміни можна розбити на чотири основні категорії:

- первинні терміни, які є символами спеціальних нечітких підмножин, наприклад, «молодий, старий» та інше;
- заперечення «НІ» та союзи «І, АБО»;
- невизначення типу «дуже, більш-менш» і т. д.;

- маркери, частіше всього це ввідні слова.

Заперечення, союзи, невизначення та інші терміни, які входять в визначення значень лінгвістичної змінної, можуть розглядатися як символи різних операцій, які визначені на нечітких підмножинах U. В зв'язку з цим розглянемо поняття нечіткого числа.

Означення 21. Нечіткі числа – нечіткі змінні, які визначені на числовій осі, тобто нечітке число визначається як нечітка множина A на множині R з функцією приналежності $\mu_A(u) \in [0,1],\ u \in R$.

Нечіткі числа відповідають значенням числової лінгвістичної змінної. Нечітке число є *нормальним*, якщо $\max \mu_A(u) = 1$, та *випукле,* якщо для $x \le y \le z$ виконується $\mu_A(x) \ge \min\{\mu_A(y), \mu_A(z)\}$. *Множина α-рівня* нечіткої множини A визначається як $A\alpha = \{u / \mu_A(u) \ge \alpha\}$. Підмножина $S_A \subset R$ називається *носієм нечіткого числа* A, якщо $S_A = \{u / \mu_A(u) > 0\}$. Нечітке число A *позитивне*, якщо $\forall u \in S_A,\ u > 0$ і *негативне*, якщо $\forall u \in S_A,\ u < 0$.

Для визначення арифметичних операцій Л. А. Заде був сформульований *принцип узагальнення*. Нехай A та B - дві нечіткі множини. $d : R^1 \otimes R^1 \to R^1$ - деяка функція, яка визначає арифметичну операцію. Тоді нечітке число $D = d(A,B)$ визначається функцією приналежності $\mu_D(u) = \sup[\mu_A(u), \mu_B(u)]$, $\sup \min(\mu_A(u), \mu_B(u))$. Тепер бінарні операції можна визначити таким чином:

$$A \otimes B = \bigcup_U \mu_D(u) / (a \otimes b).$$

Розширені бінарні арифметичні операції (складання, множення та ін.) для нечітких чисел визначаються через відповідні операції для чітких чисел з використанням принципу узагальнення таким чином:

$$C = A + B \Leftrightarrow \sup_{z=x+y} (\mu_A(x) \wedge \mu_B(y))$$

$$C = A - B \Leftrightarrow \sup_{z=x-y} (\mu_A(x) \wedge \mu_B(y))$$

$$C = A * B \Leftrightarrow \sup_{z=xy} (\mu_A(x) \wedge \mu_B(y))$$

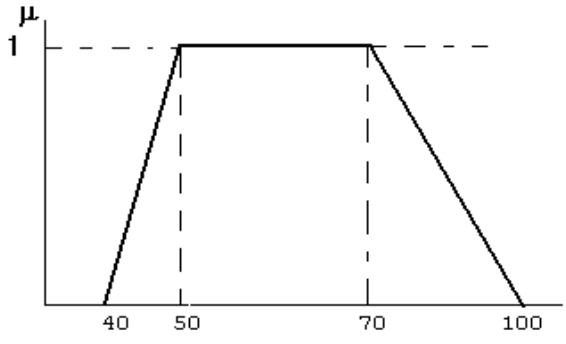
$$C = A : B \Leftrightarrow \sup_{z=x/y} (\mu_A(x) \wedge \mu_B(y)).$$

Аналіз властивостей арифметичних операцій над нечіткими числами показав, що нечітке число не має протилежного і зворотного чисел, складання і множення комутативні, асоціативні і в загальному випадку не дистрибутивні.

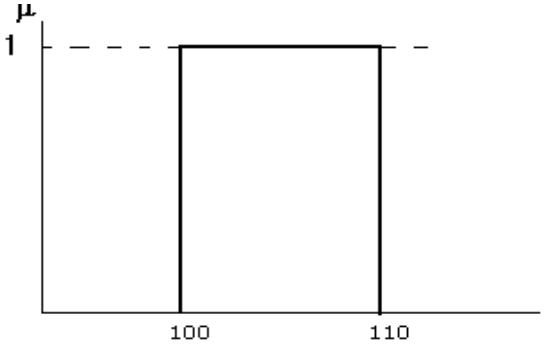
Розглянемо наступний приклад. Хай в рамках складання проекту бюджету розглядаються різні джерела фінансування. Причому деякі з них характеризуються неточністю оцінки грошових сум на день оцінювання, а інші малою надійністю. Крім того, з бюджету необхідно віддати борги, кількість яких також неточна, оскільки залежить від того, чи зажадає кредитор все або лише частину в наступному фінансовому періоді.

- Джерело A: фінансування забезпечується, його сума може змінюватися від 40 до 100 млн. грн. залежно від кон'юнктури, але з найбільшою ймовірністю можна чекати поступлень в сумі від 50 до 70 млн. грн.
- Джерело B: джерело надійне і розумно вважати, що фінансування буде надано і складе суму 100 - 110 млн. грн.
- Джерело *C*: джерело ненадійне, а якщо і дасть, то не більше 20 млн. грн.
- Борг D: плата за кредити 50 - 100 млн. грн., але найбільш ймовірна виплата 80 млн. грн.

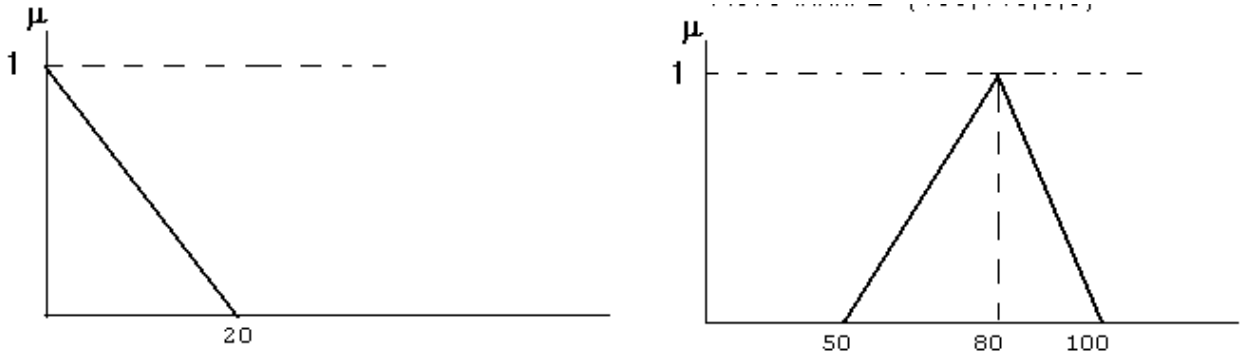
Таким чином, маємо три джерела надходжень і одне джерело витрат. Побудуємо на основі їх описів трапецієвидні функції приналежності для кожної з чотирьох нечітких змінних (рис. 6.17) Після задання всіх нечітких змінних, встає задача визначення суми всього бюджету, яка також буде нечіткою величиною.



Джерело A=(50, 70, 10, 30)



Джерело B=(100, 110, 0, 0)



Джерело C=(0, 0, 0, 20) Борг D=(80, 80, 30, 20)

Рис. 6.17. Проект бюджету.

Слід зазначити, що в випадку застосування трапецієвидних функцій приналежність $\mu_A(u)$ характеризується четвіркою параметрів $(\overline{m}, \underline{m}, \alpha, \beta)$ (рис.6.18).

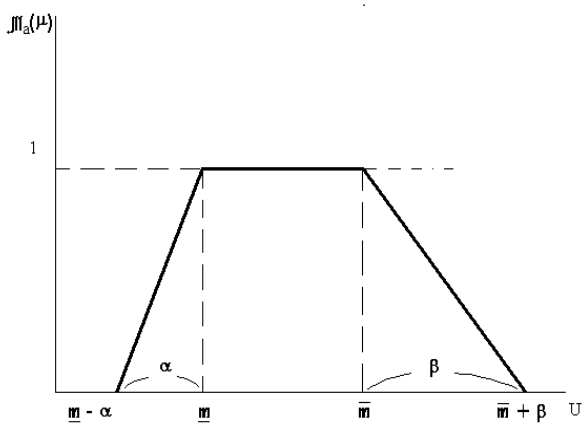


Рис. 6.18. Трапецієвидна функція приналежності.

Визначення арифметичних операції розглянемо для випадку двох нечітких змінних $\tilde{A}_1$ і $\tilde{A}_2$, які задані своїми трапецієвидними функціями приналежності вигляду:

$$\tilde{A}_1 = (\overline{m}_1, \underline{m}_1, \alpha_1, \beta_1),$$

$$\tilde{A}_2 = (\overline{m}_2, \underline{m}_2, \alpha_2, \beta_2)$$

Результатом операції буде також нечітка змінна $A=(\overline{m},\underline{m},\alpha,\beta)$, яка також має трапецієвидну функцію приналежності, параметри якої визначаються залежно від виду арифметичної операції (табл. 6.1).

Табл. 6.1.

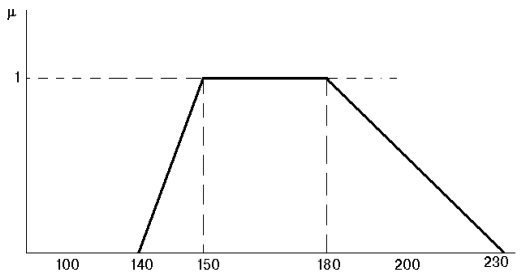
Тип операції	Параметри функції приналежності
$A=\tilde{A}_1(+)\tilde{A}_2$	$\overline{m}=\overline{m}_1+\overline{m}_2,\quad \underline{m}=\underline{m}_1+\underline{m}_2,$ $\alpha=\alpha_1+\alpha_2,\quad \beta=\beta_1+\beta_2$
$A=\tilde{A}_1(-)\tilde{A}_2$	$\overline{m}=\overline{m}_1-\underline{m}_2,\quad \underline{m}=\underline{m}_1-\overline{m}_2,$ $\alpha=\alpha_1+\beta_2,\quad \beta=\beta_1+\alpha_2$
$A=\tilde{A}_1(*)\tilde{A}_2$	$\overline{m}=\overline{m}_1*\overline{m}_2,\quad \underline{m}=\underline{m}_1*\underline{m}_2,$ $\alpha=\underline{m}_1*\underline{m}_2-(\underline{m}_1-\alpha_1)(\underline{m}_2-\alpha_2),$ $\beta=(\overline{m}_1+\beta_1)(\overline{m}_2+\beta_2)-\overline{m}_1*\overline{m}_2$
$A=\tilde{A}_1(/)\tilde{A}_2$	$\overline{m}=\overline{m}_1/\underline{m}_2,\quad \underline{m}=\underline{m}_1/\overline{m}_2,$ $\alpha=(\underline{m}_1\beta_2+\overline{m}_2\alpha_1)/(\overline{m}_2^2+\overline{m}_2\beta_2),$ $\beta=(\underline{m}_2\beta_1+\overline{m}_1\alpha_2)/(\underline{m}_2^2-\underline{m}_2\alpha_2)$

На основі приведених вище описів арифметичних операцій можна для даного прикладу визначити оцінку бюджету без врахування боргів (F) як суму трьох джерел фінансування. Причому результат буде також нечіткою змінною

$F = A(+)B(+)C =$(50+100+0, 70+110+0, 10+0+0, 30+0+20) = (150, 180, 10, 50)

з трапецієвидною функцією приналежності (рис. 6.19). Для здобуття повної оцінки передбачуваного бюджету необхідно з отриманого результату відняти передбачувані плати по кредитах. При цьому бюджет з врахуванням боргів (F_1) також буде нечіткою змінною (рис. 6.20):

$F_1 = D(-)D =$(150 - 80, 180 - 80, 10 + 20, 50 + 30) = (70, 100, 30, 80).



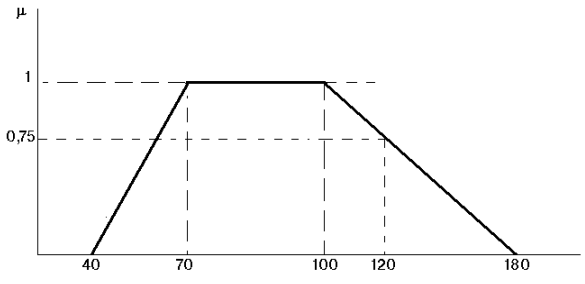


Рис. 6.19. Бюджет без врахування боргу. Рис. 6.20. Бюджет з боргом.

Таким чином, в бюджеті може бути сума від 40 до 180 млн. грн., але з найбільшою мірою упевненості можна говорити про суми від 70 до 100 млн. грн.

Існують процедури по обчисленню деякої чіткої функції $H(A,B)$ від нечітких аргументів, які називаються індексом ранжирування. Значення індексу для конкретної пари чисел дає підставу вирішити питання про те, яке з двох нечітких чисел більше (або з яким ступенем більше). Приведемо приклад індексу ранжирування:

$$H(A,B) = H_+(A) - H_+(B),\ H_+(A) = \int_0^1 M(A_0)dA,$$

де A_0 - α-рівнева підмножина нечіткої множини А. При цьому, якщо $H(A,B) \geq 0$, то $A \geq B$.

Розглянемо приклад застосування індексу ранжирування. Два літака протиборчих повітряних армій керуються стратегіями:

А: Якщо снарядів мало, то ймовірність поразки супротивника мала, інакше не мала.

В: Якщо снарядів не мало, то ймовірність поразки супротивника велика, інакше не велика. Відомо, що

мало снарядів = $A = (0.8/3,\ 0.4/15,\ 0.3/30)$,

мала ймовірність = $B = (0.1/0.9,\ 0.5/0.5,\ 0.8/0.1)$,

велика ймовірність = $C = (0.8/0.9,\ 0.5/0.5,\ 0.3/0.2)$.

Кількість снарядів не дуже мала. Хто переможе? Визначимо всі необхідні для вирішення задачі нечіткі множини:

не мало снарядів = $\overline{A} = (0.2/3, 0.6/15, 0.3/30)$,

не мала ймовірність = $\overline{B} = (0.9/0/9, 0.5/0.5, 0.2/0.1)$,

не велика ймовірність = $\overline{C} = (0.2/0.9, 0.5/0.5, 0.7/0.2)$,

$x =$ не дуже мало = $\overline{(мало)^2}$, $(мало)^2 = (0.64/3, 0.16/15, 0.09/30)$,

$\overline{(мало)^2} = (0.36/3, 0.84/15, 0.91/30)$.

Визначимо нечітке відношення стратегії А:

$$R_1 = A \times B \cup \overline{A} \times \overline{B} = \begin{vmatrix} 0.2 & 0.5 & 0.8 \\ 0.6 & 0.5 & 0.4 \\ 0.3 & 0.3 & 0.3 \end{vmatrix}$$

$$y_1 = x \circ R_1 = 0.36 \quad 0.84 \quad 0.91 \circ \begin{vmatrix} 0.2 & 0.5 & 0.8 \\ 0.6 & 0.5 & 0.4 \\ 0.3 & 0.3 & 0.3 \end{vmatrix} = (0.6/0.9, 0.5/0.5, 0.4/0.1)$$

Визначимо нечітке відношення стратегії В:

$$R_2 = \overline{A} \times C \cup A \times \overline{C} = \begin{vmatrix} 0.16 & 0.4 & 0.56 \\ 0.48 & 0.3 & 0.28 \\ 0.24 & 0.15 & 0.21 \end{vmatrix}$$

$$y_2 = x \circ R_2 = 0.36 \quad 0.84 \quad 0.91 \circ \begin{vmatrix} 0.16 & 0.4 & 0.56 \\ 0.48 & 0.3 & 0.28 \\ 0.24 & 0.15 & 0.21 \end{vmatrix} = (0.48/0.9, 0.36/0.5, 0.36/0.2)$$

Порівняємо отримані результати y_1 та y_2 між собою, для чого скористаємося індексом ранжирування $H(y_1, y_2)$.

$$H_+(y_1) = 0.4 \cdot (0.1 + 0.9)/2 + 0.5 \cdot (0.5 + 0.9)/2 + 0.6 \cdot (0.9 + 0.9)/2 = 1.09$$

$$H_+(y_2) = 0.36 \cdot (0.2 + 0.9)/2 + 0.48 \cdot (0.9 + 0.9)/2 = 0.63$$

$$H(y_1, y_2) = 1.09 - 0.63 = 0.46 > 0$$

Таким чином, літак зі стратегією А переможе.

Важливим моментом в теорії нечіткої логіки є застосування нечіткого логічного виводу в умовах невизначеної інформації.

Означення 22. *Нечітким логічним виводом* (fuzzy logic inference) називається апроксимація залежності $Y = f(X_1, X_2, ..., X_n)$ кожної вихідної лінгвістичної змінної від вхідних лінгвістичних змінних і здобуття висновку у вигляді нечіткої множини, відповідної поточним значенням входів, з використанням нечіткої бази знань і нечітких операцій. Основу нечіткого логічного виводу складає *композиційне правило* Заде.

Означення 23. *Композиційне правило виводу* Заде формулюється таким чином: якщо відоме нечітке відношення $\tilde{R}$ між вхідною (x) і вихідною (y)

змінними, то при нечіткому значенні вхідної змінної $x = \tilde{A}$, нечітке значення вихідний змінної визначається як $y = \tilde{A} \circ \tilde{R}$, де «$\circ$» - максміна композиція.

Означення 24. *Нечіткою базою знань* називається сукупність нечітких правил «Якщо – то», що визначають взаємозв'язок між входами і виходами досліджуваного об'єкту. Узагальнений формат нечітких правил такий: «**Якщо** *посилка правила*, **то** *висновок правила*».

Посилка правила або антецедент є твердженням типа «x є низький», де «низький» - це терм (лінгвістичне значення), заданий нечіткою множиною на універсальній множині лінгвістичної змінної x. Квантифікатори «дуже», «більш-менш», «не», «майже» і тому подібне можуть використовуватися для модифікації термів антецедента.

Висновок або наслідок правила є твердженням типа «y є d", в якому значення вихідний змінної (d) може задаватися:

- нечітким термом: «y є високий»;
- класом рішень: «y є бронхіт»;
- чіткою константою: «$y = 5$»;
- чіткою функцією від вхідних змінних: «$y = 5 + 4x$».

Розглянемо такий приклад: нечітка база знань описує залежність між віком водія (x) і можливістю дорожньо-транспортного випадку (y):

Якщо x = Молодий, **то** y = Висока;

Якщо x = Середній, **то** y = Низька;

Якщо x = Дуже старий, **то** y = Висока.

Хай функції приналежності термів мають вигляд, показаний на рис. 6.21. Тоді нечіткі відношення, відповідні правилам бази знань, будуть такими, як на рис. 6.22.

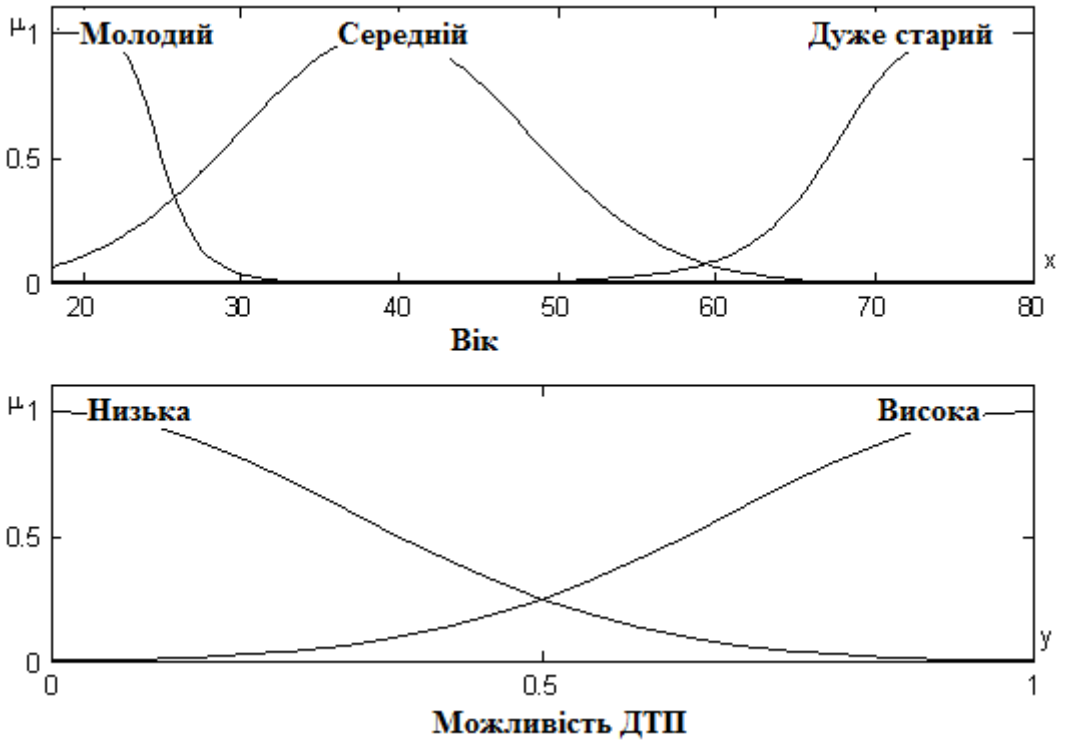


Рис. 6.21. Функції приналежності термів.

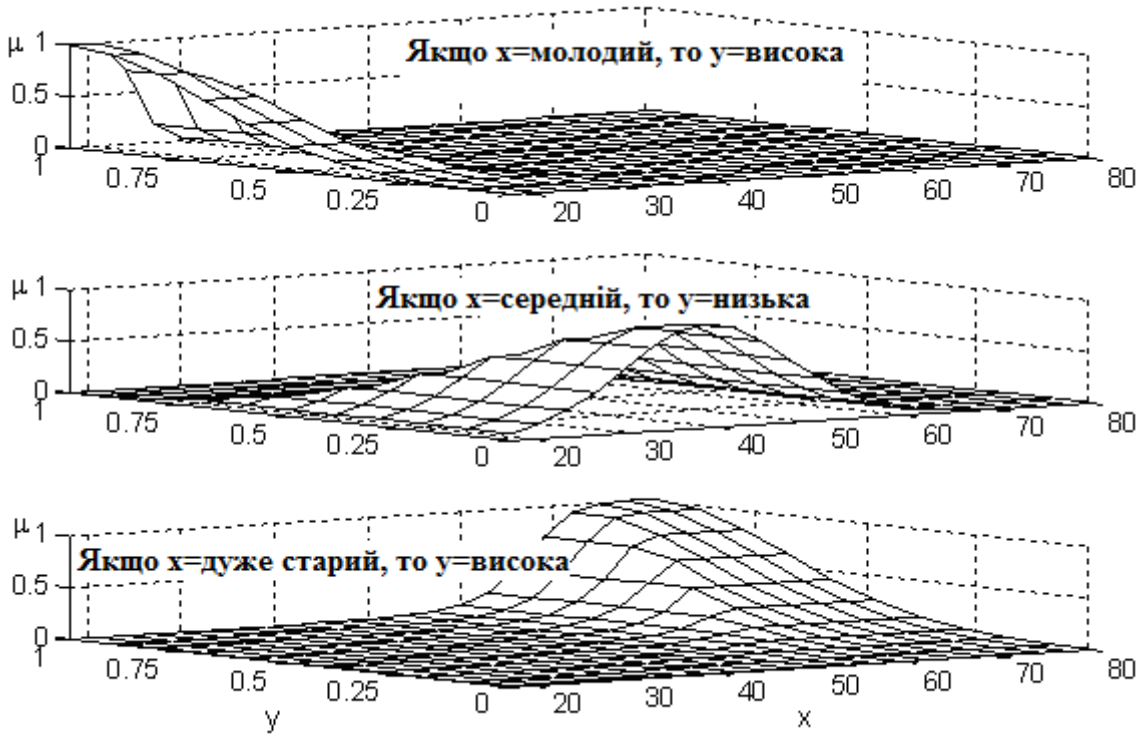


Рис. 6.22. Нечіткі відношення, відповідні правилам бази знань.

У загальному випадку нечітке виведення рішення відбувається за три (або чотири) кроки:

1. *Етап фаззифікації.* За допомогою функцій приналежності всіх термів вхідних лінгвістичних змінних і на підставі чітких значень, що задаються, з універсумів вхідних лінгвістичних змінних визначаються міри упевненості в тому, що вихідна лінгвістична змінна набуває конкретного значення. Ця міра упевненості є ордината точки перетину графіка функції приналежності терма і прямої $x =$ чітке значення лінгвістичної змінної.

2. *Етап безпосереднього нечіткого виводу*. На підставі набору правил - нечіткої бази знань - обчислюється значення істинності для передумови кожного правила на підставі конкретних нечітких операцій, відповідної кон'юнкції або диз'юнкції термів в лівій частині правил. В більшості випадків це або максимум, або мінімум із ступенів упевненості термів, обчислених на етапі фаззифікації, який застосовується до висновку кожного правила. Використовуючи один із способів побудови нечіткої імплікації, отримаємо нечітку змінну, відповідну обчисленому значенню ступеню упевненості в лівій частині правила і нечіткій множині в правій частині правила.

Зазвичай в якості виводу використовується мінімізація або правила продукції. При мінімізуючому логічному виводі вихідна функція приналежності обмежена зверху відповідно до обчисленого ступеню істинності посилки правила (нечітке логічне І). У логічному виводі з використанням продукції вихідна функція приналежності масштабується за допомогою обчисленого ступеню істинності передумови правила.

3. *Етап композиції (агрегації, акумуляції).* Всі нечіткі множини, призначені для кожного терма кожної вихідної лінгвістичної змінної, об'єднується разом, і формується єдина нечітка множина - значення для кожної лінгвістичної змінної, що виводиться. Зазвичай використовуються функції MAX або SUM.

4. Етап дефаззифікації (необов'язковий). Використовується тоді, коли корисно перетворити нечіткий набір значень лінгвістичних змінних, що виводяться, до точних. Є чимала кількість методів переходу до точних значень

(принаймні, 30). Два приклади загальних методів – «методи повної інтерпретації» і «по максимуму». У методі повної інтерпретації точне значення змінної, що виводиться, обчислюється як значення «центру тяжіння» функції приналежності для нечіткого значення. У методі максимуму як точне значення змінної, що виводиться, приймається максимальне значення функції приналежності.

У теорії нечітких множин процедура дефаззифікації аналогічна знаходженню характеристик положення (математичного чекання, моди, медіани) випадкових величин в теорії ймовірностей. Простим способом виконання процедури дефаззифікації є вибір чіткого числа, відповідного максимуму функції приналежності. Проте придатність цього способу поширюється лише на одноекстремальні функції приналежності. Для багатоекстремальних функцій приналежності часто використовуються наступні методи дефаззифікації:

- COG (Center Of Gravity) – «центр тяжіння». Фізичним аналогом цієї формули є знаходження центру тяжіння плоскої фігури, обмеженої осями координат і графіком функції приналежності нечіткої множини.
- MOM (Mean Of Maximums) – «центр максимумів». При використанні методу центру максимумів потрібно знайти середнє арифметичне елементів універсальної множини, що мають максимальні ступені приналежності.
- First Maximum – «перший максимум» - максимум функції приналежності з найменшою абсцисою.

Серед найбільш відомих механізмів логічного виводу слід відмітити нечіткий логічний вивід по алгоритму Мамдані, по алгоритму Сугено, синглтонну модель нечіткого логічного виводу, ієрархічні системи нечіткого логічного виводу, адаптивну мережа нечіткого виводу та деякі інші.

Розглянемо більш докладніше алгоритм нечіткого виводу на конкретному прикладі. Хай у нас є деяка система, наприклад, реактор, описувана трьома параметрами: температура, тиск і витрата робочої речовини. Всі показники вимірювані, і множина можливих значень відома. Також з

досвіду роботи з системою відомі деякі правила, що зв'язують значення цих параметрів. Передбачимо, що зламався датчик, що вимірює значення одного з параметрів системи, але знати його свідчення необхідно хоч би приблизно. Тоді встає завдання про відшукання цього невідомого значення (хай це буде тиск) при відомих показниках двох інших параметрів (температури і витрати) і зв'язку цих величин у вигляді наступних правил:

Якщо Температура низька і Витрата мала, **то** Тиск низький;

Якщо Температура середня, **то** Тиск середній;

Якщо Температура висока або Витрата велика, **то** Тиск високий.

У нашому випадку Температура, Тиск і Витрата - лінгвістичні змінні. Опишемо кожну з них.

Температура. Універсум (множина можливих значень) - відрізок [0, 150]. Початкова множина термів {Висока, Середня, Низька}. Функції приналежності термів мають наступний вигляд (рис. 6.23):

Рис. 6.23. Лінгвістична змінна «Температура».

Тиск. Універсум - відрізок [0, 100]. Початкова множина термів {Високий, Середній, Низький}. Функції приналежності термів мають наступний вигляд (рис. 6.24):

Рис. 6.24. Лінгвістична змінна «Тиск».

Витрата. Універсум - відрізок [0, 8]. Початкова множина термів {Велика, Середня, Мала}. Функції приналежності термів мають наступний вигляд (рис. 6.25):

Рис. 6.25. Лінгвістична змінна «Витрата робочої речовини».

Хай відомі значення: «Температура» - 85 і «Витрата» - 3,5. Виконаємо розрахунок значення тиску.

Послідовно розглянемо етапи нечіткого виводу. Спочатку по заданих значеннях вхідних параметрів знайдемо ступені упевненості простих тверджень вигляду «Лінгвістична змінна *A* є Терм Лінгвістичної змінної *A*». Цей етап називається фаззифікацією, тобто переходом від заданих чітких значень до ступенів упевненості. Отримуємо наступні ступені упевненості:

- Температура Висока - 0,7;
- Температура Середня - 1;
- Температура Низька - 0,3;
- Витрата Велика - 0;
- Витрата Середня - 0,75;
- Витрата Мала - 0,25.

Потім обчислимо ступені упевненості посилок правил:

- Температура Низька і Витрата Мала: min (Температура Низька, Витрата Мала) = min (0.3, 0.25) = 0.25;
- Температура Середня: 1;
- Температура Висока або Витрата Велика: max (Температура Висока, Витрата Велика) = max (0.7,0) = 0,7.

Слід зазначити також той факт, що за допомогою перетворень нечіткої множини будь-яке правило, що містить в лівій частині як кон'юнкції, так і диз'юнкції, можна привести до системи правил, в лівій частині кожного будуть або лише кон'юнкції, або лише диз'юнкції. Таким чином, не зменшуючи загальності, можна розглядати правила, що містять в лівій частині або лише кон'юнкції, або лише диз'юнкції.

Кожне з правил представляє із себе нечітку імплікацію. Ступінь упевненості посилки ми обчислили, а ступінь упевненості висновку задається функцією приналежності відповідного терма. Тому, використовуючи один із способів побудови нечіткої імплікації, отримаємо нову нечітку змінну, відповідну ступеню упевненості в значенні вихідних даних при застосуванні до заданих вхідних відповідного правила. Використовуючи визначення нечіткої імплікації як мінімуму лівої і правої частин (визначення Мамдані), маємо (рис. 6.26):

Рис. 6.26. Нечітка база знань.

Тепер необхідно об'єднати результати вживання всіх правил. Цей етап називається акумуляцією. Один з основних способів акумуляції - побудова максимуму отриманих функцій приналежності (рис. 6.27). Отриману функцію приналежності вже можна вважати результатом. Це новий терм вихідної змінної «Тиск». Його функція приналежності говорить про ступінь упевненості в значенні тиску при заданих значеннях вхідних параметрів і використанні правил, що визначають співвідношення вхідних і вихідних змінних. Але зазвичай все-таки необхідне якесь конкретне числове значення. Для його

здобуття використовується етап дефаззифікації, тобто набуття конкретного значення з універсуму по заданій на ньому функції приналежності.

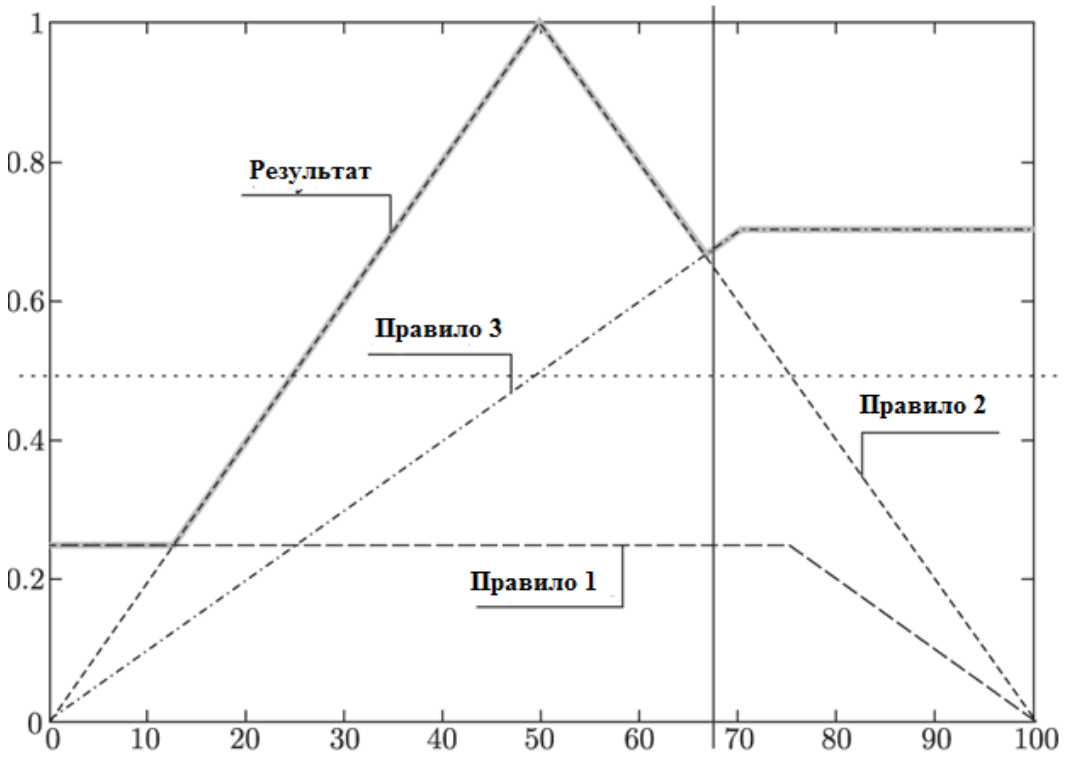


Рис. 6.27. Об'єднання результатів вживання правил.

Існує багато методів дефаззифікації, але в нашому випадку досить методу першого максимуму. Застосовуючи його до отриманої функції приналежності, отримуємо, що значення тиску - 50. Таким чином, якщо ми знаємо, що температура дорівнює 85, а витрата робочої речовини - 3,5, то можемо зробити висновок, що тиск в реакторі дорівнює приблизно 50.

Різні поняття, нечіткі за своєю природою, можуть бути формально описані за допомогою нечіткої множини. Нечітка логіка, наприклад, дозволяє формалізувати прості логічні зв'язки нечітких змінних за допомогою нечітких висловів. Для опису ж складних співвідношень між змінними зручно використовувати *нечіткі алгоритми*.

Під алгоритмом розуміється точно визначене правило дій (програма), для якого задана вказівка, як і в якій послідовності це правило необхідно застосовувати до вихідних даних задачі, аби отримати її рішення. Характеристиками алгоритму є:

- детермінованість - однозначність результату процесу при незмінних вихідних даних;
- дискретність визначуваного алгоритмом процесу - розчленованість його на окремі елементарні дії, можливість виконання яких людиною або машиною не викликає сумніву;
- масовість - вихідні дані для алгоритму можна вибрати з деякої множини даних, тобто алгоритм повинен забезпечити рішення будь-якої задачі з класу однотипних задач.

Нечіткий же алгоритм визначається *впорядкованою множиною нечітких інструкцій (нечітких висловів), які містять поняття, що формалізуються нечіткими множинами.* Під нечіткими інструкціями розуміються інструкції, що містять нечітке поняття, наприклад, «пройти близько 100 метрів», а під машинними - інструкції, що не містять жодних нечітких понять: «пройти 100 метрів». Чіткі інструкції ми називатимемо машинними, аби підкреслити можливість моделювання нечітких алгоритмів на ЕОМ, що сприймають лише читання інструкцій.

Приведемо точне визначення нечіткого алгоритму. Для формулювання необхідно ввести ряд первинних визначень і позначень. По-перше, замість інтервалу [0, 1], загальноприйнятої множини значень функції приналежності, розглядається непорожня множина W з відношенням часткового порядку $>$ і операціями $\otimes,\oplus$, що задовольняють властивостям комутативності, асоціативності і дистрибутивності, а також містять нульовий і одиничний елементи.

По-друге, розглядаються інструкції наступного вигляду

Start: go to L (інструкція початку);

L: do F, go to L_1 (інструкція операції);

L: if P then go to $(L_1,...,L_n)$ (інструкція умови);

L: halt (інструкція закінчення),

де $(L_1,...,L_n) \in L$ - множина символів міток інструкцій, $f \in F$ - символ оператора або функції, $P \in P$ - символ предикатів або умов.

Введення поняття інструкції дозволяє визначити поняття програми. Під програмою розуміється кінцева множина інструкцій π, що містить єдину інструкцію початку. Жодні інструкції з π не мають однакових міток.

По-третє, визначається поняття W-машини. W-машина є функція M, визначена на множині символів $\{O\} \cup \{I\} \cup F \cup P$, для яких існують множина входів X, множина станів пам'яті M і множина виходів Y, а також виконані наступні умови:

$M(I): X \times M \to W$ (функція входів);

$\forall F \in F \;\; M(F): M \times M \to W$ (функція операції);

$\forall P \in P,\; n > 0 \;\; M(P): M \times \{1,...,n\} \to W$ (функція умов);

$M(O): M \times Y \to W$ (функція виходу).

Символи I та O позначають вхід і вихід. Нарешті, по-четверте, програма π разом з W-машиною, яка допускає π, називається *нечіткою програмою*. Отже, послідовністю інструкцій, складаючих нечітку програму, називається *нечітким алгоритмом*. Конкретні типи алгоритмів можуть бути отримані за допомогою вибору множин $\{W, M, X, Y\}$, функцій (входів, дій, умов, виходів), операцій $\{\otimes, \oplus\}$, відношення $>$.

Приклад. Хай є послідовність інструкцій для водія автомобіля і карта місцевості. Водієві пропонується знайти місце призначення, використовуючи карту і послідовність нечітких інструкцій, що описують маршрут. Для простоти викладу припустимо, що всі точки на плоскості мають тільки цілочисельні координати. Типові інструкції для водія: «рухатися прямо біля L метрів», «повернути наліво», «повернути направо», «рухатися прямо до тих пір, поки не побачиш ...».

Сконструюємо відповідну W-машину M. Така машина має множину станів пам'яті M у вигляді впорядкованих трійок $(a, b, \bar{u})$, де (a, b) - точка на площині, відповідна місцезнаходженню автомобіля, $\bar{u}$ - одиничний вектор

напряму руху автомобіля. Множина входів $X = M$ і множина виходів Y складаються з впорядкованих пар (a,b); M_I - функція входів, відповідає тождественій функції; M_O - функція виходів, відповідає функції, що відображає кожну трійку $(a,b,\bar{u})$ в (a,b).

Машина M не має жодної функції умови. Кожній інструкції, приведеній вище, відповідає функція операції. При цьому i інструкція в послідовності інструкцій може бути перетворена в інструкцію операції виду «$do\ \ F_i;\ goto\ \ L_i$». Сукупність таких інструкцій і інструкцій «$start : goto\ \ L_0$» та «$L_n : halt$», де n - довжина послідовності, складає програму π. Процес виконання програми π на машині M визначається послідовністю інструкцій і картою місцевості. Стислості ради приведемо тільки функцію операції для інструкції типу «рухатися прямо біля L метрів»:

$$M_{F_L} = ((a_2,b_2,\bar{u}_2)\big|(a_1,b_1,\bar{u}_1)) = f_L(\sqrt{(a_2-a_1)^2+(b_2-b_1)^2}) \times G((a_2,b_2,\bar{u}_2)\big|(a_1,b_1,\bar{u}_1)),$$

де $f_L(d)$ - ступінь відповідна відстані d, $\left|G((a_2,b_2,\bar{u}_2)\right|(a_1,b_1,\bar{u}_1))$ - ступінь, відповідна твердженню: «точка (a_2,b_2) і напрям $\bar{u}_2$ досяжні при русі прямо з точки (a_1,b_1) по напряму $\bar{u}_1$.

Розглянуті нечіткі теорії мають тісну інтеграцію з інтелектуальними парадигмами. Подібно до того, як нечіткі множини розширили рамки класичної математичної теорії множин, нечітка логіка «вторглася» практично в більшість методів Data Mining наділивши їх новою функціональністю.

Нечіткі нейронні мережі. Нечіткі нейронні мережі (fuzzy-neural networks) здійснюють висновки на основі апарату нечіткої логіки, проте параметри функцій приналежності настроюються з використанням алгоритмів навчання нейронних мереж. Тому для підбору параметрів таких мереж застосовується метод зворотного розповсюдження помилки, спочатку запропонований для навчання багатошарового персептрона. Для цього модуль нечіткого управління представляється у формі багатошарової мережі. Нечітка нейронна мережа як правило складається з чотирьох шарів: шару фаззифікації

вхідних змінних, шару агрегації значень активації умови, шару агрегації нечітких правил і вихідного шару. Найбільшого поширення в даний час набула архітектура нечіткої нейронної мережі виду ANFIS і TSK. Доведено, що такі мережі є універсальними апроксиматорами. Швидкі алгоритми навчання і інтерпретуємість накопичених знань – ці чинники зробили сьогодні нечіткі нейронні мережі одним з найперспективніших і ефективніших інструментів м'яких обчислень.

Адаптивні нечіткі системи. Класичні нечіткі системи володіють тим недоліком, що для формулювання правил і функцій приналежності необхідно застосовувати знання експертів тієї або іншої предметної області, що не завжди вдається забезпечити. Адаптивні нечіткі системи (adaptive fuzzy systems) вирішують цю проблему. У таких системах підбір параметрів нечіткої системи проводиться в процесі навчання на експериментальних даних. Алгоритми навчання адаптивних нечітких систем відносно трудомісткі і складні в порівнянні з алгоритмами навчання нейронних мереж, і, як правило, складаються з двох стадій: генерації лінгвістичних правил і корегування функцій приналежності. Перша задача відноситься до задач переборного типу, друга – до оптимізації в безперервних просторах. При цьому виникає певна суперечність: для генерації нечітких правил необхідні функції приналежності, а для проведення нечіткого виводу – правила. Крім того, при автоматичній генерації нечітких правил необхідно забезпечити їх повноту і несуперечність. Значна частина методів навчання нечітких систем використовує генетичні алгоритми. У англомовній літературі цьому відповідає спеціальний термін – Genetic Fuzzy Systems. Значний внесок в розвиток теорії і практики нечітких систем з еволюційною адаптацією внесла група іспанських дослідників на чолі з Ф. Херрера (F. Herrera).

Нечіткі запити. Нечіткі запити до баз даних (fuzzy queries) - перспективний напрям в сучасних системах обробки інформації. Даний інструмент дає можливість формулювати запити на природній мові, наприклад: «Вивести список недорогих пропозицій про знімання житла близько до центру

міста», що неможливе при використанні стандартного механізму запитів. Для цієї мети розроблена нечітка реляційна алгебра і спеціальні розширення мов SQL для нечітких запитів. Велика частина досліджень в цій області належить західноєвропейським ученим Д. Дюбуа и Г. Праде.

Нечіткі асоціативні правила. Нечіткі асоціативні правила (fuzzy associative rules) – інструмент для витягання з баз даних закономірностей, які формулюються у вигляді лінгвістичних висловів. Тут введені спеціальні поняття нечіткої транзакції, підтримки і достовірності нечіткого асоціативного правила.

Нечіткі когнітивні карти. Нечіткі когнітивні карти (fuzzy cognitive maps) були запропоновані Б. Коско в 1986 р. і використовуються для моделювання причинних взаємозв'язків, виявлених між концептами деякої області. На відміну від простих когнітивних карт, нечіткі когнітивні карти є нечіткий орієнтований граф, вузли якого є нечіткою множиною. Направлені ребра графа не лише відображають причинно-наслідкові зв'язки між концептами, але і визначають міру впливу зв'язуваних концептів. Активне використання нечітких когнітивних карт як засіб моделювання систем обумовлене можливістю наочного представлення аналізованої системи і легкістю інтерпретації причинно-наслідкових зв'язків між концептами. Основні проблеми пов'язані з процесом побудови когнітивної карти, який не піддається формалізації. Крім того, необхідно довести, що побудована когнітивна карта адекватна реальній системі. Для вирішення даних проблем розроблені алгоритми автоматичної побудови когнітивних карт на основі вибірки даних.

Нечітка кластеризація. Нечіткі методи кластеризації, на відміну від чітких методів (наприклад, нейронні мережі Кохонена), дозволяють одному і тому ж об'єкту належати одночасно декільком кластерам, але з різною мірою. Нечітка кластеризація в багатьох ситуаціях «природніша», ніж чітка, наприклад, для об'єктів, розташованих на кордоні кластерів. Найбільш поширені: алгоритм нечіткої самоорганізації c-means і його узагальнення у вигляді алгоритму Густафсона-Кесселя. Список можна продовжити і далі:

нечіткі дерева рішень, нечіткі мережі Петрі, нечітка асоціативна пам'ять, нечіткі самоорганизующі карти і інші гібридні методи.

6.3. Програмне забезпечення нечітких методів

Математична теорія нечітких множин будучи предметом інтенсивних досліджень відкриває досить великі можливості перед системними аналітиками. Засновані на цій теорії різні комп'ютерні системи, у свою чергу, істотно розширюють сферу застосування нечіткої логіки. В даний час найбільший інтерес, як на Україні, так і за кордоном, викликає область нечіткого інтелектуального аналізу даних як одна з найрезультативніших сфер застосування теорії нечітких множин. Ключовими перевагами нечіткої логіки в порівнянні з іншими технологіями інтелектуального аналізу є: по-перше, при тих же об'ємах вхідної і вихідної інформації, центральний блок ухвалення рішень стає компактніше і простіше для сприйняття людиною, по-друге, рішення складної і громіздкої задачі обчислення точних дій підміняється значно більш простою і гнучкою стратегією адаптивного підстроювання - при збереженні необхідної точності результату [5, 25, 56, 77].

Розвиток додатків з нечіткою логікою відбувається не лише на логічному рівні. У 1986 році в AT&T Bell Labs створювалися процесори з «прошитою» нечіткою логікою обробки інформації. У Європі і США ведуться інтенсивні роботи по інтеграції fuzzy команд в асемблери промислових контролерів вбудованих пристроїв (чіпи Motorola). Крім того, розробляються різні варіанти fuzzy- співпроцесорів, які контактують з ЦП через загальну шину даних, і концентрують свої зусилля на розмиванні ущільненні інформації та оптимізації використання правил (продукти Siemens Nixdorf). Та все ж по проведених аналізах вартості продукції дешевше «емулювати» нечітку логіку, хоча складним питанням є трансляція fuzzy-системи на традиційну мову програмування. Компанія Aptronix пропонує використовувати мову Java, що

має все необхідне для досить адекватного відтворення інструкцій нечіткої логіки додатка методами мови. Крім того, використання Java API відкриває нові перспективи для дослідження fuzzy-систем у взаємодії. Internet, як глобальне середовище поширення Java-додатків, ідеально підходить для інтеграції прикладних пристроїв, створених за допомогою алгоритмів нечіткої логіки. Включення JavaBeans і Java сервлетів в комплекти розробника додатків представляє широкі можливості по програмуванню бізнес – логіки і обробки даних, уникаючи залежності від клієнтської операційної системи і призначеного для користувача інтерфейсу.

Сьогодні на світовому ринку програмних ресурсів представлено більше 100 пакетів, що в тому або іншому вигляді використовують нечітку логіку. У трьох десятках СУБД реалізована функція нечіткого пошуку. Власні програми на основі нечіткої логіки анонсували такі гіганти як IBM, Oracle та інші. Небагато знають, що нечіткій логіці зобов'язано своїм народженням і нове покоління систем імітаційного моделювання. Більшість програмних комплексів, що використовуються в світі для економічного, політичного і фінансового моделювання, базуються на методах т.з. динаміки систем («system dynamics»). А остання, у свою чергу, використовує апарат нечітких когнітивних схем (FCM), запропонованих Коско на початку 80-х і вперше випробуваних «в бойових умовах» під час політичної кризи в Південній Африці. У ті дні уряд США встав перед необхідністю приймати важливі військово-політичні рішення в умовах неповноти і явної невірогідності інформації, що поступала із-за океану. Ціна можливої помилки була вельми висока - на карті стояла доля крупних американських інвестицій. Традиційні методи пасували перед складним неформалізованим задачами. І тоді була побудована когнітивна модель ситуації, заснована на якісних викладеннях аналітика Уїльямся (Williams). Виявилось, що заплутаний клубок причинно-наслідкових зв'язків укладається в компактну графову модель, верхній рівень якої містить всього дев'ять ключових елементів, що повністю визначають розвиток ситуації. З тих пір без систем когнітивного моделювання не обходиться жоден ситуаційний

центр військового і політичного керівництва західних країн. Першовідкривачем ринку став пакет iThink, що вже встиг продемонструвати свої можливості при моделюванні виборів Президента Росії. Модель розвитку політичної ситуації, зроблена за допомогою пакету iThink в середині травня і опублікована на початку червня, точно визначила співвідношення електорату основних претендентів - 34% проти 30% і передбачила результат другого туру виборів задовго до початку першого. Примітно, що відхилення результатів найпершого прогнозу, зробленого за півтора місяці до голосування, від точних відповідей «уклалися» в рамки погрішності методів, вживаних соціологічними службами. Досить вірогідно, що пакет iThink в недалекому майбутньому стане і першовідкривачем нового ринку України - ринку послуг з підвищення прибутковості бізнесу (т.з. BPR - Business Processing Reengineering).

Проте перераховані вище програми - це складні комплексні системи, що вимагають певних зусиль по освоєнню і налаштуванню. На іншому полюсі ринку знаходиться сімейство легких і компактних програм, заснованих на нечіткій алгебрі. Їх найбільш яскравим представником є пакет FuziCalc американської фірми FuziWare, який є порівняно молодим (1995 рік), але встиг завоювати популярність як недорогий інструмент, що дозволяє проводити швидкі (приблизні) розрахунки в різних областях бізнесу і отримувати результати із сповна прийнятною мірою точності. Зовні FuziCalc - це звичайна електронна таблиця, і доки проводяться точні обчислення, різниця неощутима. Проте традиційна електронна таблиця втратить працездатність при першому ж нечітко введеному значенні. FuziCalc пропонує інший шлях, значно простіший. Поля, значення яких відомі неточно, позначаються спеціальним значком (у FuziCalc це сірий трикутник). Самі значення в простому випадку представляються четвіркою чисел (мінімум, максимум і найбільш імовірний діапазон). Наприклад: «Зазвичай в магазині буває від 30 до 50 продажів в день, але ніколи не менше 10 і не більше 80». У графічному представленні такому вислову відповідає трапецієвидна функція розподілу (втім, пакет дозволяє описувати і значно складніші функції). Підсумковий результат обчислень буде

представлений подібними ж четвірками чисел, наприклад: «Завтрашній прибуток найімовірніше знаходитиметься в діапазоні 1050 - 1200 доларів, в найгіршому випадку - близько 800, в найкращому – 1200». Унікальна здатність пакету проводити швидкі оціночні обчислення без накопичення помилки міцно «прописала» його в арсеналі різних служб швидкого реагування - рятувальників з МНС і ін. Там, де вихідні дані неточні і неповні, а швидкість здобуття перших оцінок критична - нечітка алгебра практично не має альтернатив.

У складі пакету поставляються декілька прикладів вирішення задач. Робоче середовище пакету має виглядає: (рис. 6.28)

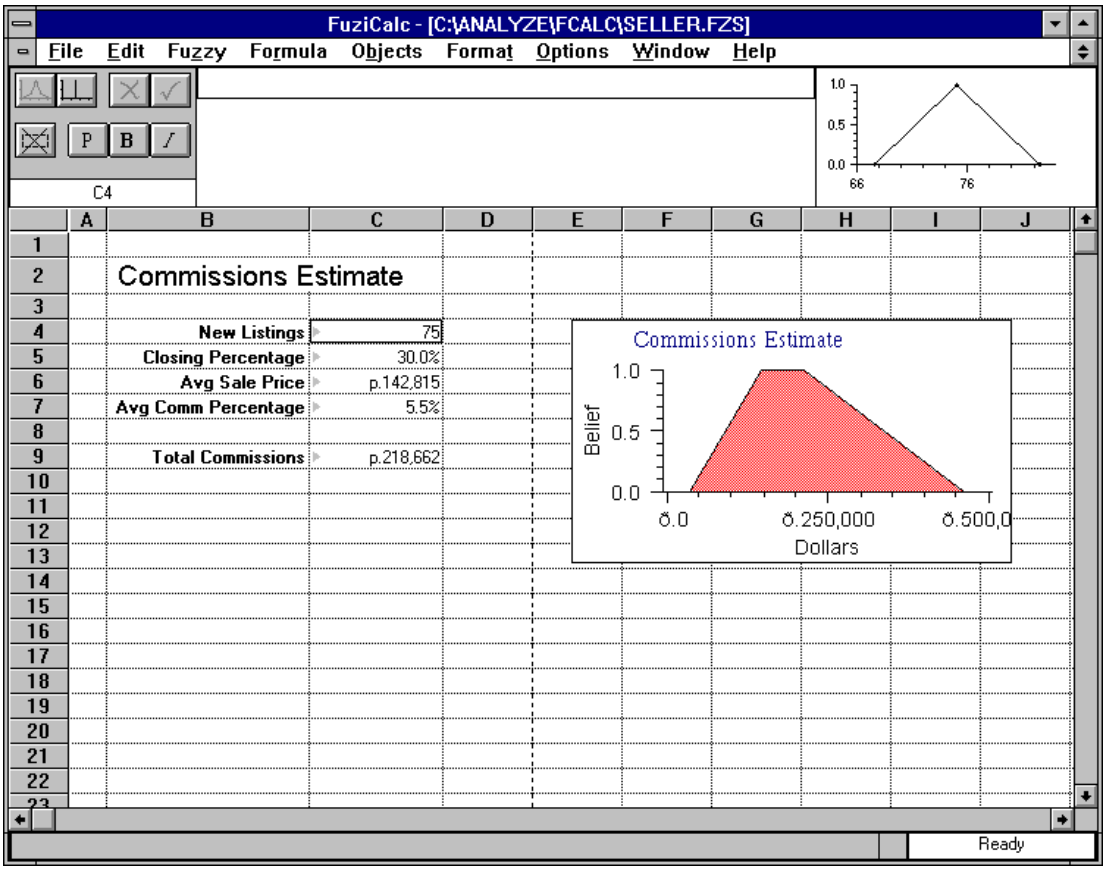


Рис. 6.28. Головне меню пакету FuziCalc.

На рисунку 6.28 показаний приклад рішення задачі оцінки прибутку при роботі фірми на ринку нерухомості. Постановка задачі така: менеджер планує діяльність фірми, що працює на ринку нерухомості наступного року. Цільова функція задачі – діапазон прибутку, на який можна розраховувати.

Для вирішення задачі існують чотири нечіткі твердження, виявлених із статистики діяльності фірми за декілька минулих років:

- Протягом кожного року у фірму приходять приблизно 75 потенційних клієнтів.
- З потенційних клієнтів приблизно 30% здійснюють операції.
- Вартість нерухомості, що фігурує в операціях, складає приблизно $ 142 815.
- За проведену операцію з кожного клієнта береться приблизно 5.5% комісійних.

Якщо просто перемножити ці чотири значення, ми отримаємо цифру $ 176 733. Така сума прибутку за рік, якщо її обчислювати звичайним способом. При цьому менеджер ясно усвідомлює, що:

- коли він говорить про 75 потенційних клієнтів в рік, то в різні роки виходить по-різному, але абсолютно точно відомо, що ніколи не бувало менше 67 і ніколи не було більше 87 (рис. 6.29).

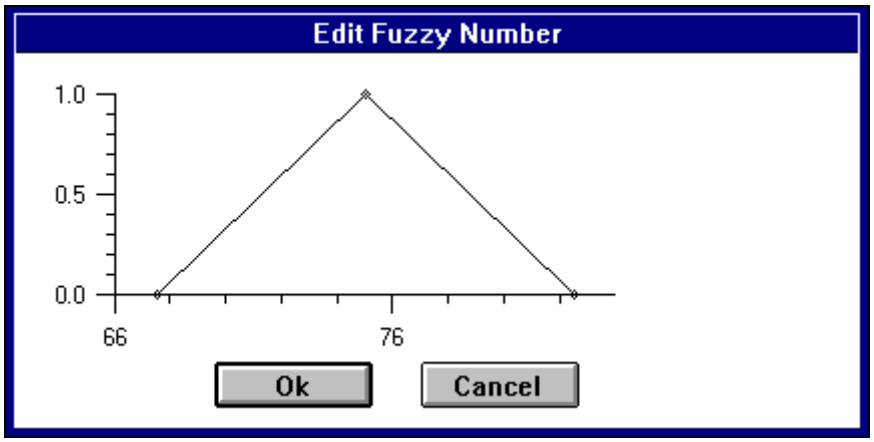


Рис. 6.29. Неточна функція клієнтів.

- коли він говорить про 30% потенційних клієнтів, що здійснюють операції, насправді це значення, як правило, змінюється в межах 25-35%, але ніколи не було менше 10% і ніколи не було більше 50 % (рис. 6.30).

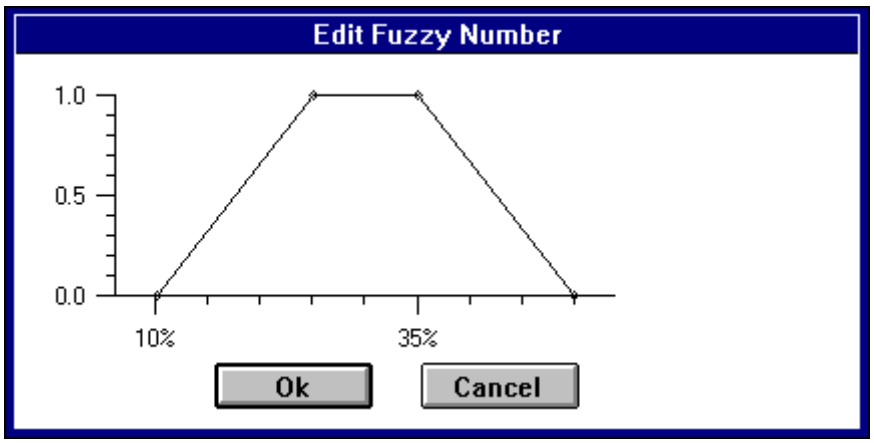


Рис. 6.30. Нечітка функція операцій.

- коли він говорить про вартість нерухомості в $ 142 815, то фактично вартість різних об'єктів операцій розподіляється таким чином: найчастіше відбуваються операції за ціною $ 138 000 – $ 140 000, але ніколи не дешевше $ 130 000 і не дорожче $ 150 000. Звернете увагу на злам графіка і відхід в точку, відповідну $ 160 000. Тут менеджер має на увазі, що наступного року фірма спробує проводити операції з дорожчою нерухомістю, але не більше $ 160 000 (рис. 6.31).

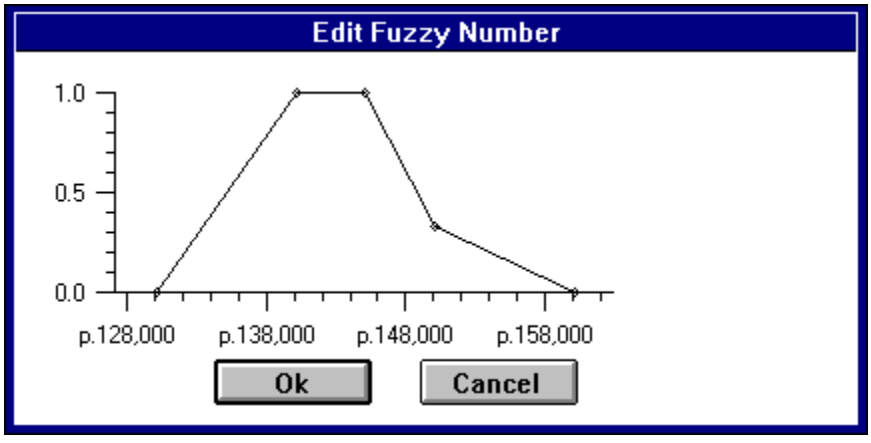


Рис. 6.31. Неточна функція вартості нерухомості.

- коли він говорить про комісійні в 5.5%, то насправді в різних випадках розмір комісійних розподіляється частішим всього в межах 5-6%, але не менше 4% і не більше 7% (рис. 6.32).

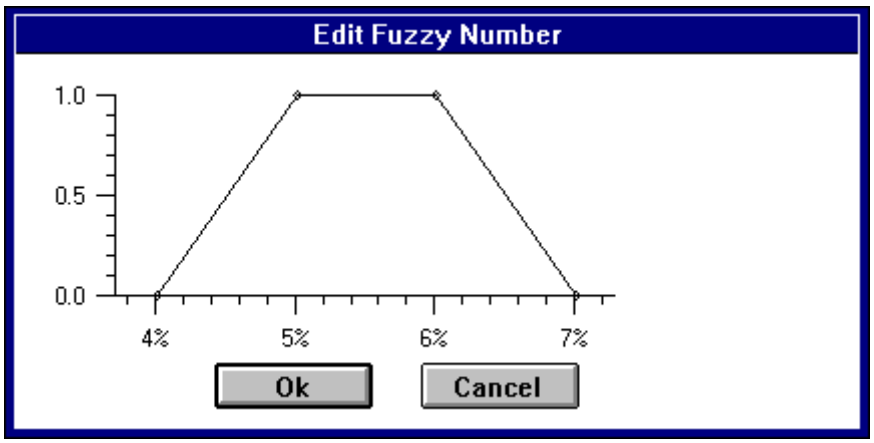


Рис. 6.32. Неточна функція комісійних виплат.

Система при такій постановці задачі результат обчислює перемножуванням інтегралів, що описують дані криві. Оскільки типи кривих відомі, число їх також відомо, то вживання 5 стандартних функцій, що описують ці криві робить обчислення практично миттєвими. У даному прикладі бачимо, що, побудувавши 4 нечітких твердження і перемноживши їх значення, відповідь можна побачити таку: «найбільш достовірне значення прибутку знаходитиметься в межах $ 150 000 – $ 200 000» (рис. 6.33).

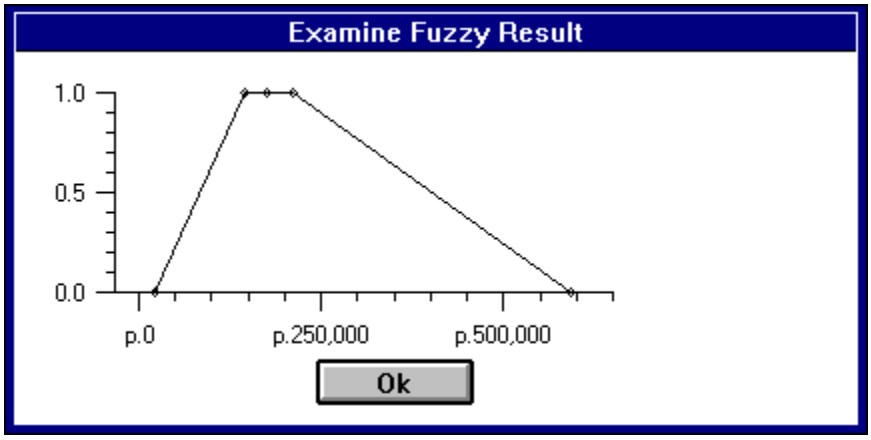


Рис. 6.33. Неточна функція прибутку.

Іншим відомим програмним продуктом є пакет CubiCalc фірми HiperLogic - управління динамічними системами. Ефективність вживання CubiCalc в задачах така, що відома організація КОКОМ (США), та, що свого

часу стежила за тим, аби нові американські технології в комп'ютерній області не підвищували чужий воєнно-промисловий потенціал, накладала дуже жорсткі обмеження на поширення цього пакету. Робоче середовище інструменту представлено на рисунку 6.34.

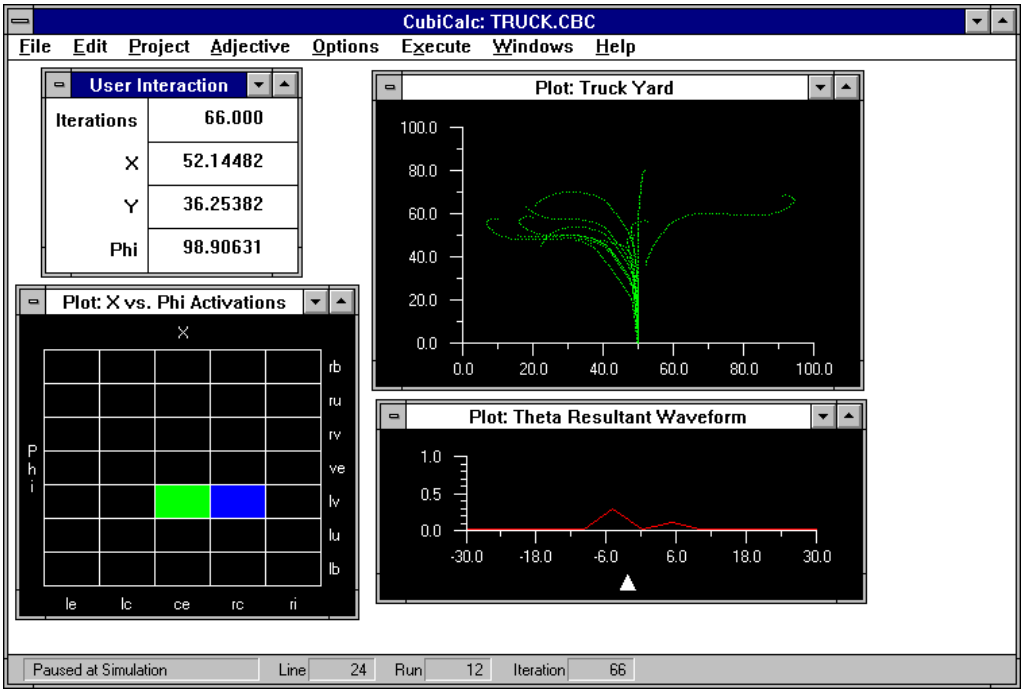


Рис. 6.34. Загальний вигляд пакету CubiCalc.

Розглянемо один з прикладів, що увійшов до всіх підручників нечіткої математики. Вантажівка заїжджає в гараж. Вантажівка досить довга, а гараж досить вузький (рис. 6.35). Кращі результати виходять, коли на підніжку трака встає помічник і, користуючись критеріями: «трохи лівіше… ще лівіше… стоп… тепер трохи назад і вправо», разом з водієм досягає бажаного результату. В принципі, задачу можна ускладнювати скільки завгодно: замість вантажівки це може бути важкий літак, замість гаража - другий літак, шуканий результат – заправка в повітрі, при входящій умови – бовтанка і так далі.

Нас цікавлять принципи, по яких може функціонувати подібна система, а також те, як ці принципи звести в деяку систему. Причому задача ставиться так, що будь-яка вантажівка з будь-якої точки двору за будь-яких умов повинна заїжджати у ворота, розташовані в будь-якому місці двору. Якщо задачу

вирішувати класичними засобами, то доведеться писати системи рівнянь, що описують вантажівку, гараж, можливі зовнішні чинники, алгоритми управління і інше.

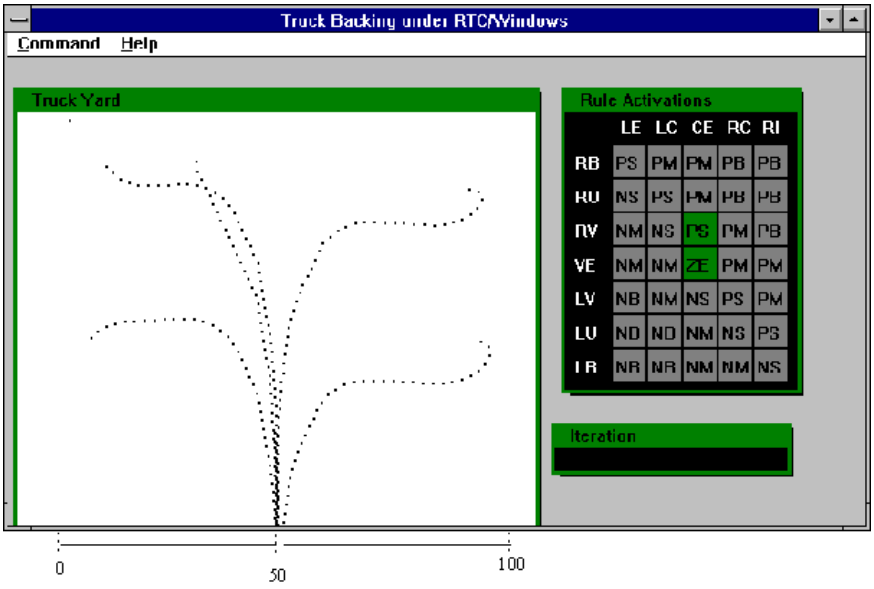


Рис. 6.35. Моделювання ситуації в пакеті CubiCalc.

Розглянемо, як таку задачу вирішує CubiCalc. По-перше, визначимо змінні, якими будемо оперувати. Для цього використаємо меню PROJECT/VARIABLES. На вході заміряємо «азимут вантажівки відносно воріт» (у градусах, 0.360) і її «зміщення відносно осі воріт» (у метрах, 0.100). Цільова функція – відхилення керма, причому таке, що на в'їзді до воріт вантажівка повинна рухатися строго уздовж їх осі. Тому на виході задаємо змінну «поворот керма» в градусах (- 30.+30). Оскільки, як вже було сказано, нам важливі принципи, для кожної змінної визначимо ряд нечітких розподілів, відповідним поняттям «лівіше-правіше» для азимута, відхилення керма і зміщення відносно воріт

В меню PROJECT/ADJECTIVES FOR це виглядатиме так:

- для азимута (мається на увазі, що азимут 90 градусів відповідає напряму на ворота) (рис. 6.36).

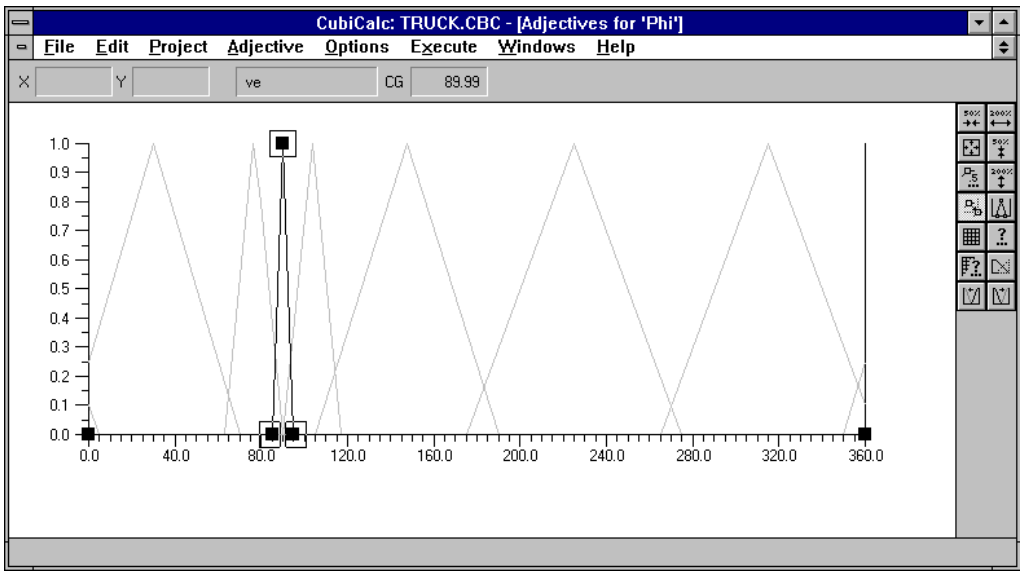


Рис. 6.36. Задання азимуту в пакеті CubiCalc.

Як видно зі схеми, всі значення, які лежать між 85 і 95 градусами азимута, відповідають критерію «точно прямо». Тут же описані розподіли значень азимута, відповідні критеріям «лівіше, вліво, глибоко вліво; правіше, вправо, глибоко вправо». Для кожного розподілу задається своя внутрішня (temporary) змінна – вони знадобляться пізніше, для створення правил.

- Для зміщення відносно воріт (мається на увазі, що значення 0.49 м відповідають «ліво», 51.100 м – «право». Значення 50 м- відповідає критерію «точно у ворота» (рис. 6.37).

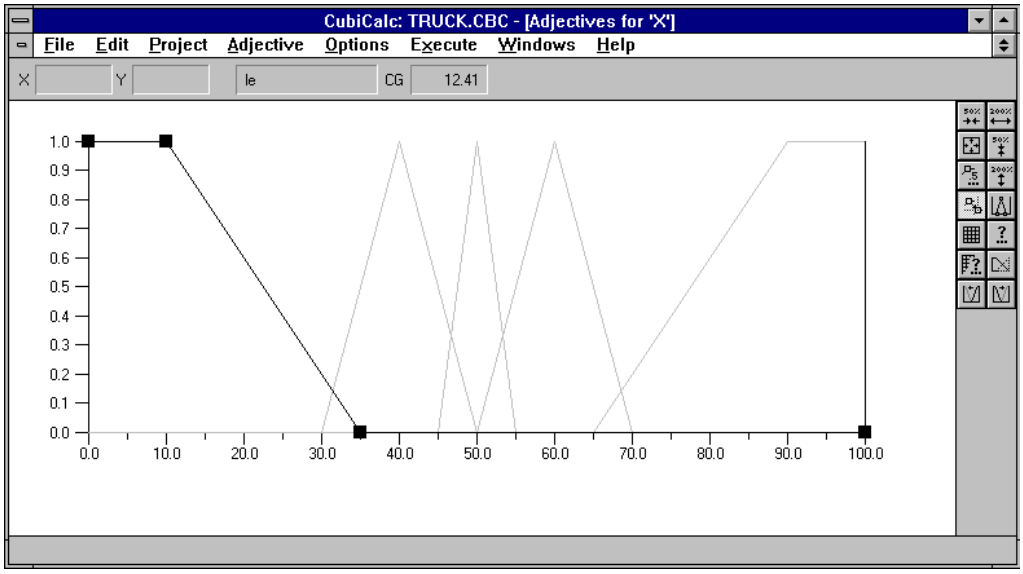


Рис. 6.37. Задання змінної «зміщення відносно воріт».

- для повороту керма (+/- 30 градусів) (рис. 6.38).

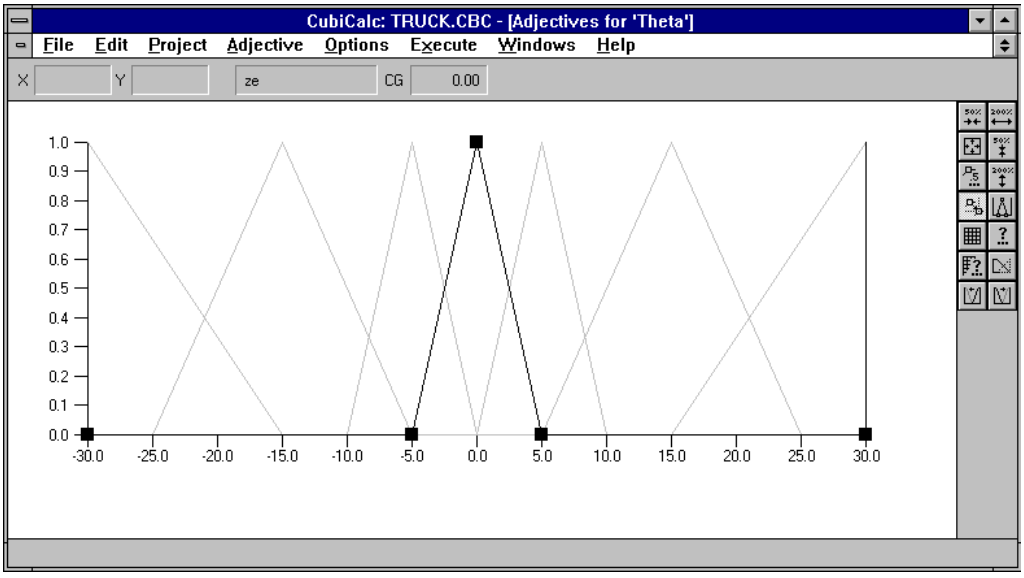


Рис. 6.38. Задання змінної «поворот керма».

Далі в меню PROJECT/VARIABLES задаємо: проміжні змінні (кожному нечіткому критерію «лівіше», «правіше» і тому подібне задається своя змінна); константи, що використовуються при розрахунках (в даному випадку – швидкість) (рис. 6.39).

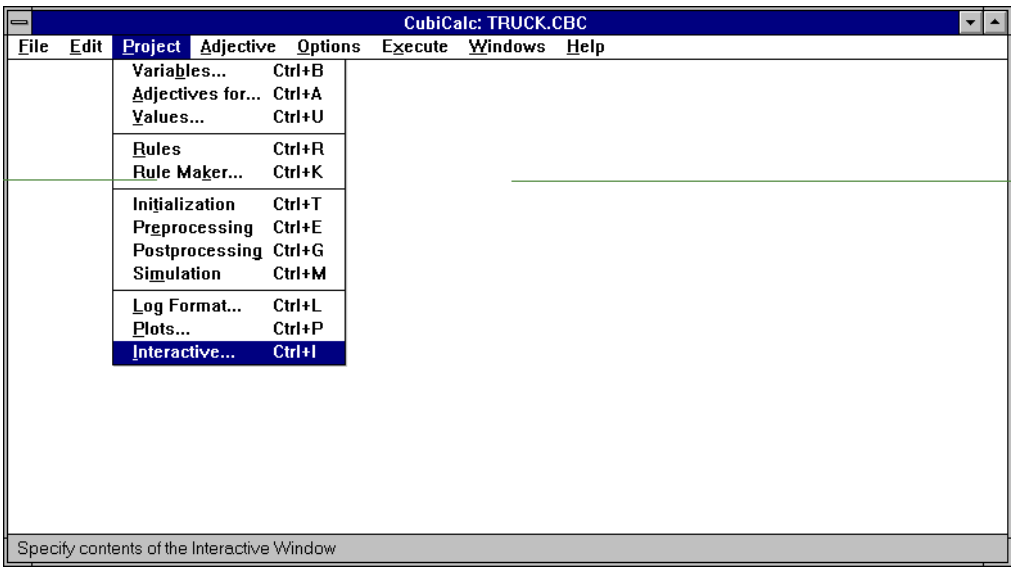


Рис. 6.39. Задання даних в меню PROJECT/VARIABLES.

Застосовуя меню PROJECT/INITIALIZATION, виконаємо ініціалізацію системи (параметри запуску), в даному випадку – випадкове розташування вантажівки у дворі, а за допомогою меню PROJECT/PREPROCESSING ../POSTPROCESSING, ../SIMULATION - відповідно, передобчислювання (до роботи Fuzzy - частини системи), постобчислення і проміжні обчислення. Меню PROJECT/LOG FORMAT ../PLOT, ../INTERACTIVE дозволяє задавати змінні, значення яких спостерігаються в процесі роботи, побудову значень вказаних змінних в графічному вигляді, управління значеннями змінних в процесі роботи.

Особливо слід сказати про меню PROJECT/RULES. Правила (у нашому сенсі - принципи) роботи системи виглядають таким чином (рис. 6.40).

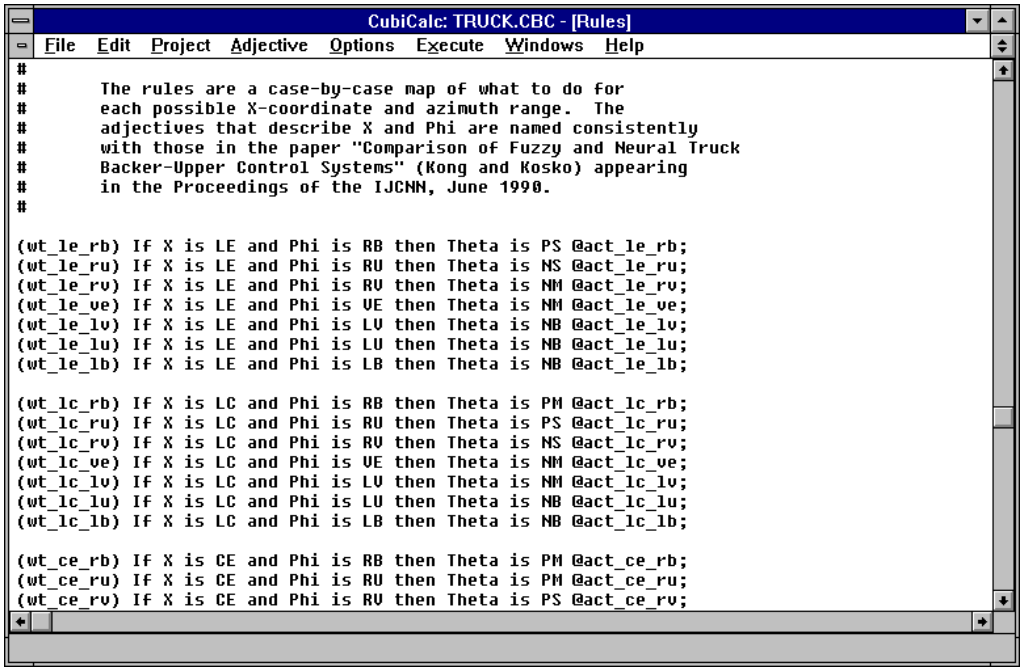


Рис. 6.40. Правила в меню PROJECT/RULES.

Як видно з малюнка 6.40, правила задаються у формі, інтуїтивно зрозумілій користувачеві: “**якщо** змінна X має значення Y (і/або змінна Z має значення F, і/або …) **то** змінна A набуває значення B, **інакше** змінна A набуває … і так далі. Наприклад, «**якщо** ніс вантажівки дивиться **вліво** і сама вантажівка знаходиться **лівіше** відносно воріт, то треба повернути кермо **глибоко вправо** і просунутися на наступний крок». Тобто виконується стандартна схема «якщо –

то – інакше». Для автоматичної побудови правил існує опція RULEMAKER. На останок в меню EXECUTE можна запустити отриману систему в безперервному або циклічному режимі і вивести додатковий екран для управління системою.

Отже, задача управління вантажівкою вирішена. Часу на її рішення було потрібно мало. Складність полягає в тому, аби правильно формалізувати потік даних і виділити в ньому змінні. Слід зазначити, що за допомогою CubiCalc успішно вирішуються задачі динамічного управління у фінансовому плануванні, технологічні процеси, моделювання складних наочних областей з параметрами, що якісно змінюються, порівняльно-оцінні задачі.

Для формалізації даних і виявлення правил широкого поширення набула програма Rule Maker для CubiCalc. Це є додаток до пакету CubiCalc. Її призначення – обробка масивів даних, виділення в масивах груп даних по деяких ознаках (кластеризація), виявлення зв'язків між виділеними групами (побудова правил). Оскільки границі кластерів задаються нечітко, можливі альтернативні висновки з вказівкою мірі їх достовірності, тобто операція за критеріями «краще-гірше”», «можливо» і так далі (рис. 6.41)

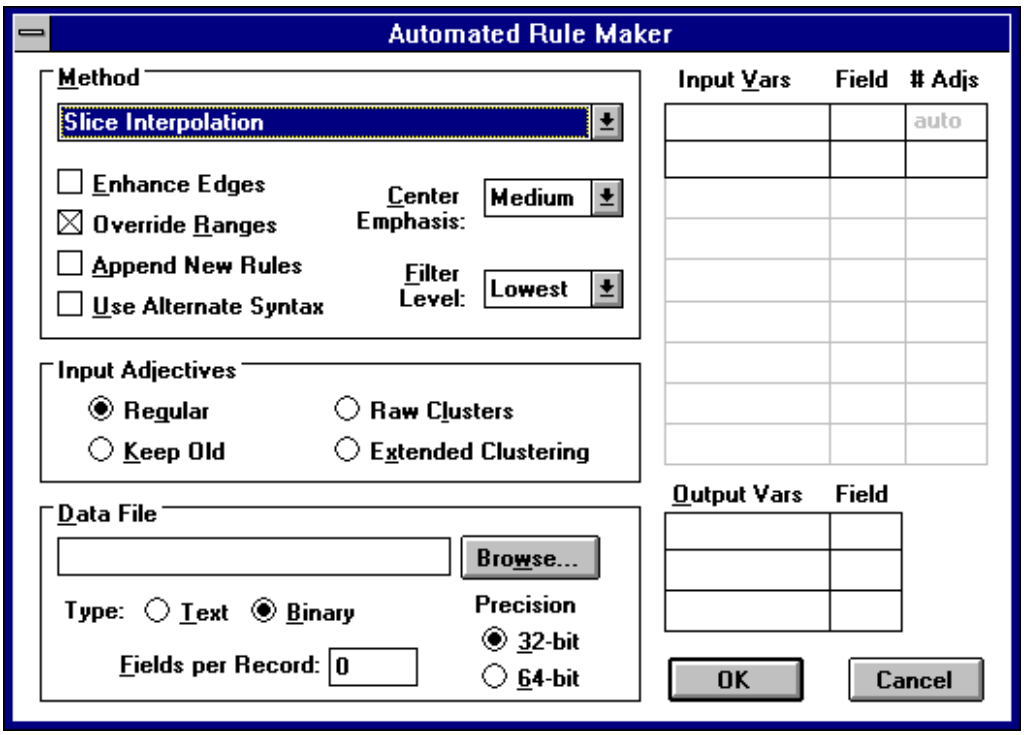


Рис. 6.41. Вигляд робочої області програми.

Задача знаходження правил вирішується в два прийоми:

1. *Кластеризація даних*. Термін «кластер» точно трактується як «точка концентрації». Тобто під кластеризацією в даному випадку розуміється знаходження областей (діапазонів значень), де співвідношення «вхід-вихід» зустрічається найчастіше. Ці діапазони описуються як fuzzy-змінні відповідно для входів і виходів. Оскільки границі кластерів описуються нечітко, це дозволяє відразу задавати деяку точність обчислень.

2. *Виявлення кореляційних залежностей між вхідними і вихідними значеннями*. Відмінність від класичного методу в тому, що кореляції шукаються і обчислюються не між змінами даних, а між змінами у межах знайдених кластерів. Результат виражається в наборі правил, що відображається в меню RULES. Тобто система знову ж таки оперує на рівні принципів поведінки об'єкту. На рисунку 6.42 показано, як працює механізм кусочно-лінійної інтерполяції функції при представленні її програмою Rule Maker. Також можливий механізм згладженої інтерполяції.

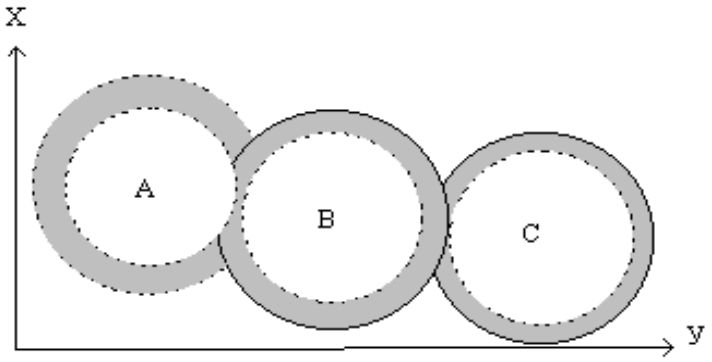


Рис. 6. 42. Механізм інтерполяції в Rule Maker.

На принципах нечіткої алгебри побудований ще один програмний продукт «Бізнес-прогноз». Призначення цього пакету - оцінка ризиків і потенційної прибутковості різних бізнес-планів, інвестиційних проектів і просто ідей по розвитку бізнесу. «Ведучи» користувача за сценарієм його задуму, програма задає низку запитань, що допускають як точні кількісні відповіді, так і наближені якісні оцінки, - типа «малоймовірно», «міра ризику

висока» і так далі. Узагальнивши всю отриману інформацію у вигляді єдиної схеми бізнес-проекту, програма не лише виносить остаточний вердикт про ризикованість проекту і очікуваних прибутках, але і вказує критичні точки і слабкі місця в авторському задумі. Від аналогічних іноземних пакетів «Бізнес-прогноз» відрізняється простотою і дешевизною.

Механізми нечітких обчислень представлені у вже відомому пакеті PolyAnalyst, зокрема, модуль Classify (CL) - класифікатор на основі нечіткої логіки. Алгоритм CL призначений для класифікації записів на два класи. У основі його роботи лежить побудова функції приналежності і знаходження порогу розділення на класи. Функція приналежності набуває значень від окресності 0 до окресності 1. Якщо повертаємо значення функції для даного запису більше порогу, то цей запис належить до класу «1», якщо менше, то до класу «0» відповідно. Цільова змінна для цього модуля має бути логічного типа (рис. 6.43).

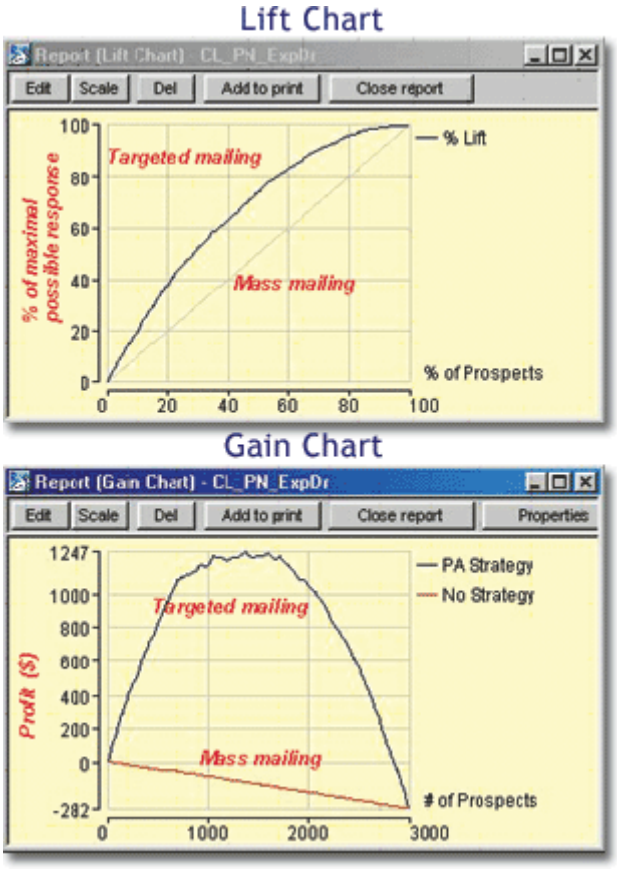


Рис. 6.43. Нечітка логіка в PolyAnalyst.

Алгоритм Discriminate (дискримінація) є модифікацією алгоритму CL. Він призначений для того, щоб з'ясувати, чим дані з вибраної таблиці відрізняються від останніх даних, включених в проект, іншими словами для виділення специфічних рис, що характеризують деяку підмножину записів проекту. На відміну від алгоритму CL, він не вимагає задання цільової змінної, досить вказати лише таблицю, для якої потрібно знайти відмінності.

Японський промисловий концерн Toshiba розробив новий програмний продукт Enfant (ENhanced Fuzzy And Neuro Tool), заснований на принципах нечіткої логіки. Структура базового блоку програми Enfant - т.з. Item (об'єкт, елемент) дозволяє будувати будь-які мережеві конструкції - семантичні і нейронні мережі, багатокаскадні управляючи комплекси і так далі. Річ у тому, що в структурі Item реалізована ідея універсального оброблювального елементу. Він здатний пропускати через себе два інформаційні потоки - прямий (Forward) і зворотний (Backward). При цьому вхідний потік, проходячи через Item, модифікується відповідно до поточного налаштування внутрішньої структури елементу, а зворотний потік, навпаки, може цю структуру міняти. Фахівці з нейронних мереж вже, ймовірно, взнали в подобі Item базову цеглинку для реалізації класичного нейромережевого алгоритму навчання back-propagation, а фахівці з теорії управління - універсальний контур зворотного зв'язку. А при чому тут нечітка логіка? Річ у тому, що до складу бібліотеки базових класів Enfant ENCL (Enfant Class Library) входять всі основні функції нечіткої логіки і нечіткої алгебри, а також можливості будь-якого комплексированія базових елементів. Пакет дозволяє будувати всілякі системи інтелектуального аналіза даних, управління, підтримки ухвалення рішень та ін. Ці системи можуть бути використані в освіті, дослідницьких роботах, моделюванні складних систем, створенні прототипів нових інтелектуальних пакетів.

Важливим елементом практичного впровадження теорії нечітких множин є можливість формування нечітких запитів до баз даних. Механізми нечітких запитів (fuzzy queries, flexible queries) до реляційних баз даних були

вперше запропоновані в 1984 році і згодом отримали розвиток в роботах Д. Дюбуа и Г. Прада.

Велика частина даних, що обробляються в сучасних інформаційних системах, носять чіткий, числовий характер. Проте в запитах до баз даних, які намагається формулювати людина, часто присутні неточності і невизначеності. Не дивно, коли на запит в пошуковій системі Інтернету користувачеві видається множина посилань на документи, впорядковані по мірі релевантності запиту. Тому що текстовій інформації спочатку властива нечіткість і невизначеність, причинами якої є семантична неоднозначність мови, наявність синонімів і так далі. З базами даних інформаційних систем, або з чіткими базами даних ситуація інша. Нечіткі запити неможливо виконати засобами мови SQL.

Продемонструємо обмеженість чітких запитів на наступному прикладі. Хай потрібно отримати відомості про менеджерів по продажах не старше 25 років, в яких сума річних операцій перевищила 200 тис. грн. по такому-то регіону. Даний запит можна записати на мові SQL таким чином:

Select FIO from Managers
where (Managers.Age <= 25 AND Managers.Sum > 200000 AND
Managers.RegionID = 1)

Менеджер по продажах 26 років з річною сумою продажів в 400 тис. грн., або 19 років з сумою в 198 тис. грн. не попадуть в результат запиту, хоча їх характеристики майже задовольняють вимогам запиту. Нечіткі запити допомагають впоратися з подібними проблемами «пропажі» інформації.

Механізм роботи нечітких запитів заснований на теорії нечіткої множини. Формалізуємо нечітке поняття «Вік співробітника компанії». Це буде назва відповідною лінгвістичною змінною. Задамо для неї область визначення X=[18, 70], три лінгвістичні терми – «Молодий», «Середній», «Вище середнього» і побудуємо функції приналежності для кожного лінгвістичного терма. Для цього виберемо трапецеїдальні функції приналежності з наступними

координатами: «Молодий»=[18, 18, 28, 34], «Середній»=[28, 35, 45, 50], «Вище середнього»=[42, 53, 60, 60] (рис. 6.44).

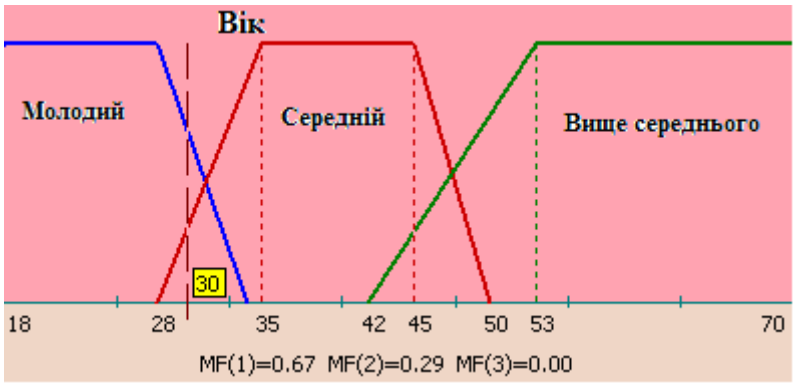


Рис. 6.44. Лінгвістична змінна «Вік».

Тепер можна обчислити міру приналежності співробітника 30 років до кожної з нечіткої множин: MF[Молодий](30)=0,67; MF[Середній](30)=0,29; MF[Вище середнього](30)=0.

Повернемося до прикладу з менеджерами по продажам. Для простоти передбачимо, що вся необхідна інформація знаходяться в одній таблиці з наступними полями: ID – номер співробітника, AGE – вік і SUM (річна сума операцій).

ID	AGE	SUM
1	23	120 500
2	25	164 000
3	28	398 000
4	31	489 700
5	33	251 900
…		

Визначимо ще одну лінгвістичну змінну для поля SUM з областю визначення X=[0, 600000], термами «Мала», «Середня» і «Велика» і побудуємо для них функції приналежності (рис. 6.45):
«Мала»=[0, 0, 0, 200000], «Середня»=[90000, 180000, 265000, 330000], «Велика»=[300000, 420000, 600000, 600000].

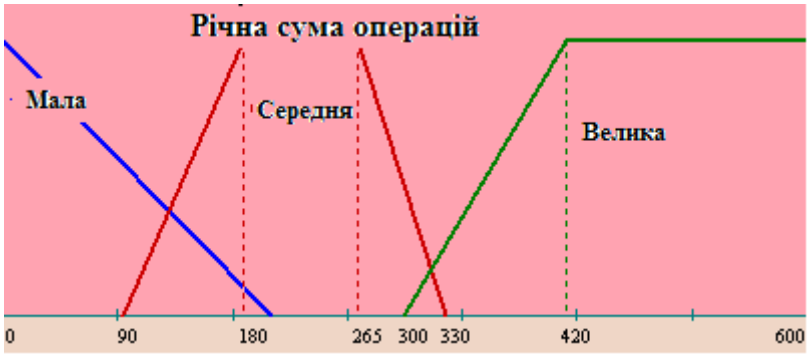


Рис. 6.45. Лінгвістична змінна «Річна сума операцій».

До такої таблиці можна робити нечіткі запити. Наприклад, отримати список всіх молодих менеджерів по продажах з великою річною сумою операцій, що на SQL запишеться так:

*Select * from Managers where (Age = «Молодий» AND Sum = «Велика»).*

Розрахувавши для кожного запису агреговане значення функції приналежності MF, отримаємо результат нечіткого запиту:

ID	AGE	SUM	MF
3	28	398 000	0,82
4	31	489 700	0,50

Записи 1, 2, 5 не попали в результат запиту, оскільки для них значення функції приналежності дорівнює нулю. Записів, що точно задовольняють поставленому запиту (MF=1), в таблиці не знайшлося. Менеджер по продажах 28 років і річною сумою 398000 відповідає запиту з функцією приналежності 0,82. На практиці зазвичай вводять порогове значення функції приналежності, при перевищенні якого записи включаються в результат нечіткого запиту.

Використовуючи нечіткі модифікатори, можна формувати і складніші запити:

*Select * from Managers where (Age = «<Більш-меньш Середній» AND Sum = «Середня»).*

Результат:

ID	AGE	SUM	MF
5	33	251 900	0,85

Часто потрібно оперувати не лінгвістичними змінними, а нечіткими аналогами точних значень. Для цього існує нечітке відношення «БІЛЯ» (наприклад, «Ціна близько 20»). Для реалізації подібних нечітких відношень аналогічно будується нечітка множина з відповідною функцією приналежності, але вже на деякому відносному інтервалі (наприклад [-5, 5]) для уникнення залежності від контексту. При обчисленні функції приналежності нечіткого відношення «Біля Q» (Q – деяке чітке число) виконується масштабування на відносний інтервал.

Проілюструємо вищесказане на прикладі таблиці з даними про коштовні папери. Хай вона має в своєму складі наступні поля: PRICE (вартість коштовного паперу), RATIO (відношення ціни до прибутку, price-to-earnings ratio), AYIELD (усереднений дохід за останній квартал, average yield, %).

ID	PRICE	RATIO	AYIELD
1	260	11	15,0
2	380	5	7,0
3	810	6	10,0
4	110	9	14,0
5	420	10	16,0

Хай потрібно знайти коштовні папери для покупки не дорожче $150, з прибутковістю 15% і відношенням ціни прибутку 11. Це еквівалентно наступному SQL-запиту:

*Select * from some table where ((PRICE <= 150) AND (RATIO = 11) AND (AYIELD = 15)).*

Результат такого запиту буде порожнім. Тоді сформулюємо цей же запит в нечіткому вигляді з використанням відношення «БІЛЯ»:

*Select * from some table where ((PRICE = «Близько 150») AND (RATIO = «Близько 11») AND (AYIELD = «Близько 15»)).*

Побудуємо нечітку множину для відношення «БІЛЯ» у відносному інтервалі [-5, 5]. Це буде трапеція з координатами [-2, -1, 1, 2] (рис. 6.46).

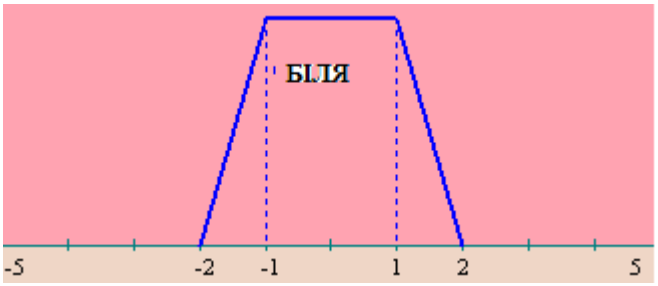


Рис. 6.46. Нечітка множина для відношення «Біля».

Розрахуємо значення нечіткого запиту «Ціна близько 250» для ціни 380. Заздалегідь задамо області визначення кожної лінгвістичної змінної: PRICE – [0, 1000], RATIO – [0, 20], AYIELD – [0, 20]. Значення 130 (отримане як різниця між 380 і 250) відмасштабуємо на інтервал [-5, 5], отримаємо величину x=1,3 і MF(1,3)=0,7. Застосувавши нечітке відношення «БІЛЯ» до кожного поля PRICE, RATIO і AYIELD і розрахувавши агреговане значення функції приналежності за допомогою операції нечітке «І», отримаємо наступний результат запиту:

ID	PRICE	RATIO	AYIELD	MF
1	260	11	15,0	1
4	110	9	14,0	0,9

Нечіткі запити перспективно використовувати в областях, де здійснюється вибір інформації з баз даних з використанням якісних критеріїв і нечітко сформульованих умов, наприклад, Direct Marketing. У прямому маркетингу дуже важливий етап виділення цільовій аудиторії, для якої застосовуватимуться різні інструменти Direct Marketing. Наприклад, це пряма поштова реклама (direct mail), що використовується при просуванні товарів і послуг організаціям і приватним особам. Проте для здобуття максимального ефекту від direct mail необхідний ретельний вибір адресатів. Якщо відбір адресатів буде ліберальним, то зростуть невиправдані витрати на прямий маркетинг, якщо дуже строгим – буде втрачений ряд потенційних клієнтів. Наприклад, компанія проводить рекламну акцію серед своїх клієнтів про нові послуги за допомогою прямої поштової розсилки. Служба маркетингу

встановила, що найцікавіший новий вигляд послуги буде чоловікам середніх років, батькам сімейств з річним доходом вище середнього. Для здобуття списку адресатів до бази даних клієнтів, швидше за все, буде зроблений наступний запит: «вибрати всіх осіб чоловічої статі у віці від 40 до 55 років, що мають мінімум 1 дитину, річний дохід від 20 до 30 тисяч дол.». Такі точні критерії запиту можуть відсіяти множину потенційних клієнтів: чоловік 39 років, батько трьох дітей з доходом в 31 тисяча не попаде в результат запиту, хоча це потенційний споживач нової послуги.

Аналогічним чином нечіткі запити можна використовувати при виборі туристичних послуг, підборі об'єктів нерухомості. Інструмент нечітких запитів дозволяє погоджувати формальні критерії і неформальні вимоги до круга потенційних клієнтів і задавати інтервали вибору потенційних клієнтів як нечітку множину. В такому разі клієнти, що не задовольняють якомусь одному критерію, можуть бути вибрані з бази даних, якщо вони мають хороші показники по інших критеріях.

Потужним засобом для практичного впровадження нечітких обчислень є *Fuzzy Logic Toolbox*, пакет прикладних програм, що входить до складу середовища MatLab. Він дозволяє створювати системи нечіткого логічного виводу і нечіткої класифікації в рамках середовища MatLab, з можливістю їх інтеграції в Simulink. Базовим поняттям Fuzzy Logic Toolbox є FIS-структура - система нечіткого виводу (Fuzzy Inference System). FIS-структура містить всі необхідні дані для реалізації функціонального відображення «входи-виходи» на основі нечіткого логічного виводу згідно певної схеми (рис. 6.47).

Fuzzy Logic Toolbox містить наступні категорії програмних інструментів:

- функції;
- інтерактивні модулі з графічним користувацьким інтерфейсом;
- блоки для пакету Simulink;
- демонстраційні приклади.

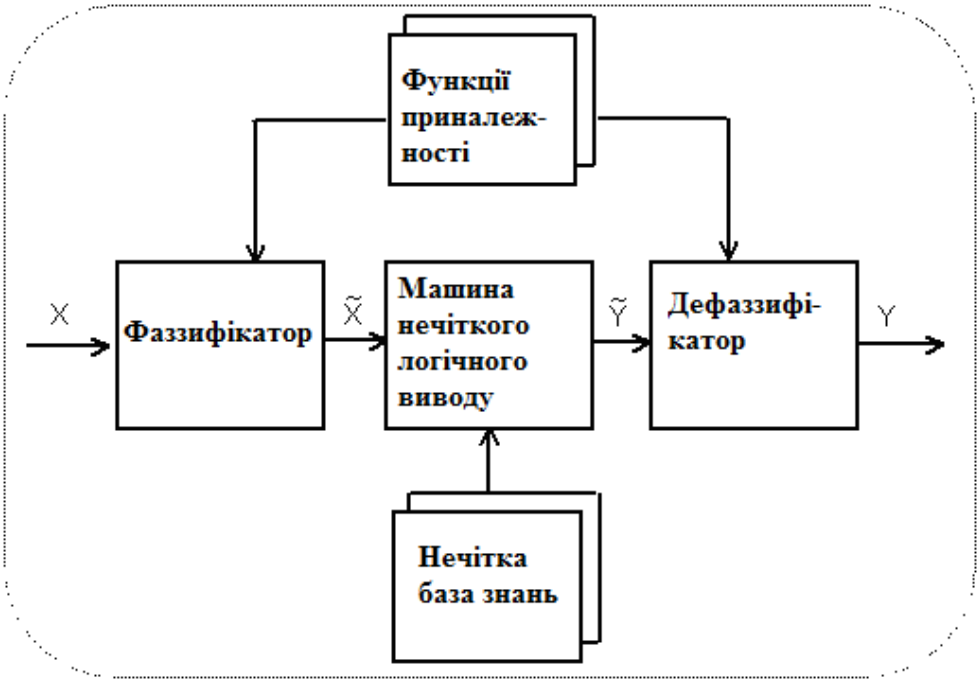


Рис. 6.47. Нечіткий логічний вивід.

Модуль Fuzzy дозволяє будує нечіткі системи двох типів - Мамдані і Сугено. У системах типа Мамдані база знань складається з правил вигляду «Якщо $x_1 =$ низький і $x_2 =$ середній, то $y =$ високий». У системах типа Сугено база знань складається з правил вигляду «Якщо $x_1 =$ низький і $x_2 =$ середній, то $y = a_0 + a_1 x_1 + a_2 x_2$». Таким чином, основна відмінність між системами Мамдані і Сугено полягає в різних способах задання значень вихідної змінної в правилах, які створюють базу знань. У системах типа Мамдані значення вихідної змінної задаються нечіткими термами, в системах типа Сугено - як лінійна комбінація вхідних змінних.

Проектування систем типу Мамдані виконується таким чином:

- задаємо функції приналежності змінної x_1 (рис. 6.48);
- задаємо функції приналежності змінної x_2 (аналогічно);
- задаємо функції приналежності змінної y;
- на основі візуального спостереження за графіком формулюємо правила і вводимо їх в базу знань (рис. 6.49);
- виконуємо візуалізацію нечіткого логічного виводу.

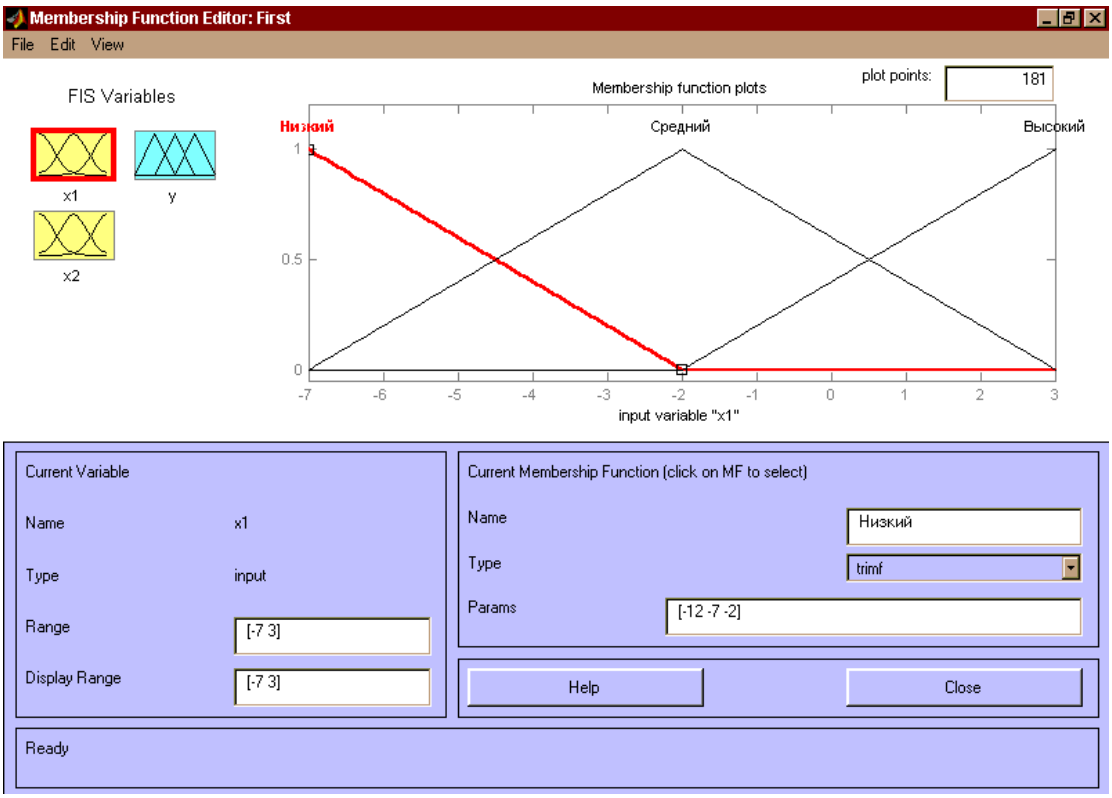


Рис. 6.48. Функції приналежності змінної x_1.

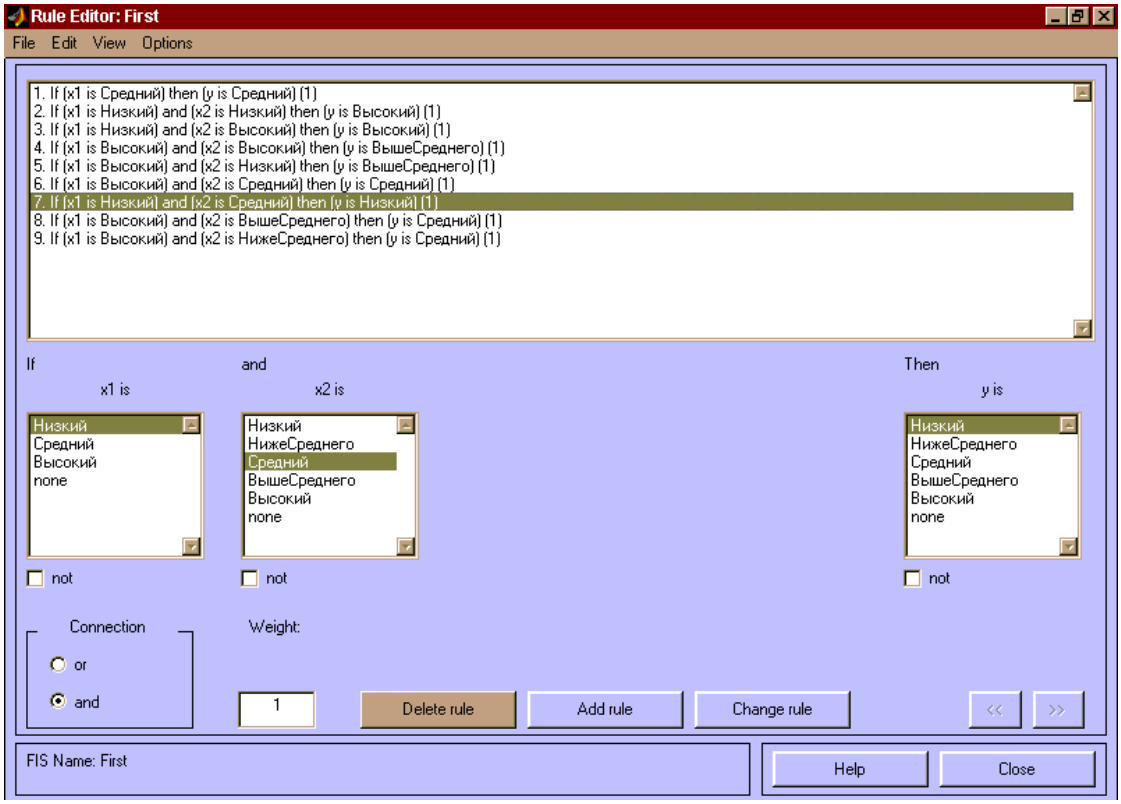


Рис.. 6.49. База знань в RuleEditor.

6.4. Сучасна практика застосування нечітких методів

Найбільш вражаючою властивістю людського інтелекту є здатність приймати правильні рішення в обстановці неповної і нечіткої інформації. Побудова моделей наближених до міркувань людини і використання їх в комп'ютерних системах майбутніх поколінь представляє сьогодні одну з найважливіших проблем науки. Зміщення центру досліджень нечітких систем у бік практичних застосувань привів до постановки цілого ряду проблем таких, як створення нечітких систем інтелектуального аналізу даних, нечіткого фінансового менеджменту, інструментальних засобів розробки, методів розрахунку і розробки нечітких систем управління і багато що інше. Розглянемо деякі з таких проблем [23, 27, 28, 77].

Прийняття рішень в нечітких умовах. У 1970 році Беллман і Заде опублікували статтю «Decision - Making in Fuzzy Environment», яка послужила відправною точкою для більшості робіт по нечіткій теорії прийняття рішень. У тій роботі розглядається процес прийняття рішень в умовах невизначеності, коли цілі і обмеження задані нечіткою множиною. Прийняття рішення - це вибір альтернативи, яка одночасно задовольняє і нечітким цілям, і нечітким обмеженням. У цьому сенсі, цілі і обмеження є симетричними відносно рішення, що стирає відмінності між ними і дозволяє представити рішення як злиття нечітких цілей і обмежень.

Хай $X = \{x\}$ - множина альтернатив. Нечітку мету $\tilde{G}$ ототожнюватиме з нечіткою множиною $\tilde{G}$ в X. Наприклад, якщо альтернативами є дійсні числа, тобто $X = R$, а нечітка мета сформульована як «x має бути близько 10», то її можна представити нечіткою множиною з такою функцією приналежності

$$\mu_G(x) = \frac{1}{1 + (x - 10)^2}$$

Аналогічним чином нечітке обмеження $\tilde{C}$ визначається як деяка нечітка множина на універсальній множині X. Наприклад, нечітке обмеження «x має

бути значно більше 8» при $X = R$ можна представити нечіткою множиною з такою функцією приналежності

$$\mu_C(x) = \begin{cases} 0, x < 5 \\ \dfrac{1}{1 + \exp(-0.8(x-8))}, x \geq 5 \end{cases}.$$

Нечітке рішення $\tilde{D}$ також визначається як нечітка множина на універсальній множині альтернатив X. Функція приналежності цієї нечіткої множини показує наскільки добре рішення задовольняє нечітким цілям «І» обмеженням. Логічній операції «І», яка зв'язує цілі і обмеження, відповідає операція пересічення нечітких множин. Отже, рішення - це пересічення нечіткої мети з нечітким обмеженням

$$\tilde{D} = \tilde{G} \cap \tilde{C}.$$

Приклад. Нечітка мета $\tilde{G}$ і нечітке обмеження $\tilde{C}$ сформульовані так:

$\tilde{G}$: «x має бути близько 10» та $\tilde{C}$: «x має бути значно більше 8».

Функції приналежності нечітких множин $\tilde{G}$ і $\tilde{C}$ задані вищеприведеними виразами. Необхідно знайти нечітке рішення $\tilde{D}$.

Нечітке рішення $\tilde{D}$ знайдемо як операцію пересічення вказаних нечітких множин. Враховуючи, що пересіченню нечітких множин відповідає операція мінімуму над функцією приналежності, отримуємо

$$\mu_D(x) = \begin{cases} 0, x < 5 \\ \min\left(\dfrac{1}{1 + \exp(-0.8(x-8))}, \dfrac{1}{1 + (x-10)^2} \right), x \geq 5 \end{cases}.$$

Взаємозв'язок між нечіткими метою, обмеженням і рішенням показана на рис. 6.50. Мета і обмеження конфліктують між собою, тому в нечіткій множині $\tilde{D}$ немає жодного елементу з мірою приналежності рівною 1. Значить, не існує альтернативи, яка повністю задовольняє і меті, і обмеженню. Як чітке рішення в таких випадках зазвичай вибирають альтернативу з максимальною мірою приналежності нечіткій множині $\tilde{D}$.

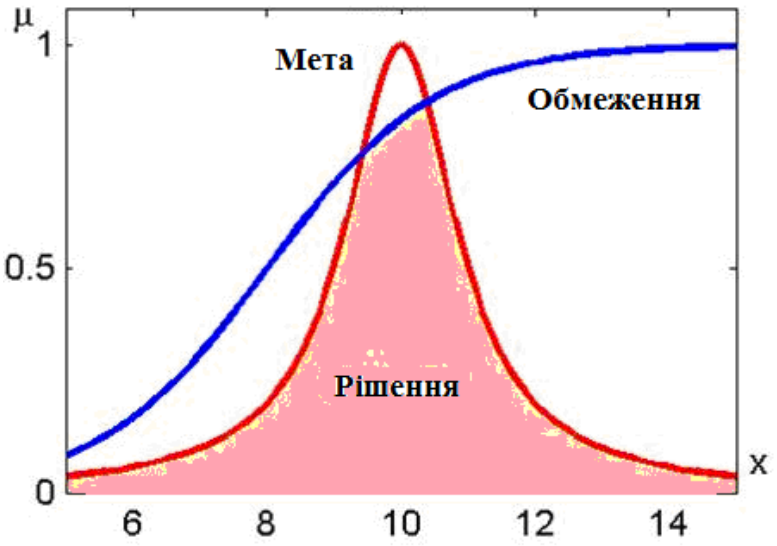


Рис. 6.50. Взаємозв’язок між нечіткими множинами.

При прийнятті рішень за схемою Беллман - Заде не робиться жодної відмінності між метою і обмеженнями. Всяке розділення на мету і обмеження є умовним. Можна поміняти місцями мету з обмеженням, при цьому рішення не зміниться. У традиційній теорії прийняття рішень подібні заміни функції переваги на обмеження недопустимі. Проте, і тут прослідкується деяка прихована схожість між цілями і обмеженнями. Вона стає явним при використанні методу невизначених множників Лагранжа і штрафних функцій, коли мета і обмеження об'єднуються в одну функцію. У загальному випадку, коли є n цілей і m обмежень, результуюче рішення за схемою Беллмана - Заде визначається пересіченням всіх цілей і обмежень $\tilde{D} = \tilde{G}_1 \cap ... \cap \tilde{G}_n \cap \tilde{C}_1 \cap ... \cap \tilde{C}_m$, і відповідно $\mu_D = \mu_{G_1} \cap ... \cap \mu_{G_n} \cap \mu_{C_1} \cap ... \cap \mu_{C_m}$.

До цих пір передбачалося, що всі цілі і обмеження, що входять в $\tilde{D}$, мають однакову важливість. Звичніше ситуація, в якій задоволення одним цілям і (або) обмеженням, важливіше чим іншим. Позначимо через $\alpha_i \in [0,1]$ - коефіцієнт відносної важливості і-ої цілі, а через $\beta_j \in [0,1]$ - коефіцієнт відносної важливості j-го обмеження $\sum_{i=1}^{n} \alpha_i + \sum_{j=1}^{m} \beta_j = 1$. Тоді функція приналежності рішення визначається так

$$\mu_D = (\mu_{G_1})^{\alpha_1} \wedge \ldots \wedge (\mu_{G_n})^{\alpha_n} \wedge (\mu_{C_1})^{\beta_1} \wedge \ldots \wedge (\mu_{C_m})^{\beta_m}.$$

Чим менше коефіцієнт відносної важливості, тим відповідна нечітка множина мети або обмеження стає більш розмазаною, і, отже, її роль в прийнятті рішення знижується. На рис. 6.51 приведені нечіткі рішення при різних коефіцієнтах важливості мети і обмеження.

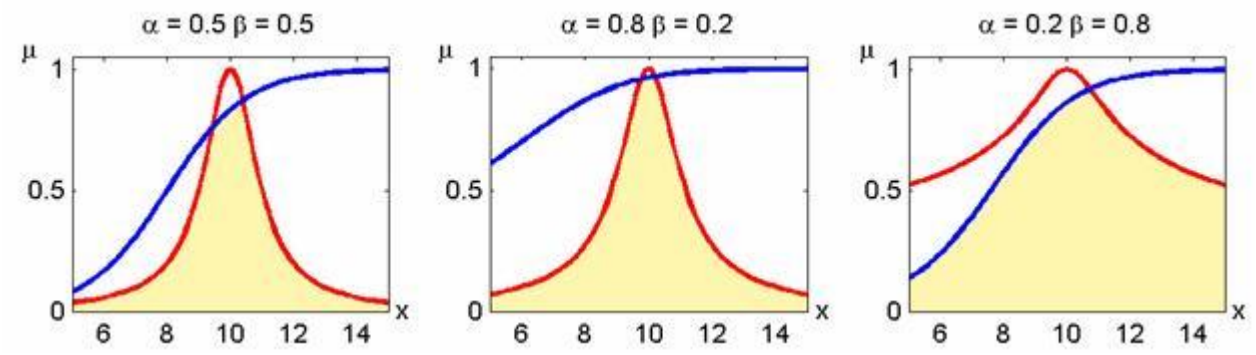


Рис. 6.51. Прийняття рішень при різній важливості мети і обмеження.

Важливою частиною нечіткого прийняття рішень є можливість проведення нечіткого багатокритеріального аналізу варіантів. Вважатимемо відомими

$X = \{x_1, x_2, ..., x_k\}$ - множина варіантів, які підлягають багатокритеріальному аналізу;

$G = \{G_1, G_2, ..., G_n\}$ - множина кількісних і якісних критеріїв, якими оцінюються варіанти.

Задача багатокритерійного аналізу полягає у впорядковуванні елементів множини X по критеріях з множини G.

Будемо вважати, що $\mu_{G_i}(x_j)$ - число в діапазоні [0,1], яке характеризує рівень оцінки варіанту $x_j \in X$ по критерію $G_i \in G$: чим більше число $\mu_{G_i}(x_j)$, тим вище оцінка варіанту x_j по критерію G_i. Тоді критерій G_i можна представити у вигляді нечіткої множини $\tilde{G}_i$ на універсальній множині варіантів X

$$\tilde{G}_i = \left\{ \frac{\mu_{G_i}(x_1)}{x_1}, ..., \frac{\mu_{G_i}(x_k)}{x_k} \right\}.$$

Знаходити ступені приналежності вищеприведеної нечіткої множини зручно *методом побудови функцій приналежності на основі парних порівнянь*. При використанні цього методу необхідно сформувати матриці парних порівнянь варіантів по кожному критерію. Загальна кількість таких матриць збігається з кількістю критеріїв і дорівнює n. Найкращим варіантом будемо той, який одночасно кращий по всіх критеріях. Нечітке рішення $\tilde{D}$ знаходиться як пересічення приватних критеріїв

$$\tilde{D} = \tilde{G}_1 \cap ... \cap \tilde{G}_n = \left\{ \frac{\min \mu_{G_i}(x_1)}{x_1}, ..., \frac{\min \mu_{G_i}(x_k)}{x_k} \right\}$$

Згідно з отриманою нечіткою множиною $\tilde{D}$, найкращим варіантом слід вважати той, для якого ступінь приналежності є найбільшою.

Розглянемо застосування нечіткого багатокритеріального аналізу на прикладі аналізу інноваційних проектів, зокрема, порівнянні техніко-економічного рівня трьох проектів (x_1, x_2, x_3), спрямованих до інноваційного фонду з метою отримання фінансування. Для оцінки техніко-економічного рівня проектів скористаємося такими критеріями:

G_1 - масштаб проекту;

G_2 - новизна проекту;

G_3 - пріоритетність напряму;

G_4 - ступінь опрацьовування;

G_5 - правова захищеність;

G_6 - екологічний рівень.

При експертному порівнянні проектів x_1, x_2, x_3 по критеріях $G_1, G_2, ..., G_6$ були отримані лінгвістичні вислови, показані в таблиці 6.2.

Таблиця 6.2.

Парні порівняння проектів по шкалі Сааті.

Критерій	Парні порівняння
G_1	*Відсутність* переваги x_1 над x_2 *Істотна* перевага x_3 над x_1
G_2	*Майже істотна* перевага x_1 над x_3 *Слабка* перевага x_2 над x_3
G_3	*Істотна* перевага x_1 над x_2 *Явна* перевага x_1 над x_3
G_4	*Слабка* перевага x_2 над x_1 *Майже слабка* перевага x_3 над x_1
G_5	*Істотна* перевага x_1 над x_2 *Майже явна* перевага x_1 над x_3
G_6	*Майже істотна* перевага x_1 над x_2 *Майже слабка* перевага x_3 над x_1

Цим експертним висловам відповідають наступні матриці парних порівнянь:

$$A(G_1)=\begin{vmatrix}1 & 1 & 0.2\\ 1 & 1 & 0.2\\ 5 & 5 & 1\end{vmatrix};\quad A(G_2)=\begin{vmatrix}1 & 1.35 & 4\\ 0.75 & 1 & 3\\ 0.25 & 0.33 & 1\end{vmatrix};$$

$$A(G_3)=\begin{vmatrix}1 & 5 & 7\\ 0.2 & 1 & 1.4\\ 0.14 & 0.71 & 1\end{vmatrix};\quad A(G_4)=\begin{vmatrix}1 & 0.33 & 0.5\\ 3 & 1 & 1.5\\ 2 & 0.67 & 1\end{vmatrix};$$

$$A(G_5)=\begin{vmatrix}1 & 5 & 6\\ 0.2 & 1 & 1.2\\ 0.17 & 0.83 & 1\end{vmatrix};\quad A(G_6)=\begin{vmatrix}1 & 4 & 0.5\\ 0.25 & 1 & 0.13\\ 2 & 8 & 1\end{vmatrix}.$$

Застосовуючи отриману формулу до матрицями парних порівнянь, отримуємо наступну нечітку множину

$$\tilde{G}_1 = \left\{ \frac{0.14}{x_1}, \frac{0.14}{x_2}, \frac{0.72}{x_3} \right\}; \quad \tilde{G}_2 = \left\{ \frac{0.5}{x_1}, \frac{0.38}{x_2}, \frac{0.12}{x_3} \right\};$$

$$\tilde{G}_3 = \left\{ \frac{0.74}{x_1}, \frac{0.15}{x_2}, \frac{0.11}{x_3} \right\}; \quad \tilde{G}_4 = \left\{ \frac{0.17}{x_1}, \frac{0.5}{x_2}, \frac{0.33}{x_3} \right\};$$

$$\tilde{G}_5 = \left\{ \frac{0.73}{x_1}, \frac{0.15}{x_2}, \frac{0.12}{x_3} \right\}; \quad \tilde{G}_6 = \left\{ \frac{0.31}{x_1}, \frac{0.08}{x_2}, \frac{0.61}{x_3} \right\}.$$

Згідно представленій теорії результат визначимо таким чином

$$\tilde{D} = \left\{ \frac{0.14}{x_1}, \frac{0.08}{x_2}, \frac{0.11}{x_3} \right\},$$

що свідчить про істотну перевагу проекту x_1 над проектом x_2, а також про слабку перевагу проекту x_1 над проектом x_3.

Концепція оцінки кредитоспроможності фізичних осіб. В даний час проблема своєчасного повернення кредитів, виданих фізичним особам, актуальна для більшості банківських установ. Її рішення значною мірою залежить від «якості» оцінки кредитоспроможності потенційних позичальників. У банківській сфері під кредитоспроможністю розуміється можливість і, що важливо, бажання контрагента виконувати зобов'язання по погашенню заборгованості. Найбільш достовірно це визначення відображає кредитна історія, тобто інформація про те, які кредити і в яких банках отримував позичальник, і як розплачувався по ним. У системі кредитування великої кількості банків, оцінка кредитної історії виконується експертом, який в основному спирається на свій досвід і інтуїцію, що може приводити до внесення в рішення суб'єктивних міркувань, що не мають достатніх підстав.

Зниження можливості впливу експерта на рішення і підвищення в ньому долі об'єктивних чинників може бути забезпечене формалізацією прогнозу поведінки позичальника і процедури ухвалення рішення про видачу кредиту. У зв'язку з цим актуальним є задача розробки уніфікованого підходу до оцінки кредитоспроможності фізичних осіб. Основою для впровадження подібної системи є кредитні історії, експертна оцінка яких є в значній мірі вербальним

описом і внаслідок цього – нечітке. Одним із способів формалізації вербальних величин і перетворення їх в кількісні є застосування теорія нечітких множин Заде.

Розробка математичної моделі для аналізу якості виконання зобов'язань позичальника вимагає наявності адекватного формального представлення, яке б враховувало особливості кредитування фізичних осіб. Стосовно процесу оцінки кредитної історії лінгвістична змінна може бути задана у вигляді наступного набору: $\langle X, T, U, G, M \rangle$, де

X - лінгвістична змінна з ім'ям «кредитна історія».

T - множина значень лінгвістичною змінною X, областю визначення кожної з яких є множина U. У банківській практиці кредитну історію найчастіше класифікують по наступних категоріях: «позитивна», «прийнятна», «негативна».

U - набір кількісних характеристик, на підставі яких можливо визначити приналежність кредитної історії до значень, що входять в T. Наприклад, він може мати вигляд: $U = \{$«кількість прострочених платежів», «кількість днів в перебігу яких погашення не виконувалося» і т. д.$\}$.

G - синтаксичні правила, що породжують назву нових термів. Елементи множини G призначені для формування значень X, що деталізують кредитні історії. На основі комбінацій елементів $t \in T$ та $g \in G$, можуть бути введені додаткові значення множини T. Наприклад, при $G = \{$«не», «дуже», «більш-менш»$\}$, кредитній історії можуть бути надані наступні лінгвістичні значення: {«не негативна», «більш-менш прийнятна», «не позитивна»}.

M - семантичні правила, що задають функції приналежності нечітких термів, породжених синтаксичними правилами G.

Для вирішення задачі по визначенню рівня кредитоспроможності фізичних осіб розроблена концепція класифікації кредитних історій (рис. 6.52).

Вхідна інформація є вибіркою по кредитних історіях, яка використовується для навчання моделі. Відповідно до принципів побудови

скорингових систем зазвичай розглядаються кредити позичальників, що розплатилися по своїх зобов'язаннях.

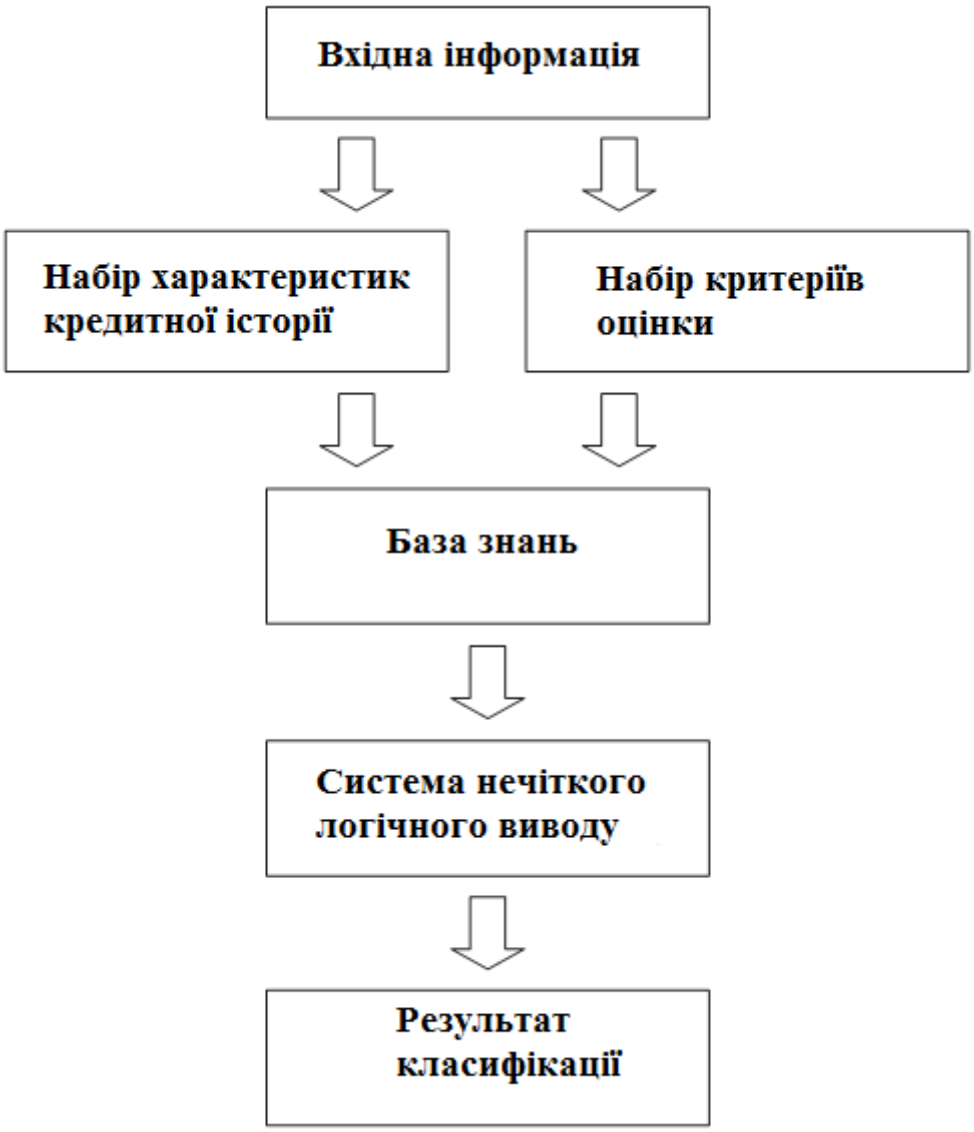


Рис. 6.52. Концепція класифікації кредитних історій.

Основним положенням концепції є побудова нечіткої бази знань, яка є відображенням знань і досвіду експерта у вигляді загального зведення правил, погодженого з політикою фінансово - кредитної організації по відношенню до кредитних ризиків, у вигляді умовних висловів: *Якщо* «Набор умов», *То* «Вивід». Наведемо приклад бази знань, орієнтованої на класифікацію «негативних» кредитних історій.

1. ЯКЩО $x_1 =$«багато», ТОДІ $y =$«висока».

2. ЯКЩО $x_1 =$«невелика кількість» І $x_2 =$«невеликий», ТОДІ $y =$«більш-менш низька».

3. ЯКЩО $x_1 =$«невелика кількість» І $x_2 =$«зовсім невеликий», ТОДІ $y =$«низька».

4. ЯКЩО $x_1 =$«багато» І $x_2 =$«невеликий», ТОДІ $y =$«більш-менш висока».

В правилах застосовані позначення: x_1 - загальна кількість пропущених періодів оплати, x_2 - максимальний термін недоплати в днях, y – можливість оцінки кредитної історії, як «негативної».

Математична інтерпретація дій експерта полягає в реалізації механізму, що дозволяє з певною мірою розділити кредитні історії, на категорії, найчастіше використовувані в банківській практиці. Рішення поставленої задачі можливе з використанням алгоритму нечіткого логічного виведення Мамдані, орієнтованого на роботу з гуманістичними системами, побудованими на підставі думок і припущень про змістовну сутність області дослідження.

Надалі, маючи в наявності інформацію про «хороші» і «погані» кредити і відомості, по відповідним їм клієнтам (анкетні дані, довідки про доходи, наявність у власності майна і т. д.) можлива побудова функціональної відповідності, що представляє залежність якості кредитної історії від характеристик позичальника. Таким чином, на етапі розгляду кредитної заявки, нечітка модель дозволить зробити висновок про найбільш імовірний рівень платіжної дисципліни потенційного клієнта.

Оцінка інвестиційних проектів. Як і в більшості задач економіки, при розрахунку показників інвестиційного проекту економісти стикаються з труднощами у вигляді неповноти, нечіткості або невизначеності вхідних даних. Для вирішення цих проблем існує досить багато способів, наприклад – вживання статистичних, або мінімаксних методів, проте, на практиці вони не завжди виявляються ефективними, а інколи навіть непридатними. Найбільш пристосованим до даних проблем виявився апарат, заснований на теорії нечітких множин.

Прикладом, що добре розкриває переваги «нечіткого підходу», є задача оцінки таких показників інвестиційних проектів, як чиста поточна вартість доходу (NPV) і внутрішня ставка прибутковості (IRR). У них поєднуються нечіткість і невизначеність даних, і неясність, розпливчатість оточення проекту. Розглянемо приклади, які показують потенційну застосовність «нечіткого підходу» як універсального апарату при оцінці ефективності інвестицій.

1. *Оцінка значення чистої поточної вартості доходу (NPV) інвестиційного проекту*. Як відомо, одним з основних показників ефективності інвестиційного проекту є показник NPV (Net Present Value) – чиста поточна вартість доходу.

$$NPV = -I + \sum_{i=0}^{N} \frac{\Delta V_i}{(1+r)^i}, \text{ де}$$

I - об'єм первинних інвестицій;

ΔV_i - оборотне сальдо поступлень і платежем в i періоді;

N - число періодів;

r_i - ставка дисконтування в i періоді.

Визначимо змінні, які можуть бути представлені в нечіткій формі. Величина I – можна лише приблизно знати, який об'єм первинних інвестицій матимемо в своєму розпорядженні на початок інвестиційного проекту. Величина ΔV_i – сальдо в конкретний рік також можна знати лише неточно. Величина r_i – досить важко точно визначити ставку дисконтування.

Після задання в нечіткій формі всіх вхідних даних проекту необхідно визначити кількість рівнів приналежності. При малій кількості рівнів точність результату може виявитися низькою, а при великому їх числі різко зростає обчислювальна складність розрахунків і трудомісткість обчислень. Для досягнення достатньої точності необхідно задати 4-5 рівнів, проте, ця рекомендація не є обов'язковою і залежить від умов задачі. Розрахуємо значення NPV для кожного з рівнів приналежності.

$$[NPV_1, NPV_2] = [-I_1 + \sum_{i=1}^{N} \frac{\Delta V_{i2}}{(1+r_1)^r}, -I_2 + \sum_{i=1}^{N} \frac{\Delta V_{i1}}{(1+r_2)^r}].$$

Потім, розраховані таким чином інтервали апроксимуємо по їх крайніх точках, отримаємо результат у вигляді нечіткої множини.

Проілюструємо дану методику на наступному прикладі. На початку інвестиційного проекту необхідно отримати близько 3 млн. грн. Передбачається, що термін реалізації проекту складе 2 роки, в кожен з яких він в середньому приноситиме 2 млн. грн. Ставка дисконтування передбачається на рівні приблизно 25%.

Як видно з умови, в задачі є нечіткі параметри, ознакою яких є слова «біля», «в середньому», «приблизно» і так далі. Представимо їх в нечіткій формі, і виразимо графічно.

- $I = (2.9;\ 3;\ 3.1)$ – передбачається, що швидше за все на початок проекту отримаємо 3 млн. грн., але залежно від умов, можемо отримати від 2.9 до 3.1 млн. грн (мал. 6.53).

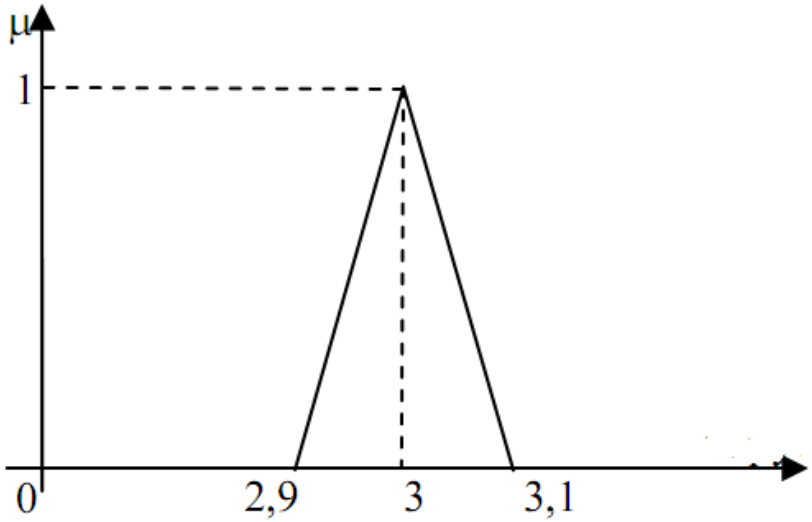


Рис. 6.53. Нечітке число «Розмір інвестицій».

- $\Delta V_1 = \Delta V_2 = (1.3;\ 2;\ 2.7)$ – передбачається, що, швидше за все прибуток в кожен рік складе близько 2 млн. грн., проте він може коливатися в межах від 1.3 до 2.7 млн. грн (рис. 6.54).

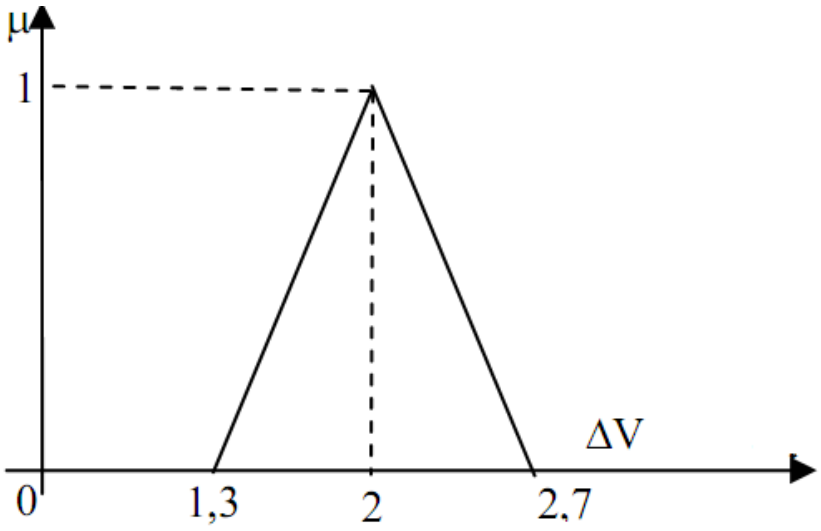


Рис. 6.54. Нечітка множина «Зворотне сальдо».

- $r_1 = r_2 = (0.2;\ 0.25;\ 0.35)$ – ставка дисконтування може коливатися від 20% до 35%, але, швидше за все, вона складе 25% (рис. 6.55).

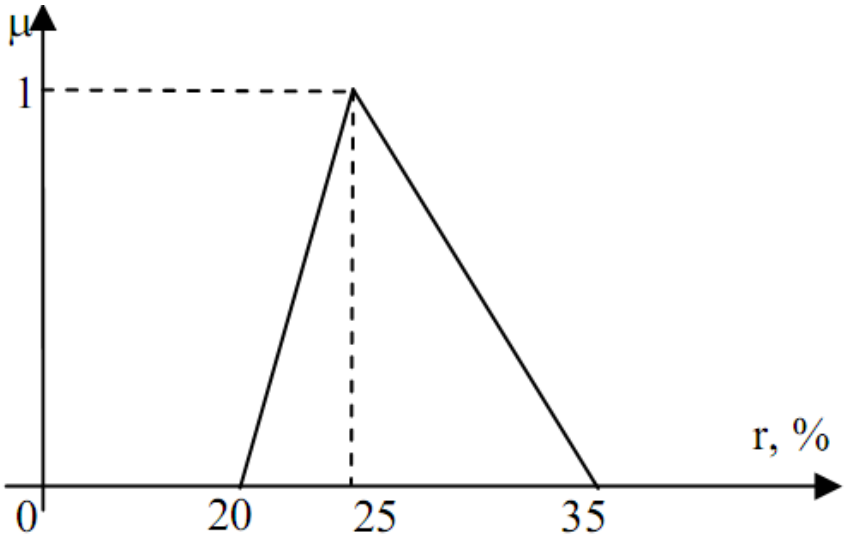


Рис. 6. 55. Нечітке число «Ставка дисконтування».

Для здобуття результату, що задовольняє по точності, досить задатися 5-ма рівнями приналежності. Розрахуємо для кожного з рівнів інтервал значень NPV. Результати обчислень занесемо в таблицю (табл. 6.3) і виразимо результат графічно (мал. 6.56).

Табл. 6.3.

Результати розрахунку NPV по α - рівням

α - рівень	I	ΔV	r	NPV
0	[2.9, 3.1]	[1.3, 2.7]	[0.2, 0.35]	[1.225, -1.424]
0.25	[2.925, 3.075]	[1.475, 2.525]	[0.2125, 0.325]	[0.875, -1.112]
0.5	[2.95, 3.05]	[1.65, 2.35]	[0.225, 0.3]	[0.534, -0.804]
0.75	[2.975, 3.025	[1.825, 2.175]	[0.2375, 0.275]	[0.203, -0.471]
1	[3, 3]	[2, 2]	[0.25, 0.25]	[-0.12, -0.12]

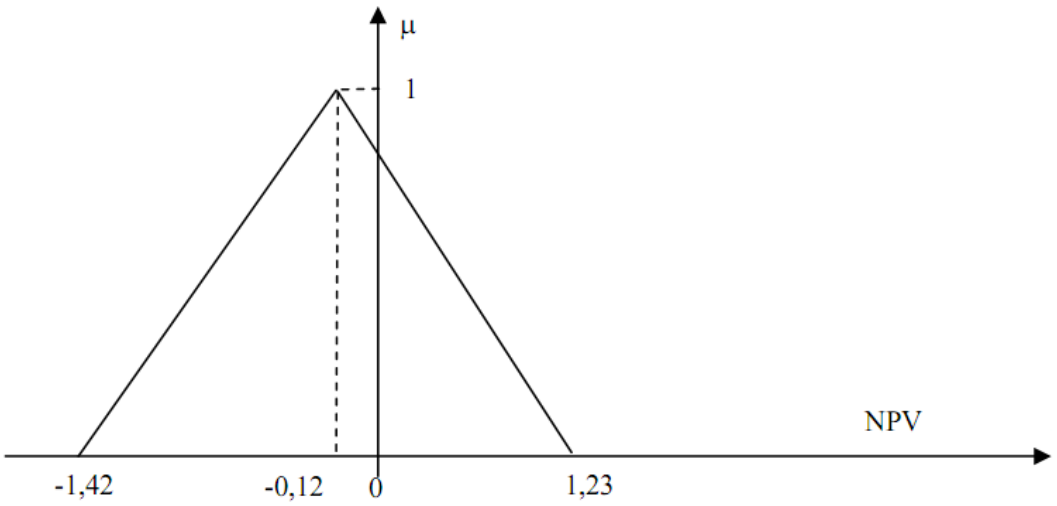


Рис. 6.56. Розраховане значення NPV.

Як видно з графіка, найбільш імовірне значення NPV знаходитиметься біля -0,12 млн. грн. Таким чином, швидше за все проект не принесе прибутку і буде збитковим. Проте, за деяких умов можливе набуття позитивного значення NPV. Ризик його негативних значень, можна визначити як співвідношення площі, яка потрапляє в негативну область і загальній площі отриманої фігури. А найбільш імовірне значення визначатиметься як аргумент точки, , що є центром тяжесті даної фігури.

2. *Оцінка внутрішньої ставки прибутковості (IRR - internal rate of return) інвестиційного проекту.* Іншим важливим показником ефективності інвестиційного проекту є показник внутрішньої ставки прибутковості IRR.

Даний показник дозволяє оцінити запас міцності проекту. Сенс його полягає в знаходженні значення ставки дисконтування r, яка обертає NPV в нуль, і зводиться до вирішення степенного рівняння

$$NPV = -I + \sum_{i=1}^{N} \frac{\Delta V_i}{(1+r)^i} = 0.$$

Обґрунтування необхідності представлення даних в нечіткій формі представлене в попередньому прикладі. Вирішення ж рівнянь в нечіткому вигляді принципово не відрізняється від обчислення нечіткого вираження і зводиться до:

- визначенню кількості α - рівнів;
- рішенню на кожному рівні інтервальних степенних рівнянь

$$[-I_1 + \sum_{i=1}^{N} \frac{\Delta V_{i2}}{(1+r_1)^r} = 0; -I_2 + \sum_{i=1}^{N} \frac{\Delta V_{i1}}{(1+r_2)^r} = 0]$$

- апроксимації отриманого нечіткого значення.

Розглянемо приклад розрахунку IRR в нечіткій формі. Вихідні дані візьмемо з попередньої задачі: $I = (2.9;\ 3;\ 3.1)$, $\Delta V_1 = \Delta V_2 = (1.3;\ 2;\ 2.7)$, $\alpha = 5$. Вирішуючи, послідовно для кожного рівня рівняння, отримаємо результат у вигляді таблиці (табл. 6.4).

Табл. 6.4.

Результати розрахунку IRR по α - рівням

α - рівень	I	ΔV	r
0	[2.9, 3.1]	[1.3, 2.7]	[-10.97%, 53.68%]
0.25	[2.925, 3.075]	[1.475, 2.525]	[-2.72%, 45.61%]
0.5	[2.95, 3.05]	[1.65, 2.35]	[5.42%, 37.57%]
0.75	[2.975, 3.025	[1.825, 2.175]	[13.49%, 29.54%]
1	[3, 3]	[2, 2]	[21.53%, 21.53%]

В результаті обчислень і апроксимації отримуємо результат, що виражається нечітким числом: $r = (-10{,}97\%;\ 21{,}53\%;\ 53{,}68\%)$, і представимо його графічно (рис. 6.57).

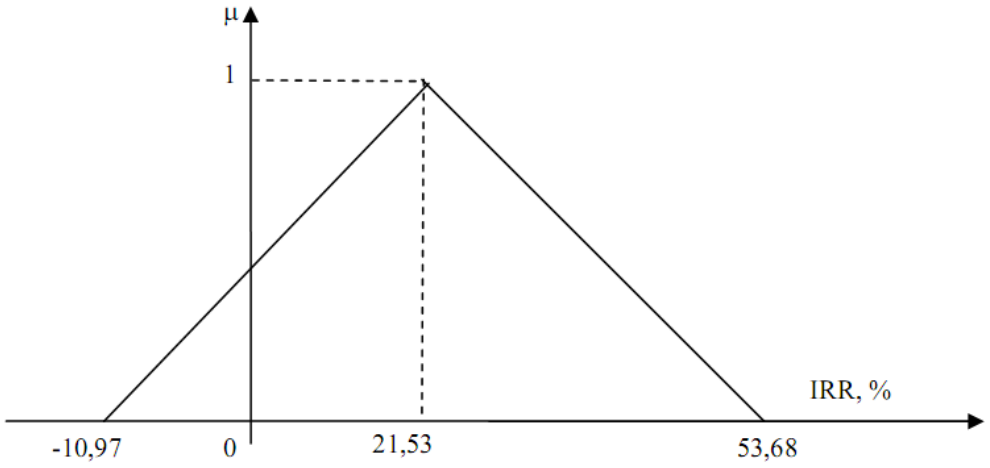


Рис. 6.57. Результат розрахунку IRR в нечіткій формі.

Таким чином, на прикладах показано, що розрахунки в нечіткій формі дають додаткові можливості для аналізу інвестиційних проектів. Це може бути корисно, тим більше що даний метод враховує все нечіткості і невизначеності у вихідних даних, що не дозволяє зробити жодна із загальноприйнятих методик.

Застосування нечітких множин в бізнесі. В даний час починають з'являтися інформаційні ресурси, в рамках яких агрегується інформація про діяльність сотень українських корпорацій. Весь цей накопичений об'єм даних серйозно ще ніким не досліджувався, - та і не було уявлення про те, як все це різноманіття кількісної та якісної інформації можна аналізувати в одному ключі. Сьогодні такий підхід до аналізу економічних даних нарешті сформувався - підхід Fuzzy Economics. Справа лише за тим, аби на основі розробленої методології створити системі інтелектуального аналізу, що виконують аналітичну обробку даних і вироблення оптимальних економічних рішень.

Однією з таких економічних проблем, яку дозволяють ефективно розв’язувати нечіткі методі є кваліметрія на базі агрегації ієрархій чинників. Нехай деяка властивість економічного об'єкту (фінансова стійкість підприємства, інвестиційна привабливість коштовного паперу, рівень менеджменту управляючої компанії, ринкова привабливість території під

забудову і так далі) може бути представлено як деревовидна ієрархія чинників (рис. 6.58), причому:

- в рамках ієрархії визначені системи відношення переваги одних підвластивостей іншим для одного рівня ієрархії;
- підвластивості, що становлять низові ланки ієрархії, можуть бути виміряні як кількісно, так і якісно (у тому числі словесно).

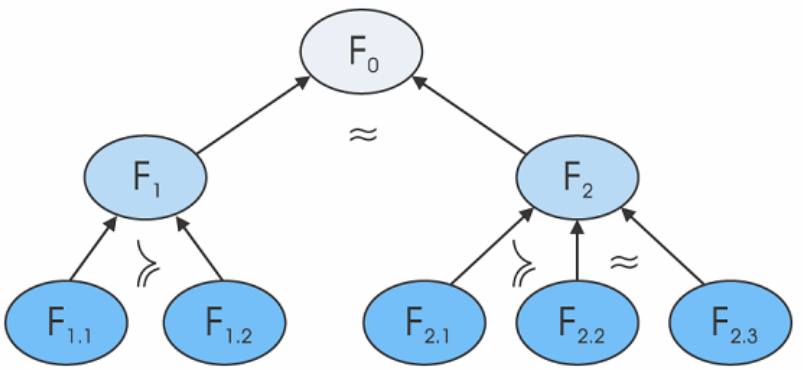


Рис. 6.58. Деревовидна ієрархія властивостей.

В цьому випадку, можна здійснити комплексну оцінку сили базової властивості, якщо:

- виробляти всі виміри на якісному базисі, виконуючи нечітку класифікацію кількісних чинників;
- для моделювання систем переваги застосовувати системи вагів Сааті або Фішберна;
- виробляти комплексування якісних рівнів чинників в рамках двовимірної згортки, де однією з систем вагів виступають ваги чинників, а іншою системою вагів – вузлові точки класифікатора.

Коли рівень сили комплексної властивості, що нормується на деякому стандартному носієві (наприклад, 01-інтервал), отриманий, можна виконати розпізнавання якісного рівня даної комплексної властивості на основі відповідного нечіткого класифікатора. Також, на підставі отриманої оцінки якості, можна виконувати зв'язаний аналіз якості об'єкту і деяких додаткових його властивостей (наприклад, ціни), аби виділяти такі об'єкти, для яких

досягається наприклад, максимум якості при фіксованій ціні, або, навпаки, мінімум ціни при фіксованому рівні якості. Виділені об'єкти утворюють множину Еджворта - Парето. Перерахуємо характерні постановки економічних задач, де вживання нечітких обчислень досягає вражаючих результатів.

- *Стратегічне планерування*. Позиціювати бізнеси, описувати стан ринків, конкурентоспроможності, оцінювати сили і слабкості відповідних бізнесів – все це в чистому вигляді кваліметричні задачі, які можуть бути успішно поставлені і вирішені в нечіткій постановці. І саме на цьому принципі працює система стратегічного планерування, впроваджена в зарубіжному регіональному співтоваристві Сименс.
- *Комплексний аналіз стану корпорації*. Знову кваліметрична задача. Мистецтво полягає в тому, аби виділити чинники і оцінити їх ієрархію з відповідними системами переваги. У простому випадку, коли всі чинники укладаються у вектор, тоді комплексна оцінка досягається в ході двовимірної матричної агрегації поточних якісних рівнів чинників. В рамках цього підходу виявляється можливим одночасно аналізувати всі сторони життя корпорації (фінанси, управління, процеси, задоволеність клієнтів, стратегічне положення корпорації і так далі).
- *Оптимізація фондового портфеля*. Якщо прибутковість компоненти фондового портфеля – нечітка випадкова величина, то параметри відповідних імовірнісних розподілів – нечіткі числа. Якщо і коваріаційна матриця зібрана на нечітких числах, то можливо записати класичну задачу Марковіца в нечіткій постановці, вирішенням якої буде ефективна границя портфельної множини як криволінійна смуга в координатах «ризик-прибутковість».

Якщо розглядати прибутковість активу як нечітке число і нехтувати ефектом впливу коваріаційної матриці, то результуюча прибутковість портфеля – теж нечітке число. Якщо зафіксувати норматив гранично допустимої прибутковості портфеля, то можна оцінити ризик неефективності портфеля. І тоді можна відновити ефективну границю портфельної множини в координатах «очікувана прибутковість – ризик неефективності» портфеля.

- *Оцінка інвестиційної привабливості коштовних паперів*. Кваліметрична задача, де ієрархія формується на фундаментальних чинниках коштовних паперів (відношення ціна-доход, віддача на інвестований капітал, ліквідність активів, фінансова автономія емітента і так далі).

- *Прогнозування фондових індексів*. Якщо побудувати макроекономічну модель, де в якості екзогенних чинників моделі виступають прогнозні рівні макроекономічних параметрів регіону, в якому випускаються коштовні папери, а як виходи моделі виступають фондові індекси, то вихідні прогнози і зв'язки між чинниками усередині моделі можуть мати нечіткий вигляд. Відповідно, результуючі прогнози фондових індексів мають вигляд нечітких функцій.

- *Оцінка нерухомості*. Вибір земельної території під забудову, оцінка вартості будинку або квартири, аналіз перспективності відкриття торгівельної точки, оцінка раціональних рівнів «ціна-дохід» по об'єктах нерухомості – все це кваліметричні задачі.

- *Транспортна логістика*. Якщо ділянки транспортної мережі володіють нечіткою тривалістю перевезення, то розумно вибирати маршрут, що володіє, з одного боку, мінімальною середнечекаємою тривалістю, а, з іншого боку, мінімальним ризиком зриву плану за витратами на перевезення. Всі відповідні оптимальні маршрути утворюють множину Еджворта - Парето.

- *Вибір корпоративної інформаційної системи*. Приклад задачі, де кваліметрія виконується відразу по декількох розрізах (опційні ефекти від впровадження системи, інформаційні ефекти, господарські ризики). При цьому з'являється додатковий рівень невизначеності, пов'язаний з неточністю виміру якісних чинників. Відповідно, з портфеля всіляких послідовностей впровадження модулів інформаційної системи можна виділити підмножину Еджворта - Парето, якщо розглядати його в координатах «інтегральний ефект – невизначеність виміру ефекту».

- *Аналіз новинного фону*. Сьогодні на фондових ринках всі новини про коштовні папери збираються і розподіляються у відповідних розділах інформаційних порталів в автоматичному режимі. Існують технології

смислового розпізнавання тексту. Ці технології можуть бути доповнені технологіями нечіткої класифікації новин, з точки зору їх:

а) корисності для одержувача новин;

б) рівня тривожності відносно стану емітента коштовного паперу.

Всі ці задачі по суті є кваліметричними.

Одним з перспективних напрямів розвитку систем нечітких обчислень є широке застосування нечітких нейромереж. Необхідно відзначити, що глибинна інтеграція нечітких систем і нейромереж пов'язана з розробкою нової архітектури елементів нейромережі. Тобто необхідний перехід від класичних нейронів до нечітких нейронів. Зміна елементу нейромережі для адаптації до нечітких систем може стосуватися вибору функції активації, реалізації операцій складання і множення, оскільки в нечіткій логіці складання моделюється будь-якою трикутною нормою ($\max, a+b-ab$), а операція множення – трикутною нормою ($\min, ab$). Нечітка нейромережа функціонує стандартним чином на основі чітких дійсних чисел. Нечіткою є лише інтерпретація результатів.

Тест.

1. Під поняттям «м’які обчислення» Ви розумієте:

а) сукупність технологій, які пристосовані до роботи з точними, кількісними або якісними даними;

б) сукупність технологій, які пристосовані до роботи з точними, визначеними та істинними даними;

в) сукупність технологій, які пристосовані до роботи з неточними, невизначеними або частково істинними даними.

2. В яких випадках застосування «м'яких обчислень» може дати найкращі результати?

а) в випадку, коли вони можуть застосовуватися для пошуку «достатньо хорошого» рішення задачі за «достатньо короткий час»;

б) в випадку, коли вони можуть застосовуватися для пошуку точного оптимального рішення задачі за «достатньо короткий час»;

в) в випадку, коли вони можуть застосовуватися для пошуку статистичних характеристик рішення задачі за «достатньо короткий час»;

г) в випадку, коли вони можуть застосовуватися для пошуку «достатньо хорошого» рішення задачі за «відведений час».

3. Причиною виникнення теорії нечітких множин, на Ваш погляд, є …

а) необхідність математичної формалізації якісної інформації для побудови математичних моделей;

б) необхідність математичної формалізації нечіткої інформації для побудови математичних моделей;

в) необхідність математичної формалізації кількісної інформації для побудови математичних моделей.

4. Використовуя теорію нечітких множин Ви зазвичай застосовуєте функцію приналежності, яка визначає …

а) ступінь належності елемента впорядкованої множині;

б) ступінь належності елемента словесній множині;

в) ступінь належності елемента універсальній множині;

г) ступінь належності елемента нечіткій множині.

5. Якщо в нечіткій множині визначена підмножина таких точок U, для яких величина $\mu_A(u)$ позитивна, то вона визначається як …

а) висота нечіткої множини;

б) носій нечіткої множини;

в) точки переходу нечіткої множини;

г) ядро нечіткої множини.

6. Елементи множини U, для яких ступінь приналежності $\mu_A(u) = 0.5$ називаються …

а) точками переходу нечіткої множини;

б) висотою нечіткої множини;

в) носієм нечіткої множини.

7. Множина елементів універсуму, в яких міра нечіткості дорівнює 1 отримала назву …

а) границі нечіткої множини;

б) ядра нечіткої множини;

в) носія нечіткої множини.

8. Нечітка множина має вигляд $A = (0/a, 0.5/b, 0/c, 0.7/d, -0.85/e)$. Носієм цієї множини є …

а) $S(A) = \{b, c, d, e\}$;

б) $S(A) = \{a, b, c, d\}$;

в) $S(A) = \{b, d\}$;

г) $S(A) = \{a, b, c, d, e\}$.

9. Якщо нечітка множина $A = \{0.3/a, 0.74/d, 0.7/c, 0.68/f, 0.9/b\}$, то множиною α-рівня при $\alpha = 0.7$ буде множина

а) $A_{07} = \{d, c, b\}$;

б) $A_{07} = \{a, d, c,\}$;

в) $A_{07} = \{d, c, f\}$.

10. Задані дві нечіткі множини $A = 0.8/u_1 + 0.2/u_2 + 0.4/u_3 + 1/u_4$ та $B = 0.7/u_1 + 0.9/u_2 + 0.1/u_3 + 1/u_4$. Тоді їх перетином є …

а) $A \cap B = 0.7/u_1 + 0.2/u_2 + 0.4/u_3 + 1/u_4$;

б) $A \cap B = 0.7/u_1 + 0.9/u_2 + 0.1/u_3 + 1/u_4$

в) $A \cap B = 0.8/u_1 + 0.2/u_2 + 0.1/u_3 + 1/u_4$

г) $A \cap B = 0.7/u_1 + 0.2/u_2 + 0.1/u_3 + 1/u_4$.

11. Задані дві нечіткі множини $A = 0.8/u_1 + 0.2/u_2 + 0.4/u_3 + 1/u_4$ та $B = 0.7/u_1 + 0.9/u_2 + 0.1/u_3 + 1/u_4$. Тоді їх об'єднанням є …

а) $A \cup B = 0.7/u_1 + 0.9/u_2 + 0.4/u_3 + 1/u_4$;

б) $A \cup B = 0.8/u_1 + 0.9/u_2 + 0.4/u_3 + 1/u_4$;

в) $A \cup B = 0.8/u_1 + 0.9/u_2 + 0.1/u_3 + 1/u_4$.

12. На Ваш погляд лінгвістична змінна відрізняється від числової змінної тим, що …

а) її значеннями є числові та функціональні залежності на формальній мові;

б) її значеннями є числа, а також слова або словосполучення в природній або формальній мові;

в) її значеннями є не числа, а слова або словосполучення в природній або формальній мові.

13. При застосуванні лінгвістичної змінної Ви повинні вказати …

а) її найменування, терм-множину, синтаксичну та семантичну процедури;

б) її найменування, функціональну залежність, синтаксичну та семантичну процедури;

в) її найменування, терм-множину, функцію приналежності;

г) її найменування, терм-множину, синтаксичну та семантичну процедури, універсальну множину.

14. Під нечіткими числами зазвичай розуміють …

а) нечіткі змінні, які визначені на числовій осі;

б) чіткі змінні, які визначені на числовій осі;

в) нечіткі змінні, які визначені як слова або словосполучення.

15. При застосуванні нечіткого логічного виводу виконується

а) апроксимація залежності кожної вихідної лінгвістичної змінної від вхідних лінгвістичних змінних і здобуття висновку у вигляді нечіткої множини з використанням бази даних і логічних операцій;

б) апроксимація залежності кожної вихідної лінгвістичної змінної від вхідних лінгвістичних змінних і здобуття висновку у вигляді нечіткої множини з використанням нечіткої бази знань і нечітких операцій;

в) пошук залежності кожної вихідної лінгвістичної змінної від вхідних лінгвістичних змінних і здобуття висновку у вигляді чіткої множини з використанням бази даних і нечітких операцій.

16. Якщо йде розмова про нечітку базу знань, то Ви розумієте, що це є …

а) сукупність нечітких правил «Якщо – то, інакше…», що визначають взаємозв'язок між входами і виходами досліджуваного об'єкту;

б) сукупність нечітких правил «Коли – тоді», що визначають взаємозв'язок між входами і виходами досліджуваного об'єкту;

в) сукупність нечітких правил «Якщо – то», що визначають взаємозв'язок між входами і виходами досліджуваного об'єкту.

17. Коли при інтелектуальному аналізі застосовується нечіткий алгоритм, то мова йде про …

а) впорядковану множину нечітких інструкцій, які містять поняття, що формалізуються чіткими множинами;

б) впорядковану множину чітких інструкцій, які містять поняття, що формалізуються нечіткими множинами;

в) впорядковану множину нечітких інструкцій, які містять поняття, що формалізуються нечіткими множинами.

18. При впроваджені в інтелектуальний аналіз даних нечітких нейронних мереж слід розуміти, що …

а) нечіткі нейронні мережі здійснюють висновки на основі апарату нечіткої логіки, проте параметри функцій приналежності настроюються з використанням нечітких алгоритмів навчання нейронних мереж;

б) нечіткі нейронні мережі здійснюють висновки на основі апарату нечіткої логіки, проте параметри функцій приналежності настроюються з використанням алгоритмів навчання нейронних мереж;

в) нечіткі нейронні мережі здійснюють висновки на основі апарату нечіткої логіки, проте параметри функцій приналежності настроюються з використанням алгоритмів побудови нейронних мереж.

19. Під нечіткими асоціативними правилами Ви розумієте …

а) інструмент для витягання з баз даних закономірностей, які формулюються у вигляді лінгвістичних висловів;

б) інструмент для витягання з баз даних закономірностей, які формулюються у вигляді нечітких правил;

в) інструмент для витягання з баз даних закономірностей, які формулюються у вигляді лінгвістичних функцій.

20. Нечітка кластеризація передбачає, що …

а) дозволяється декільком об'єктам належати одночасно декільком кластерам, але з різною мірою;

б) дозволяється одному і тому ж об'єкту належати одночасно декільком кластерам, але з різною мірою;

в) дозволяється одному об'єкту належати тільки одному кластерам, але з різною мірою приналежності.

Розділ 7.

КЛАСИЧНІ ТЕХНОЛОГІЇ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

7.1. Класичні технології класифікації в Data Mining

Класифікація є найбільш простою і одночасно найбільш часто вирішуваною задачею Data Mining [1, 9, 29, 58]. Зважаючи на поширеність задач класифікації необхідне чітке розуміння суті цього поняття. Наведемо декілька означень.

Означення 1. Класифікація - системний розподіл предметів, явищ, процесів, які вивчаються, за родами, видами, типами, за якими-небудь істотними ознаками для зручності їх дослідження; групування вихідних понять і розташування їх у певному порядку, що відображає міру цієї схожості.

Означення 2. Класифікація - впорядкована за деяким принципом множина об'єктів, які мають схожі класифікаційні ознаки (одну або декілька властивостей), вибраних для визначення схожості або відмінності між цими об'єктами.

Класифікація вимагає дотримання наступних правил:

- у кожному акті ділення необхідно застосовувати лише одну підставу;
- ділення має бути відповідним, тобто загальний обсяг видових понять повинен дорівнювати обсягу поділеного родового поняття;
- члени ділення повинні взаємно виключати одне одного, їх обсяги не повинні перехрещуватися;
- ділення має бути послідовним.

В класичній теорії розглядають:

- допоміжну (штучну) класифікацію, яка виконується за зовнішньою ознакою і служить для придання множині предметів (процесів, явищ) потрібного порядку;

- природну класифікацію, яка виконується за суттєвими ознаками, що характеризують внутрішню спільність предметів і явищ. Вона є результатом і важливим засобом наукового дослідження, оскільки передбачає і закріплює результати вивчення закономірностей об'єктів, що класифікуються.

Залежно від вибраних ознак, їх поєднання і процедури ділення понять класифікація може бути:

- простою - ділення родового поняття лише за ознакою і лише один раз до розкриття всіх видів. Прикладом такої класифікації є дихотомія, при якій членами ділення бувають лише два поняття, кожне з яких є таким, що суперечить іншому (тобто дотримується принцип: « і не »);
- складною - застосовується для ділення одного поняття за різними підставами і синтезу таких простих ділень в єдине ціле. Прикладом такої класифікації є періодична система хімічних елементів.

Ми під класифікацією будемо розуміти віднесення об'єктів (спостережень, подій) до одного із заздалегідь відомих класів. Класифікація - це закономірність, що дозволяє робити висновок відносно визначення характеристик конкретної групи. Таким чином, для проведення класифікації мають бути присутніми ознаки, що характеризують групу, до якої належить та або інша подія або об'єкт (зазвичай при цьому на підставі аналізу вже класифікованих подій формулюються деякі правила).

Класифікація відноситься до стратегії навчання з вчителем (supervised learning), яка також іменують контрольованим або керованим навчанням. Задачею класифікації часто називають передбачення категоріальної залежної змінної (тобто залежної змінної, що є категорією) на основі вибірки безперервних і категоріальних змінних. Наприклад, можна передбачити, хто з клієнтів фірми є потенційним покупцем певного товару, а хто - ні, хто скористається послугою фірми, а хто - ні, і так далі. Цей тип задач відноситься до задач бінарної класифікації, в них залежна змінна може набувати лише два значення (наприклад, «так чи ні», «0 або 1»). Інший

варіант класифікації (багатокласова класифікація) виникає, якщо залежна змінна може приймати значення з деякої множини зумовлених класів. Наприклад, коли необхідно передбачити, яку марку автомобіля захоче купити клієнт. У цих випадках розглядається множина класів для залежної змінної.

Класифікація може бути одновимірною (за однією ознакою) і багатовимірною (по двом і більше ознакам). Багатовимірна класифікація була розроблена біологами при вирішенні проблем дискримінації для класифікації організмів. Однією з перших робіт, присвячених цьому напряму, рахують роботу Р. Фішера, в якій організми розділялися на підвиди залежно від результатів вимірів їх фізичних параметрів. Біологія була і залишається найбільш жаданим і зручним середовищем, для розробки багатовимірних методів класифікації.

Розглянемо задачу класифікації на простому прикладі [15]. Допустимо, є база даних про клієнтів морського курорту з інформацією про вік і дохід за місяць. Є рекламний матеріал двох видів: дорожчий і комфортніший відпочинок і дешевший, традиційний відпочинок. Відповідно, визначено два класи клієнтів: клас 1 і клас 2. База даних приведена в таблиці 7.1.

Табл. 7.1.

База даних клієнтів туристичного агентства

Код клієнта	Вік	Дохід	Клас
1	25	250001	
2	26	800001	
3	30	700001	
4	32	120000	1
5	22	150002	
6	27	290001	
7	21	200002	
8	20	230002	
9	22	750001	

Необхідно визначити, до якого класу належить новий клієнт і який з двох видів рекламних матеріалів йому варто посилати.

Для наочності представимо базу даних в двомірному вимірі (вік – вісь і дохід – вісь), у вигляді множини об'єктів, що належать класам 1 і 2 (рис. 7.1). Рішення нашої задачі полягатиме в тому, аби визначити, до якого класу відноситься новий клієнт (біла мітка).

Рис. 7.1. Множина об'єктів бази даних в двомірному вимірі.

Мета процесу класифікації полягає в тому, аби побудувати модель, яка використовує прогнозуючі атрибути як вхідні параметри і отримує значення залежного атрибуту. Процес класифікації полягає в розбитті множини об'єктів на класи по певному критерію. Класифікатором називається деяка сутність, що визначає, якому із зумовлених класів належить об'єкт по вектору ознак.

Для проведення класифікації за допомогою математичних методів необхідно мати формальний опис об'єкту, яким можна оперувати, використовуючи математичний апарат класифікації. Таким описом в нашому випадку виступає база даних. Кожен об'єкт (запис бази даних) несе інформацію про деяку властивість об'єкту. Набір даних розбивають на дві множини: навчальну і тестову. Навчальна множина (training set) - множина, яка включає дані, що використовуються для навчання (конструювання) моделі. Така множина містить вхідні і вихідні (цільові) значення прикладів. Вихідні значення призначені для навчання моделі. Тестова (test set) множина також містить вхідні і вихідні значення прикладів. Тут вихідні значення використовуються для перевірки працездатності моделі.

Процес класифікації складається з двох етапів: конструювання моделі і її використання.

Конструювання моделі передбачає опис множини зумовлених класів. Зокрема, на цьому етапі виконуються наступні дії:

- кожен приклад набору даних відноситься до одного зумовленого класу;
- використовується навчальна множина, на якій відбувається конструювання моделі;
- отримана модель представляється класифікаційними правилами, деревом рішень або математичною моделлю.

Використання моделі здійснює класифікацію нових або невідомих значень. Зокрема, на цьому етапі виконуються такі дії:

1. Оцінюється правильність та точність моделі, тобто:

- відомі значення з тестового прикладу порівнюються з результатами використання отриманої моделі;
- рівень точності визначається як відсоток правильно класифікованих прикладів в тестовій множині;
- тестова множина не повинна залежати від навчальної множини.

2. Якщо точність моделі допустима, можливе використання моделі для класифікації нових прикладів, клас яких невідомий.

Процес класифікації, а саме, конструювання моделі, представлений на рис. 7.2.

Оцінка точності класифікації може проводитися за допомогою крос-перевірки.

Крос-перевірка (Cross-validation) - це процедура оцінки точності класифікації на даних з тестової множини, яку також називають крос-перевірочною множиною. Точність класифікації тестової множини порівнюється з точністю класифікації навчальної множини. Якщо процедура дає приблизно такі ж результати по точності, то вважається, що дана модель пройшла крос-перевірку.

Рис. 7.2. Конструювання моделі класифікації.

При виборі методів класифікації слід проводити їх оцінювання, виходячи з таких характеристик: швидкість, робастність, інтерпретуємість, надійність. Швидкість характеризує час, який потрібний на створення моделі і її використання. Робастність, тобто стійкість до яких-небудь порушень деяких передумов, означає можливість роботи із зашумленими даними і пропущеними значеннями в даних. Інтерпретуємість забезпечує можливість розуміння моделі аналітиком. Надійність методів класифікації передбачає можливість роботи цих методів за наявності в наборі даних шумів і викидів.

Розглянемо деякі прикладні задачі, які ефективно вирішуються методами класифікації.

Задачі медичної діагностики. В ролі об'єктів виступають пацієнти. Ознаки характеризують результати обстежень, симптоми захворювання і методи лікування, що застосовувалися. Приклади бінарних ознак: стать, наявність головного болю, слабкості. Порядкова ознака - тяжкість стану (задовільний, середньої тяжкості, важкий, вкрай важкий). Кількісні ознаки - вік, пульс, артеріальний тиск, вміст гемоглобіну в крові, доза препарату. Ознаковий опис пацієнта є, по суті справи, формалізованою історією хвороби. Накопивши достатню кількість прецедентів в електронному вигляді, можна вирішувати різні задачі:

- класифікувати вигляд захворювання (диференціальна діагностика);
- визначати найбільш доцільний спосіб лікування;
- передбачати тривалість і результат захворювання;
- оцінювати ризик ускладнень;
- знаходити синдроми - найбільш характерні для даного захворювання.

Цінність такого роду систем в тому, що вони здатні миттєво аналізувати і узагальнювати величезну кількість прецедентів - можливість, недоступна фахівцеві-лікареві.

Передбачення родовищ корисних копалин. Ознаками є дані геологічної розвідки. Наявність або відсутність тих або інших порід на території району кодується бінарними ознаками. Фізико - хімічні властивості цих порід можуть описуватися як кількісними, так і якісними ознаками. Навчальна вибірка складається з прецедентів двох класів: районів відомих родовищ і схожих районів, в яких копалина, що цікавить, виявлена не була. При пошуку рідких корисних копалин кількість об'єктів може виявитися набагато менше, ніж кількість ознак. У цій ситуації погано працюють класичні статистичні методи. Задача вирішується шляхом пошуку закономірностей в наявному масиві даних. В процесі рішення виділяються короткі набори ознак, що володіють найбільшою інформативністю - здатністю щонайкраще розділяти класи. По аналогії з медичними задачами, можна сказати, що відшукуються «синдроми» родовищ. Це важливий результат дослідження, що представляє значний інтерес для геофізиків і геологів.

Оцінювання кредитоспроможності позичальників. Ця задача вирішується банками при видачі кредитів. Потреба в автоматизації процедури видачі кредитів вперше виникла в період буму кредитних карт 60-70-х років в США і інших європейських країнах. Об'єктами в даному випадку є фізичні або юридичні особи, що претендують на здобуття кредиту. В разі фізичних осіб ознаковий опис складається з анкети, яку заповнює сам позичальник, і, можливо, додаткової інформації, яку банк збирає про нього з власних джерел. Приклади бінарних ознак: стать, наявність телефону. Номінальні ознаки - місце мешкання, професія, працедавець. Порядкові ознаки - освіта, посада. Кількісні ознаки - сума кредиту, вік, стаж роботи, дохід сім'ї, розмір заборгованостей в інших банках. Навчальна вибірка складається з позичальників з відомою кредитною історією. У простому випадку ухвалення рішень зводиться до класифікації позичальників на два класи: «хороших» і «поганих». Кредити видаються лише позичальникам першого класу. У складнішому випадку оцінюється сумарне число балів (score)

позичальника, набраних по сукупності інформативних ознак. Чим вище оцінка, тим більше надійним вважається позичальник. На стадії навчання виконується синтез і відбір інформативних ознак і визначається, скільки балів призначати за кожну ознаку, аби ризик рішень, що приймаються, був мінімальний. Наступна задача - вирішити, на яких умовах видавати кредит: визначити процентну ставку, термін погашення, і інші параметри кредитного договору. Ця задача також може бути вирішення методами навчання по прецедентах.

Серед важливих задач, які також вирішуються методами класифікації, слід відзначити задачу передбачення відтоку клієнтів, оптичне розпізнавання символів, розпізнавання мови, виявлення спаму, класифікація документів та інше.

Існує велика різноманітність методів класифікації. Найбільш поширеними з них є:

- класифікація методом опорних векторів;
- байєсівська класифікація;
- статистичні методи, зокрема, лінійна регресія;
- класифікація за допомогою методу найближчого сусіда;
- класифікація CBR-методом;
- класифікація за допомогою штучних нейронних мереж;
- класифікація за допомогою дерев рішень;
- класифікація за допомогою генетичних алгоритмів.

Зупинимося докладніше на головних з них, які отримали найбільше практичне застосування. Зазначимо, що три останні методи вже були розглянуті відповідно в розділах 3, 4, 5.

Метод опорних векторів. У 60–70-і роки колективом математиків під керівництвом В. Н. Вапника був розроблений метод узагальненого портрета, заснований на побудові оптимальної розділяючої гіперплощини. Вимога оптимальності полягало в тому, що навчальні об'єкти мають бути віддалені від розділяючої поверхні настільки далеко, наскільки це

можливо. У 90-і роки метод здобув широку світову популярність і після деякої переробки і серії узагальнень став називатися машиною опорних векторів (Support Vector Machine - SVM). В даний час він вважається одним з кращих методів класифікації.

Метод опорних векторів відноситься до групи граничних методів. Він визначає класи за допомогою границь областей. За допомогою даного методу вирішуються задачі бінарної класифікації.

Метод SVM володіє декількома чудовими властивостями. По-перше, навчання SVM зводиться до задачі квадратичного програмування, яка має єдине рішення, яке обчислюється досить ефективно навіть на вибірках в сотні тисяч об'єктів. По-друге, рішення володіє властивістю розрідженості: положення оптимальної розділяючої гіперплощини залежить лише від невеликої долі навчальних об'єктів. Вони і називаються опорними векторами; останні об'єкти фактично не задіюються. Нарешті, за допомогою введення функції ядра метод узагальнюється на випадок нелінійних розділяючих поверхонь.

В основі методу лежить поняття площин рішень. Площина (plane) рішення розділяє об'єкти з різною класовою приналежністю. Розглянемо задачу класифікації для двох класів, що не перетинаються, в якій об'єкти описуються - мірними векторами: , . Побудуємо лінійну функцію класифікації вигляду

,

де - ознаковий опис об’єкту , вектор та скалярний поріг є параметрами алгоритму (рис. 7.3).

Рис. 7.3. Лінійно-подільна вибірка.

Об'єкти і знаходяться на границі розділяючої смуги. Вектор нормалі до розділяючої гіперплощини визначає ширину смуги. Рівняння описує гіперплощину, що розділяє класи в просторі . Новий об'єкт, що потрапляє

направо, класифікується як об'єкт класу або - як об'єкт класу , якщо він розташувався ліворуч від розділяючої прямої.

Умова задає смугу, що розділяє класи. Жодна з точок навчальної вибірки не може лежати усередині цієї смуги. Границями смуги є дві паралельні гіперплощини з направляючим вектором . Точки, найближчі до розділяючої гіперплощини, лежать точно на границях смуги (рис. 7.4).

Рис. 7.4. Розділяюча смуга.

Аби розділяюча гіперплощина якнайдалі відстояла від точок вибірки, ширина смуги має бути максимальною і визначається як

.

Ширина смуги максимальна, коли норма вектора мінімальна. Таким чином, метод відшукує зразки, що знаходяться на границях між двома класами, тобто опорні вектора; що змальовані на рис. 7.5.

Рис. 7.5. Опорні вектори.

Опорними векторами називаються об'єкти множини, що лежать на границях областей. Класифікація вважається хорошою, якщо область між границями порожня.

Сформулюємо умови задачі пошуку оптимальної розділяючої смуги для випадку лінійної роздільності. Нехай існують обмеження . При цих обмеженнях і є константами, оскільки це елементи навчальної множини, і є змінними. Потрібно знайти такі і , аби виконувалися всі лінійні обмеження, і при цьому якомога менше була норма вектора (отже, ширше розділяюча смуга), тобто необхідно мінімізувати: . Така задача називається задачею квадратичної оптимізації - при лінійних обмеженнях знайти мінімум квадратичної функції.

Найкращою функцією класифікації є функція, для якої очікуваний ризик мінімальний. Поняття очікуваного ризику в даному випадку означає очікуваний рівень помилки класифікації. Безпосередньо оцінити очікуваний рівень помилки побудованої моделі неможливо, це можна зробити за допомогою поняття емпіричного ризику. Проте слід враховувати, що мінімізація останнього не завжди приводить до мінімізації очікуваного ризику. Цю обставину слід пам'ятати при роботі з відносно невеликими наборами навчальних даних. Емпіричний ризик - рівень помилки класифікації на навчальному наборі. Таким чином, в результаті рішення задачі методом опорних векторів для даних, що лінійно розділяються, ми отримуємо функцію класифікації, яка мінімізує верхню оцінку очікуваного ризику.

Однією з проблем, пов'язаних з вирішенням задач класифікації даним методом, є та обставина, що не завжди можна легко знайти лінійну границю між двома класами. У таких випадках один з варіантів - збільшення розмірності, тобто перенесення даних з площини в тривимірний простір, де можливо побудувати таку площину, яка ідеально розділить множину зразків на два класи. Опорними векторами в цьому випадку служитимуть об'єкти з обох класів, які є екстремальними. Таким чином, за допомогою додавання так званого оператора ядра і додаткової розмірності, знаходяться границі між класами у вигляді гіперплощин. Проте слід пам'ятати: складність побудови SVM-моделей полягає в тому, що чим вище розмірність простору, тим складніше з ним працювати. Один з варіантів роботи з даними високої розмірності - це попереднє вживання якого-небудь методу пониження розмірності даних для виявлення найбільш істотних компонент, а потім використання методу опорних векторів.

Як і будь-який інший метод, метод SVM має свої сильні і слабкі сторони, які слід враховувати при виборі даного методу. Недолік методу полягає в тому, що для класифікації використовується не вся множина зразків, а

лише їх невелика частина, яка знаходиться на границях. Достоїнство методу полягає в тому, що для класифікації методом опорних векторів, на відміну від більшості інших методів, досить невеликого набору даних. При правильній роботі моделі, побудованої на тестовій множині, цілком можливо вживання даного методу на реальних даних.

Байєсівська класифікація. Байєсівські процедури класифікації розроблені на основі теореми Байєса і спеціально призначені для роботи з вхідними даними високої розмірності. Не дивлячись на простоту таких процедур, результати їх роботи по своїх характеристиках можуть перевершити результати роботи досить складних алгоритмів класифікації. Спочатку байєсівська класифікація використовувалася для формалізації знань експертів в експертних системах, зараз байєсівська класифікація широко застосовується як один з методів Data Mining [45, 61].

З метою продемонструвати основні принципи роботи байєсівських процедур класифікації, розглянемо приклад (рис. 7.6). Як видно, об'єкти можуть бути розділені на два класи: GREEN або RED. Наша мета - класифікувати нові спостереження в міру їх поступлення, тобто потрібно вирішити до якого класу вони належать, використовуючи інформацію про приналежність класам вже наявних в нашому розпорядженні об'єктів.

Рис. 7.6. Класи об’єктів.

Оскільки об'єктів типа GREEN в два рази більше об'єктів типа RED, розумно передбачити, що шанси приналежності нового спостереження класу GREEN, в два рази більше шансів належати класу RED. В термінах байєсівського аналізу це припущення називається апріорною ймовірністю. Апріорна ймовірність визначається накопиченим досвідом (у нашому випадку процентним співвідношенням об'єктів типа GREEN і RED). Ця величина зазвичай використовується для передбачення результатів до їх реального настання. Таким чином, ми можемо записати:

Prior probability for GREEN = Number of GREEN objects/Total number of objects

Prior probability for RED = Number of RED objects/Total number of objects

Оскільки загальне число об'єктів - 60, 40 з них належать класу GREEN і 20 - класу RED, то апріорна ймовірність приналежності класу буде:

Prior probability for GREEN = 40/60

Prior probability for RED = 20/60

Визначивши апріорну ймовірність, ми готові класифікувати новий об'єкт (білий круг) (рис. 7.7). Через хороше угрупування об'єктів, розумно передбачити, що чим більше об'єктів типа GREEN (або RED) потрапляє в окресність точки , тим вірогідніше, що нове спостереження належатиме цьому класу.

Рис. 7.7. Класифікація нового об’єкта.

Для обчислення міри правдоподібності, проведемо коло з центром в точці , яка охопить апріорі вибране число точок безвідносно до їх класової приналежності. Потім підраховується число точок кожного типа. За цими даними обчислюємо міру правдоподібності

Likelihood of X given GREEN = Number of GREEN

in the vicinity of X / Total number of GREEN cases

Likelihood of X given RED = Number of RED

in the vicinity of X / Total number of RED cases

На наведеній вище ілюстрації видно, що міра правдоподібності приналежності класу GREEN нижче за відповідне значення для класу RED, оскільки коло включає 1 об'єкт типа GREEN і 3 об'єкти типа RED. Отже

Probability of X given GREEN = 1/40

Probability of X given RED = 3/40

Хоча апріорна ймовірність вказує на можливу приналежність спостереження класу GREEN (об'єктів типа GREEN в два рази більше об'єктів типа RED), величина міри правдоподібності приводить до протилежного висновку: належить класу RED (у околиці точки об'єктів типа RED більш ніж об'єктів типа GREEN). Кінцеве класифікуюче рішення в байєсівському аналізі приймається на основі двох джерел інформації: апріорній імовірності і мірі правдоподібності. Для визначення апостеріорної ймовірності застосовується правило Байєса (названо на честь Thomas Bayes).

Posterior probability of X being GREEN = Prior probability of GREEN *

Likelihood of X given GREEN = 4/6 * 1/40 = 1/60

Posterior probability of X being RED = Prior probability of RED *

Likelihood of X given RED = 2/6 * 3/40 = 1/40

В результаті ми класифікуємо як об'єкт типа RED, оскільки апостеріорна ймовірність приналежності цьому класу має найбільшого значення.

«Наївний» (спрощений) алгоритм Байєса (naive-bayes approach) є одним з ефективних алгоритмів класифікації. Точність класифікації, здійснюваної «наївним» алгоритмом, порівнюється з точністю більшості інших алгоритмів. З точки зору швидкості навчання, стабільності на різних даних і простоти реалізації, «наївний» алгоритм Байєса перевершує практично всі відомі ефективні алгоритми класифікації.

Навчання алгоритму виконується шляхом визначення відносних частот значень всіх атрибутів вхідних даних при фіксованих значеннях атрибутів класу. Класифікація здійснюється шляхом застосування правила Байєса для обчислення умовної ймовірності кожного класу для вектора вхідних атрибутів. Вхідний вектор приписується класу, умовна ймовірність якого при даному значенні вхідних атрибутів максимальна. «Наївність» алгоритму полягає в припущенні, що вхідні атрибути умовно (для кожного значення класу) незалежні один від одного, тобто для всіх атрибутів , і значень класу . Це припущення є дуже сильним, і, у багатьох випадках

неправомірним, що робить факт ефективності класифікації за допомогою «наївного» алгоритму Байєса досить несподіваним.

Результатом роботи методу є так звані «прозорі» моделі. Властивостями наївної класифікації є:

1. Використання всіх змінних і визначення всіх залежностей між ними.
2. Наявність двох припущень відносно змінних:

- всі змінні є однаково важливими;
- всі змінні є статистично незалежними, тобто значення однієї змінної нічого не говорить про значення іншої.

В зв'язку з цим закономірний пошук таких модифікацій в алгоритмі, які ослабили б передумову про умовну незалежність атрибутів. Модифікацією алгоритму, що вирішують ці проблеми, можуть служити так звані байєсівські мережі. Байєсівською мережею називається направлений граф без циклів, що дозволяє представляти спільний розподіл випадкових змінних. Кожен вузол графа представляє випадкову змінну, а дуги – прямі залежності між ними. Точніше, мережа описує наступний вислів: кожна змінна залежить лише від безпосередніх предків. Таким чином, граф описує обмеження на залежність змінних одну від одної, що зменшує кількість параметрів спільного розподілу. Параметри спільного розподілу кодуються в наборі таблиць для кожної змінної у формі умовних розподілів за умови на значення змінних-предків. Структура графа і умовні розподіли вузлів при значеннях їх предків однозначно описують спільний розподіл всіх змінних, що дозволяє вирішувати задачу класифікації як визначення значення змінної класу, що максимізувало її умовну ймовірність при заданих значеннях вхідних змінних (рис. 7.8).

Рис. 7.8. Граф «наївної» байєсівської мережі.

Вочевидь, що «наївний» алгоритм Байєса є частинним випадком байєсівської мережі, де кожна вхідна змінна залежить лише від змінної класу, яка є єдиним коренем графа.

Навчання байєсівських мереж стало одним з актуальних напрямів обчислювальної математики і до цих пір є предметом активних досліджень. Проте, до цих пір визначення структури байєсівської мережі в загальному вигляді є складним завданням як з теоретичної, так і з обчислювальної точки зору. Підхід в загальному вигляді володіє наступними недоліками:

- обчислювальна складність;
- при спробі врахувати велику кількість залежностей між змінними, оцінки умовної ймовірності набувають великої дисперсії, оскільки їх спільна поява в даних є маловірогідною подією. Таким чином, оцінки параметрів можуть стати недостовірними, що у результаті може приводити до погіршення якості класифікації навіть в порівнянні з «наївним» алгоритмом Байєса;
- через велику кількість параметрів, модель виходить дуже орієнтованою на навчальні дані. Це приводить до дуже добрих результатів класифікації на навчальних даних і незадовільних результатів на тестових даних. Тобто модель описує не загальні закономірності в структурі даних, а швидше набір окремих випадків в навчальній вибірці.

Відзначають також такі достоїнства байєсівських мереж як методу Data Mining:

- у моделі визначаються залежності між всіма змінними, це дозволяє легко обробляти ситуації, в яких значення деяких змінних невідомі;
- байєсівські мережі досить просто інтерпретуються і дозволяють на етапі прогностичного моделювання легко проводити аналіз сценарію «що, … якщо»;
- байєсівський метод дозволяє природним чином поєднувати закономірності, виведені з даних, і, наприклад, експертні знання, отримані в явному вигляді;

- використання байєсівських мереж дозволяє уникнути проблеми перенавчання (overfitting), тобто надлишкового ускладнення моделі, що є слабкою стороною багатьох методів (наприклад, дерев рішень і нейронних мереж).

Для вирішення позначених проблем використовуються обмеження на структуру графа і розглядаються такі розширення «наївної» моделі Байєса, де кожен вузол додатково може мати не більш за одного предка серед інших вхідних змінних. Модель з такими обмеженнями отримала назву TAN (Tree Augmented Naive Bayes). Оптимальна структура TAN-моделі (з точки зору функції правдоподібності) відповідає моделі з максимальною сумарною умовною по змінній класу взаємної інформації між вузлами і їх предками.

В іншому випадку застосовують байєсівські мережі, які моделюють послідовності змінних - динамічні байєсівські мережі та гібридні байєсівські мережі.

Нещодавно байєсівська класифікація була запропонована для персональної фільтрації спаму. Перший фільтр був розроблений Полем Грахемом (Paul Graham). Для роботи алгоритму потрібне виконання двох вимог. Перша вимога - необхідно, аби у об'єкта, що класифікується, була присутня достатня кількість ознак. Цьому ідеально задовольняють всі слова листів користувача, за винятком зовсім коротких і таких, що дуже рідко зустрічаються. Друга вимога - постійне перенавчання і поповнення набору «спам - не спам». Такі умови дуже добре працюють в локальних поштових клієнтах, оскільки потік «не спаму» у кінцевого клієнта досить постійний, а якщо змінюється, то не швидко.

Проте для всіх клієнтів сервера точно визначити потік «не спаму» досить складно, оскільки один і той же лист, що є для одного клієнта спамом, для іншого спамом не є. Словник виходить дуже великим, не існує чіткого розділення на спам і «не спам», в результаті якість класифікації, в даному випадку рішення задачі фільтрації листів, значно знижується.

Метод «найближчого сусіда» або системи міркувань на основі аналогічних випадків.

У багатьох прикладних задачах вимірювати міру схожості об'єктів істотно простіше, ніж формувати ознакові описи. Наприклад, набагато легко порівняти дві фотографії і сказати, що вони належать одній людині, чим зрозуміти, на підставі яких ознак вони схожі. Такі ситуації часто виникають при розпізнаванні часових рядів або символьних послідовностей. Вони характеризуються тим, що «сирі» вихідні дані не годяться як ознакові описи, але в той же час, існують ефективні і змістовно обґрунтовані способи оцінити міру схожості будь-якої пари «сирих» описів.

Є ще одна характерна особливість цих задач. Якщо міра схожості введена досить вдало, то виявляється, що схожим об'єктам, як правило, відповідають схожі відповіді. У задачах класифікації це означає, що схожі об'єкти набагато частіше лежать в одному класі, чим в різних. Якщо задача в принципі піддається рішенню, то границя між класами не може «проходить всюди»; класи утворюють компактно локалізовані підмножини в просторі об'єктів. Це припущення прийнято називати гіпотезою компактності.

Для формалізації поняття «схожості» вводиться функція відстані або метрика у просторі об'єктів . Алгоритми, засновані на аналізі схожості об'єктів, часто називають метричними, навіть в тих випадках, коли функція не задовольняє всім аксіомам метрики.

Для довільного об'єкту розташуємо елементи навчальної вибірки в порядку зростання відстаней до :

,

де через позначається номер -го сусіда об'єкту . Відповідно, відповідь на -му сусідові об'єкту є . Фактично, будь-який об'єкт породжує свою перенумерацію вибірки .

Означення 3. Метричний алгоритм класифікації з навчальною вибіркою відносить об'єкт до того класу , для якого сумарна вага найближчих навчальних об'єктів максимальна:

, ,

де вагова функція оцінює міру важливості -го сусіда для класифікації об'єкту . Функція називається оцінкою близькості об'єкту до класу .

Навчальна вибірка відіграє роль параметра алгоритму . Налаштування зводиться до запам'ятовування вибірки, і, можливо, оптимізації якихось параметрів вагової функції, проте самі об'єкти не піддаються обробці і зберігаються «як є». З цієї причини метричні алгоритми відносяться до методів міркування по прецедентах (case-based reasoning, CBR). Тут дійсно можна говорити про «міркування», оскільки на питання «чому об'єкт був віднесений до класу ?» алгоритм може дати сповна зрозуміле пояснення: «тому, що є схожі з ним прецеденти класу », і пред'явити список цих прецедентів.

Прецедент - це опис ситуації у поєднанні з детальною вказівкою дій, що робляться в даній ситуації. Підхід, заснований на прецедентах, умовно можна поділити на наступні етапи:

- збір детальної інформації про поставлену задачу;
- зіставлення цієї інформації з деталями прецедентів, що зберігаються в базі, для виявлення аналогічних випадків;
- вибір прецеденту, найбільш близького до поточної проблеми, з бази прецедентів;
- адаптація вибраного рішення до поточної проблеми, якщо це необхідно;
- перевірка коректності кожного знов отриманого рішення;
- занесення детальної інформації про новий прецедент в базу прецедентів.

Таким чином, вивід, заснований на прецедентах, є такий метод аналізу даних, який робить висновки відносно даної ситуації за результатами пошуку аналогій, що зберігаються в базі прецедентів.

Даний метод за своєю суттю відноситься до категорії «навчання без вчителя», тобто є «самонавчальною» технологією, завдяки чому робочі характеристики кожної бази прецедентів з часом і накопиченням прикладів покращуються. Розробка баз прецедентів по конкретній предметній області відбувається на природній для людини мові, отже, може бути виконана найбільш досвідченими співробітниками компанії - експертами або аналітиками, що працюють в даній предметній області. Проте це не означає, що CBR-системи самостійно можуть приймати рішення. Останнє завжди залишається за людиною, даний метод лише пропонує можливі варіанти рішення і вказує на «найрозумніший» із його точки зору.

Вибираючи вагову функцію , можна отримувати різні метричні класифікатори:

- метод найближчого сусіда (1NN);
- метод найближчих сусідів (NN);
- метод зважених найближчих сусідів;
- метод парзеновського вікна ширини ;
- метод парзеновського вікна змінної ширини;
- метод потенційних функцій.

Алгоритм найближчого сусіда (nearest neighbor, NN) є найпростішим алгоритмом класифікації. Він відносить класифікуємий об'єкт до того класу, якому належить найближчий навчальний об'єкт . Навчання NN зводиться до запам'ятовування вибірки . Єдине достоїнство цього алгоритму - простота реалізації. Недоліків значно більше:

- нестійкість до похибок. Якщо серед навчальних об'єктів є викид - об'єкт, що знаходиться в оточенні об'єктів чужого класу, то не лише він сам буде класифікований невірно, але і ті об'єкти, що оточують його, і для яких він виявиться найближчим, також будуть класифіковані невірно;
- відсутність параметрів, які можна було б налаштовувати по вибірці. Алгоритм повністю залежить від того, наскільки вдало вибрана метрика .
- в результаті - низька якість класифікації.

З метою усунення недоліків попереднього методу був розроблений алгоритм найближчих сусідів (nearest neighbors, NN). Аби згладити шумовий вплив викидів, класифікуватимемо об'єкти шляхом голосування по найближчих сусідів. Кожен з сусідів , голосує за віднесення об'єкту до свого класу . Алгоритм відносить об'єкт до того класу, який набере більше число голосів

.

При цей алгоритм збігається з попереднім, отже, нестійкий до шуму. При , навпаки, він надмірно стійкий і вироджується в константу. Таким чином, крайні значення небажані. На практиці оптимальне значення параметра визначають по критерію ковзаючого контролю з виключенням об'єктів поодинці (leave-one-out, LOO). Для кожного об'єкта перевіряється, чи правильно він класифікується по своїх найближчих сусідах.

.

Якщо класифікуємий об'єкт не виключати з навчальної вибірки, то найближчим сусідом завжди буде сам , і мінімальне (нульове) значення функціонала LOO( ) досягатиметься при .

Ще один спосіб задати ваги сусідам - визначити як функцію від відстані , а не від рангу сусіда реалізується в методі парзеновського вікна. Введемо функцію ядра , що не зростає на [0,), і розглянемо алгоритм

.

Параметр називається шириною вікна і грає приблизно ту ж роль, що і число сусідів . «Вікно» - це сферична околиця об'єкту радіусу , при попаданні в яку навчального об'єкту об'єкт «притягується» до класу .

Розглянемо докладніше принципи роботи методів для вирішення задач класифікації і регресії (прогнозування).

Спочатку дослідимо рішення задачі класифікації нових об'єктів. Ця задача схематично змальована на рис. 7.9. Приклади (відомі екземпляри) відмічені знаком «+» або «-», що визначає приналежність до відповідного класу («+»

або «-»), а новий об'єкт, який потрібно класифікувати, позначений кружечком. Нові об'єкти також називають точками запиту.

Рис. 7.9. Класифікація об'єктів множини при різному значенні параметра .

Наша мета полягає в оцінці (класифікації) відгуку точок запиту з використанням спеціальний вибраного числа їх найближчих сусідів. Іншими словами, ми хочемо взнати, до якого класу слід віднести точку запиту: як знак «+» або як знак «-».

Спершу розглянемо результат роботи методу найближчих сусідів з використанням одного найближчого сусіда. В цьому випадку відгук точки запиту буде класифікований як знак плюс, оскільки найближча сусідня точка має знак плюс. Тепер збільшимо число використовуваних найближчих сусідів до двох. Цього разу метод найближчих сусідів не зможе класифікувати відгук точки запиту, оскільки друга найближча точка має знак мінус і обоє знаки рівноцінні (тобто перемога з однаковою кількістю голосів). Далі збільшимо число використовуваних найближчих сусідів до 5. Таким чином, буде визначена ціла околиця точки запиту (на графіці її границя відмічена колом). Оскільки в області міститься 2 точки із знаком «+» і 3 точки із знаком «-», алгоритм найближчих сусідів привласнить знак «-» відгуку точки запиту.

Далі розглянемо принцип роботи методу найближчих сусідів для вирішення задачі регресії. Регресійні задачі пов'язані з прогнозуванням значення залежної змінної по значеннях незалежних змінних набору даних. Розглянемо графік, показаний на рис. 7.10.

Рис. 7.10. Рішення задачі прогнозування при різних значеннях параметра .

Змальований на ньому набір точок отриманий як зв'язок між незалежною змінною і залежною змінною (крива). Заданий набір об'єктів (тобто набір

прикладів); ми використовуємо метод найближчих сусідів для передбачення виходу точки запиту по даному набору прикладів.

Спочатку розглянемо як приклад метод найближчих сусідів з використанням одного найближчого сусіда, тобто при . Шукатимемо набір прикладів і виділяємо з їх числа найближчий до точки запиту . Для нашого випадку найближчий приклад - точка ( ). Вихід (тобто), таким чином, приймається як результат передбачення виходу (тобто). Отже, для одного найближчого сусіда можемо записати: вихід = .

Далі розглянемо ситуацію, коли , тобто розглянемо двох найближчих сусідів. В цьому випадку ми виділяємо вже дві найближчі до точки. На нашому графіку це точки і відповідно. Обчисливши середнє їх виходів, записуємо рішення для у вигляді =( )/2. Рішення задачі прогнозування здійснюється шляхом перенесення описаних вище дій на використання довільного числа найближчих сусідів таким чином, що вихід точки запиту обчислюється як середньоарифметичне значення виходів найближчих сусідів точки запиту.

Незалежні і залежні змінні набору даних можуть бути як неперервними, так і категоріальними. Для неперервних залежних змінних задача розглядається як задача прогнозування, для дискретних змінних - як задача класифікації. Критичним моментом у використанні методу найближчих сусідів є вибір параметра . Він один з найбільш важливих чинників, що визначають якість прогнозної або класифікаційної моделі. Якщо вибрано дуже маленьке значення параметра , виникає ймовірність великого розкиду значень прогнозу. Якщо вибране значення дуже велике, це може привести до сильної зміщеності моделі. Таким чином, ми бачимо, що має бути вибране оптимальне значення параметра . Тобто це значення має бути настільки великим, аби звести до мінімуму ймовірність невірної класифікації, і одночасно, досить малим, аби сусідів були розташовані досить близько до точки запиту. Таким чином, розглядається як згладжуючий параметр, для

якого має бути знайдений компроміс між силою розмаху (розкиду) моделі і її зміщеностью.

Один з варіантів оцінки параметра - проведення крос-перевірки. Така процедура реалізована, наприклад, в пакеті STATISTICA. Крос-перевірка - відомий метод здобуття оцінок невідомих параметрів моделі. Основна ідея методу - розділення вибірки даних на «складки», тобто, випадковим чином виділені ізольовані підвибірки. По фіксованому значенню будується модель найближчих сусідів для здобуття передбачень на -му сегменті (останні сегменти при цьому використовуються як приклади) і оцінюється помилка класифікації. Для регресійних задач найчастіше як оцінка помилки виступає сума квадратів, а для класифікаційних задач зручніше розглядати точність (відсоток коректно класифікованих спостережень). Далі процес послідовно повторюється для всіх можливих варіантів вибору . Після вичерпання «складок», обчислені помилки усереднюються і використовуються як міра стійкості моделі. Вищеописані дії повторюються для різних , і значення, відповідне найменшій помилці (або найбільшій класифікаційній точності), приймається як оптимальне. Слід враховувати, що крос-перевірка - обчислювально ємка процедура, і необхідно надати час для роботи алгоритму, особливо якщо об'єм вибірки досить великий.

Другий варіант вибору значення параметра - самостійно задати його значення. Проте цей спосіб слід використовувати, якщо є обґрунтовані припущення відносно можливого значення параметра, наприклад, попередні дослідження схожих наборів даних.

Прикладом реального використання описаного вище методу є програмне забезпечення центру технічної підтримки компанії Dell, розроблене компанією Inference. Ця система допомагає співробітникам центру відповідати на більше число запитів, відразу пропонуючи відповіді на поширені питання і дозволяючи звертатися до бази під час розмови по телефону з користувачем. Співробітники центру технічної підтримки,

завдяки реалізації цього методу, можуть відповідати одночасно на значне число дзвінків. Програмне забезпечення CBR зараз розгорнуте в мережі Intranet компанії Dell.

Інструментів Data Mining, що реалізовують розглянуті методи не надто багато. Серед найбільш відомих: CBR Express і Case Point (Inference Corp.), Apriori (Answer Systems), DP Umbrella (VYCOR Corp.), KATE tools (Acknosoft, Франція), Pattern Recognition Workbench (Unica, США), а також деякі статистичні пакети, наприклад, Statistica.

Алгоритми обмеженого перебору. Алгоритми обмеженого перебору були запропоновані в середині 60-х років М. М. Бонгардом для пошуку логічних закономірностей в даних. З тих пір вони продемонстрували свою ефективність при вирішенні множини задач з самих різних областей. Ці алгоритми обчислюють частоти комбінацій простих логічних подій в підгрупах даних. Приклади простих логічних подій: , , та ін., де - деякий параметр, і - константи. Обмеженням служить довжина комбінації простих логічних подій. На підставі аналізу обчислених частот робиться висновок про корисність тієї або іншої комбінації для встановлення асоціації в даних, для класифікації, прогнозування та інше.

Найбільш яскравими сучасними представниками цього підходу є системи WizWhy та WizRule компанії WizSoft. Хоча автор системи Абрам Мейдан не розкриває специфіку алгоритму, покладеного в основу роботи програмних комплексів, за результатами ретельного тестування системи були зроблені виводи про наявність тут обмеженого перебору (вивчалися результати, залежності часу їх здобуття від числа аналізованих параметрів і ін.).

Як WizWhy так і WizRule роблять, по суті, одне і те ж: переглядають задану базу даних і, зібравши статистику, відшукують правила і закономірності, яким підкоряються відомості, зібрані в базі. Потім програми поводяться по-різному. WizRule застосовує виведені закономірності до тільки що проаналізованої бази і відшукує записи, де ці закономірності порушуються,

тобто велика вірогідність того, що у зміст записів вкрались помилки. WizRule можна назвати засобом підтримки цілісності баз даних. WizWhy, проаналізувавши базу даних, дає можливість користувачеві зайнятися передбаченнями і прогнозами. Людина вводить значення відомих йому параметрів, а WizWhy, ґрунтуючись на виявлених нею в базі закономірностях, видає найбільш вірогідні значення невідомих параметрів (рис. 7.11).

Рис 7.11. Система WizWhy виявила правила, що пояснюють низьку врожайність деяких сільськогосподарських ділянок.

Автор WizWhy стверджує, що його система виявляє всі логічні правила в даних. Насправді це, звичайно, не так. По-перше, максимальна довжина комбінації в «if - then» правилі в системі WizWhy дорівнює 6, і, по-друге, з самого початку роботи алгоритму виконується евристичний пошук простих логічних подій, на яких потім будується весь подальший аналіз. Зрозумівши ці особливості WizWhy, неважко було запропонувати просте тестове завдання, яке система не змогла взагалі вирішити. Інший момент - система видає рішення за прийнятний час лише для порівняно невеликої розмірності даних. Проте, система WizWhy є на сьогоднішній день одним з лідерів на ринку продуктів Data Mining. Це не позбавлено підстав. Система постійно демонструє вищі показники при вирішенні практичних задач, чим всі останні алгоритми.

7.2. Програмне забезпечення задач класифікації

Розробка програмних систем інтелектуального аналізу даних і, зокрема, комп'ютерною підтримки рішення задач класифікації активно ведуться в провідних зарубіжних країнах. Перш за все, це статистичні пакети обробки

даних і візуалізації, в основі яких лежать методи різних розділів математичної статистики - перевірка статистичних гіпотез, регресійний аналіз, дисперсійний аналіз, аналіз часових рядів, і ін. Використання статистичних програмних продуктів стало стандартним і ефективним інструментом рішення задач класифікації, і, перш за все, початкового етапу досліджень, коли знаходяться значення різних усереднених показників, перевіряється статистична достовірність різних гіпотез, знаходяться регресійні залежності. В той же час статистичні підходи мають і істотні недоліки. Вони дозволяють оцінити статистичну достовірність значення параметра, гіпотези або залежності, проте самі методи обчислення величин, висунення гіпотез або знаходження залежностей мають очевидні обмеження. Перш за все, знаходяться усереднені по вибірці величини, що може бути досить грубим уявленням про аналізуємі або класифікуємі параметри. Будь-яка статистична модель використовує поняття «випадкових подій», «функцій розподілу випадкових величин» і тому подібне, тоді як взаємозв'язок між різними параметрами досліджуваних об'єктів, ситуацій або явищ є детермінованим. Саме використання статистичних методів передбачає наявність певного числа спостережень для обґрунтованості кінцевого результату, тоді як дане число може бути істотно більше можливого. Таким чином, при аналізі в принципі непредставимих даних, або на етапах початку накопичення даних, статистичні підходи стають неефективними як засіб аналізу і класифікації [6, 17, 25, 47].

Останніми роками з'явилися спеціалізовані пакети інтелектуального аналізу даних. Для даних пакетів характерна орієнтація на широкий круг практичних задач, а їх алгоритмічною основою є сукупність альтернативних моделей. Таким чином, на сьогоднішньому рівні розвитку методів рішення задач інтелектуального аналізу даних і класифікації, переважною представляється дорога застосування програмних засобів, що включають основні існуючі підходи. В даному випадку підвищуються

шанси підбору з наявних алгоритмів такого алгоритму, який забезпечить найбільш точне вирішення задач користувача на нових даних. Іншим важливим атрибутом систем аналізу і класифікації має бути наявність засобів автоматичного вирішення задач класифікації колективами алгоритмів. Дійсно, стандартною ситуацією є наявність декількох альтернативних алгоритмів або рішень, рівнозначних для користувача. Для вибору з них одного найбільш ефективного не вистачає інформації. Тоді природною альтернативою вибору є створення на базі наявних алгоритмів або рішень нових, більш перспективніших.

Розглянемо можливості та практику застосування найбільш відомих програмних пакетів при вирішенні задач класифікації.

Система PolyAnalyst. В пакеті реалізован багатий інструментарій для вирішення задач класифікації та для знаходження правил віднесення записів до одного з двох або до одного з декількох класів [79].

Одним з таких інструментів є модуль Stepwise Linear Regression (LR) - покрокова багатопараметрична лінійна регресія (рис. 7.12).

Рис. 7.12. Результати роботи модуля Stepwise Linear Regression.

Лінійна регресія як широко поширений метод статистичного дослідження, включена в багато статистичних пакетів і електронні таблиці. Проте, реалізація цього модуля в системі PolyAnalyst має свої особливості, а саме, автоматичний вибір найбільш значущих незалежних змінних і ретельна оцінка статистичної значущості результатів. Потрібно відмітити, що в даному випадку значущість відрізняється від значущості одиничної регресійної моделі, оскільки протягом одного запуску обчислювального процесу може бути перевірене велике число регресійних моделей. Алгоритм працює дуже швидко і може бути застосовний для побудови лінійних моделей на змішаних типах даних.

Модуль Memory based reasoning (MR) реалізує метод «найближчих сусідів». У системі PolyAnalyst використовується модифікація відомого алгоритму «метод найближчих сусідів» (рис. 7.13).

Рис. 7.13. Метод найближчих сусідів в системі PolyAnalyst.

Ідея методу дуже проста; для передбачення значення цільової змінної для даного запису, в навчальній таблиці з історичними даними, знаходяться «схожі» записи, для яких відомі значення цільової змінної, і обчислюється середнє з цих значень, яке і вважається прогнозом. На практиці реалізація цієї ідеї зустрічається з трьома основними труднощами:

- що вважати мірою близькості записів;
- скільки записів брати для усереднювання;
- який метод усереднювання використовувати, звичайне або зважене усереднювання.

У системі PolyAnalyst оптимізація цих параметрів виконується на основі генетичних алгоритмів. У цьому і полягає відмінність даної реалізації алгоритму «найближчих сусідів» від відомих аналогів. Алгоритм MR використовується для передбачення значень числових змінних і категоріальних змінних, включаючи текстові (string data type), а також для класифікації на два або декілька класів.

Для задач класифікації система PolyAnalyst також застосовує модулі: Classify (CL) - класифікатор на основі нечіткої логіки (розділ 6), Discriminate (DS) – дикрімінація (розділ 6), Decision Tree (DT) - дерево рішень (розділ 4) та Decision Forest (DF) - ліси рішень.

Засоби аналізу STATISTICA Data Miner. Програмний комплекс STATISTICA включає величезний набір різних аналітичних процедур [76]. Для спрощення роботи користувача в пакет були вбудовані готові закінчені модулі аналізу даних, призначені для вирішення найбільш важливих і

популярних задач: прогнозування, класифікації і так далі. Зокрема, це такі модулі як:

- General Classifier - класифікація. STATISTICA Data Miner включає повний пакет процедур класифікації: узагальнені лінійні моделі, дерева класифікації, регресійні дерева та інші.
- General Modeler/Multivariate Explorer - узагальнені лінійні, нелінійні і регресійні моделі. Даний елемент містить лінійні, нелінійні, узагальнені регресійні моделі і елементи аналізу дерев класифікації.

Окрім них, STATISTICA Data Miner містить набір спеціалізованих процедур Data Mining, які доповнюють лінійку інструментів Data Mining:

- General Classification and Regression Trees (GTrees) - узагальнені класифікаційні і регресійні дерева (GTrees). Модуль є повною реалізацією методів, розроблених Breiman, Friedman, Olshen і Stone. Окрім цього, модуль містить різного роду доопрацювання і доповнення, такі як оптимізації алгоритмів для великих об'ємів даних і так далі. Модуль є набором методів узагальненої класифікації і регресійних дерев.
- Interactive Classification and Regression Trees - інтерактивна класифікація і регресійні дерева. На додаток до модулів автоматичної побудови різного роду дерев, STATISTICA Data Miner також включає засоби для формування таких дерев в інтерактивному режимі.
- Boosted Trees - розширювані прості дерева. Останні дослідження аналітичних алгоритмів показують, що для деяких задач побудови «складних» оцінок, прогнозів і класифікацій використання послідовно збільшуваних простих дерев дає точніші результати, ніж нейроні мережі або складні цілісні дерева. Даний модуль реалізує алгоритм побудови простих збільшуваних (розширюваних) дерев.

Коротко проілюструємо схему роботи в Data Miner. Дії в Data Miner починаються з підміню «Добыча данных» в меню «Анализ» (рис. 7.14). Вибравши пункт «Добытчик данных – Мои процедуры», ми запустимо робоче середовище STATISTICA.

Рис. 7.14. Меню «Добытчик данных».

Після завантаження необхідного файлу у вікні діалогу «Виберите зависимые переменные и предикторы» вибираємо залежні змінні (неперервні і категоріальні) і предиктори (неперервні і категоріальні), виходячи із знань про структуру даних.

Запускаємо «Диспетчер узлов». У даному діалозі, показаному на рис. 7.15, можна вибрати вигляд аналізу або задати операцію перетворення даних.

Рис. 7.15. Меню «Диспетчер узлов».

Диспетчер вузлів включає всі доступні процедури для видобутку даних. Всього доступні близько 260 методів фільтрації і очищення даних та методів аналізу. За умовчанням, процедури поміщені в папки і відсортовані відповідно до типа аналізу, який вони виконують. Проте користувач має можливість створити власну конфігурацію сортування методів. Для того, щоб вибрати необхідний аналіз, необхідно виділити його на правій панелі. У нижній частині діалогу дається опис вибираних методів.

Виберемо, для прикладу, Descriptive Statistics і Standard Classification Trees with Deployment (C And RT) . Вікно Data Miner виглядає таким чином (рис. 7.16).

Рис. 7.16. Вікно Data Miner з вузлами вибраних аналізів.

Тепер виконаємо проект. Всі вузли, сполучені з джерелами даних активними стрілками, будуть проведені (рис. 7.17).

Рис. 7.17. Вікно Data Miner після виконання проекту.

Далі можна проглянути результати (у стовпці звітів). Детальні звіти створюються за умовчанням для кожного виду аналізу. Для робочих книг результатів доступна повна функціональність системи STATISTICA. Крім того, в диспетчерові вузлів STATISTICA Data Miner містяться всілякі процедури для класифікації і дискримінантного аналізу, регресійних моделей і багатовимірного аналізу, а також узагальнені часові ряди і прогнозування. Всі ці інструменти можна використовувати для проведення складного аналізу в автоматичному режимі, а також для оцінювання якості моделі.

Oracle Data Mining. Oracle Data Mining є модулем в Oracle Enterprise Edition. ODM підтримує всі етапи технології інтелектуального аналізу даних, включаючи постановку задачі, підготовку даних, автоматичну побудову моделей, аналіз і тестування результатів, використання моделей в реальних застосуваннях. Важливо, що моделі будуються автоматично на основі аналізу наявних даних про об'єкти, спостереження і ситуації за допомогою спеціальних алгоритмів. Основу модуля ODM складають процедури, що реалізовують різні алгоритми побудови моделей класифікації, регресії, прогнозування.

На етапі підготовки даних забезпечується доступ до будь-яких реляційних баз, текстових файлів, файлів формату SAS. Додаткові засоби перетворення і очищення даних дозволяють змінювати вигляд представлення, проводити нормалізацію значень, виявляти невизначені або відсутні значення. На основі підготовлених даних спеціальні процедури автоматично будують моделі для подальшого прогнозування, класифікації нових ситуацій, виявлення аналогій. ODM підтримує побудову п'яти різних типів моделей. Графічні засоби надають широкі можливості для аналізу отриманих результатів, верифікації моделей на тестових наборах даних, оцінки точності і стійкості результатів. Уточнені і перевірені моделі можна включати в існуючі додатки шляхом генерації їх описів на C, C++, Java, a

також розробляти нові спеціалізовані додатки за допомогою засобу розробки Software Development Kit (SDK), що входить до складу середовища ODM .

Важливою особливістю системи ODM є його технічні характеристики: робота в архітектурі клієнт-сервер, широке використання техніки паралельних обчислень, висока міра масштабованості при збільшенні обчислювальних ресурсів. Oracle Data Mining підтримує наступний спектр алгоритмів класифікації:

- класифікаційні моделі - Na_ve Bayes, Adaptive Bayes Network;
- класифікації і регресійні моделі - Support Vector Machine.

Особливість алгоритмів, реалізованих в Oracle Data Mining, полягає в тому, що всі вони працюють безпосередньо з реляційними базами даних і не вимагають вивантаження і збереження даних в спеціальних форматах. Окрім власне алгоритмів, в модуль ODM входять засоби підготовки даних, оцінки результатів, застосування моделей до нових наборів даних.

Засоби бізнес-аналізу в SQL Server. В програмному комплексі реалізовані засоби Data Mining, які доступні користувачам цієї СУБД. В якості прикладів алгоритмів розглянемо Microsoft Decision Trees і байєсівський алгоритм.

Алгоритм Microsoft Decision Tree є алгоритмом класифікації, що дозволяє прогнозувати як безперервні, так і дискретні атрибути на основі оцінки в процесі навчання моделі міри впливу вхідних атрибутів на прогнозований атрибут і побудови ієрархічної структури, яка базується на відповіді «так чи ні» на набір питань. Алгоритми побудови дерев рішень дозволяють передбачити значення якого-небудь параметра для заданого випадку (наприклад, чи поверне вчасно чоловік виданий йому кредит) на основі великої кількості даних про інші подібні випадки (зокрема, на основі відомостей про інших осіб, яким видавалися кредити).

Байєсівський алгоритм (Naive Bayes) дозволяє в процесі навчання моделі обчислити ймовірність, з якою кожен можливий стан кожного вхідного

атрибуту приводить до кожного стану прогнозованого атрибуту, а потім використовувати результати розрахунків для прогнозування. Цей алгоритм підтримує лише дискретні атрибути.

Розробка моделей Data Mining здійснюється за допомогою SQL Server Business Intelligence Development Studio. Для виконання вказаної задачі використовується шаблон Analysis Services Project. Робота над моделлю починається з вказівки джерел даних і із створення представлення джерел даних, на основі якого генеруватиметься модель (рис. 7.18).

Рис. 7.18. Представлення джерел даних для подальшого створення моделей.

Створивши представлення джерел даних, можна приступати до розробки структури моделі (Mining Structure), вибравши відповідний пункт контекстного меню папки Mining structures. При відповіді на питання майстра вибираються модель Data Mining, використовуване представлення джерел даних, таблиця, що містить рядки, призначені для навчання моделі (кожна такий рядок в термінах Data Mining носить назву Case), прогнозовані атрибути і вхідні атрибути. Створеною структурою можна скористатися повторно, наприклад для розробки моделі, що використовує інший алгоритм. Це дозволяє змінити алгоритм Data Mining без повторного вибору таблиць і полів в тому випадку, якщо аналіз із застосуванням первинного вибраного алгоритму привів до виводу про необхідність його заміни. Описавши структуру моделей, ми можемо перенести моделі на сервер аналітичних служб і здійснити навчання моделей. Відзначимо, що процес навчання моделей виконується однократно, тоді як процес передбачення, що виконується багато разів, здійснюється досить швидко.

Коли один з алгоритмів побудови дерев рішень застосовується до набору вхідних даних, результат відображується у вигляді дерева. Подібні алгоритми дозволяють здійснити декілька рівнів такого розділення, ділячи отримані групи на дрібніші на підставі інших ознак до тих пір, поки значення, які передбачається передбачати, не стануть однаковими (або, в разі безперервного значення параметра, що передбачається, близькими) для всіх отриманих груп. Саме ці значення і застосовуються для здійснення передбачень на основі даної моделі.

Дія алгоритмів побудови дерев рішень базується на застосуванні методів регресійного і кореляційного аналізу. У реалізації Microsoft Decision Tree розділення виконується на основі найбільш високого для описуваних даних коефіцієнта кореляції між параметром, згідно якому відбувається розділення, і параметром, який надалі має бути передбачений. Проглянути отримане дерево рішень можна за допомогою закладки Decision Tree засобу Mining Model Viewer (мал. 7.19).

Рис. 7.19. Отримане дерево класифікації.

Ієрархія, що представлена в отриманому дереві, створена на основі класифікації даних за правилом «Якщо, то...», причому насиченість кольору гілок залежить від кількості вхідних даних, що попали у вказану гілку. При цьому на гістограмі в окремому вікні можна побачити процентний розподіл можливих результатів передбачення для вибраної гілки. Гілці дерева можна розкривати до самого нижнього рівня, при цьому на нижньому рівні в розподілі можливих результатів передбачення зазвичай переважає якесь одне значення. Іншими словами, алгоритм побудови дерев рішень дозволяє визначити набір значень характеристик, що дозволяють відокремити одну категорію даних від іншої, - цей процес називають сегментацією.

Для вивчення взаємозалежності атрибутів в даних, використаних для навчання моделі, можна скористатися закладкою Dependency Network засобу Mining Model Viewer (рис. 7.20).

Рис. 7. 20. Вивчення взаємозалежності атрибутів.

У центральній частині представленої схеми розташований атрибут, що передбачається, довкола нього - атрибути, що використовуються для прогнозування. При цьому, переміщаючи покажчик в лівій частині вікна, можна оцінити міру впливу кожного атрибуту на результат, що передбачається: чим нижче покажчик, тим менше зв'язків відображуватиметься на представленій схемі.

За допомогою байєсівського алгоритму можна виявити відмінності у впливі, що накладається на прогнозований атрибут різними станами вхідного атрибуту. Для вивчення взаємозалежності атрибутів, визначеної за допомогою байєсівського алгоритму, можна скористатися закладкою Dependency Network засобу Mining Model Viewer.

Закладка Attribute Profiles призначена для оцінки того, яким чином різні значення вхідних атрибутів впливають на атрибут, що передбачається (рис. 7.21).

Рис. 7.21. Профілі атрибутів.

Побудувавши моделі і оцінивши їх точність, в тому випадку, якщо остання представляється задовільною, можна перейти безпосередньо до прогнозів. Для цієї мети зазвичай створюються запити на мові Data Mining Extensions (DMX). Код запиту можна як написати вручну, так і згенерувати за допомогою інструменту Prediction Query Builder на закладці Mining Model Prediction інструменту Data Mining Designer. Типові результати

класифікації для заздалегідь сформованого набору даних представлені на рис. 7.22.

Рис. 7.22. Результати роботи моделі.

Байєсівські алгоритми. Компанія Norsys Software Corp. - спеціалізується в розробці програмного забезпечення для байєсівських мереж. Програма Netica - основне досягнення компанії, яка стала комерційно доступною в 1995 році. В даний час Netica є одним з найпоширених інструментів для розробки байєсівських мереж (рис. 7.23).

Рис. 7.23. Приклад байєсівської мережі в додатку Netica.

Netica - потужна, зручна в роботі програма для роботи з графовими імовірнісними моделями. Вона має інтуїтивний і приємний інтерфейс користувача для введення топології мережі. Співвідношення між змінними можуть бути задані, як індивідуальна ймовірність, у формі рівнянь, або шляхом автоматичного навчання з файлів даних. Створені мережі можуть бути використані незалежно, і як фрагменти крупніших моделей, формуючи тим самим бібліотеку модулів. При створенні мережевих моделей доступний широкий спектр функцій і інструментів. Багато операцій можуть бути зроблені декількома клацаннями миші, що робить систему Netica зручною для пошукових досліджень, для навчання і для простого перегляду, а також для навчання моделі байєсівської мережі.

Однією з успішних розробок компанії BayesWare Ltd є програмній комплекс Bayesware Discoverer, оснований на моделях байєсівських мереж. Програма здатна автоматично будувати причинну ймовірнісну модель на основі баз

даних, використовуючи байєсівську мережу. Bayesware Discoverer автоматично визначає змінні, створює граф і визначає кількість залежностей як розподіл ймовірностей. Програма може бути застосована для дослідження різних сценаріїв відповідно до розробленої моделі, як наприклад, наявність різних ймовірнісних розподілів. Вона має вбудовану діалогову 3D графіку, яка підтримує візуалізацію складних взаємодій (рис. 7.24).

Рис. 7.24. Приклад роботи Bayesware Discoverer.

Основний продукт компанія Hugin Expert програмний комплекс Hugin почав створюватися під час робіт за проектом ESPRIT, в якому системи, засновані на знаннях, використовувалися для проблеми діагностування нервово-м'язових захворювань. До теперішнього моменту Hugin адаптована в 25 різних країнах, вона використовується у ряді різних областей, пов'язаних з аналізом рішень, підтримкою ухвалення рішень, передбаченням, діагностикою, управлінням ризиками і оцінками безпеки технологій.

Hugin є програмою реалізацією системи ухвалення рішень на основі байєсівських мереж довіри. Ця система має розвинений інтерфейс і дозволяє досить просто створювати бази знань і фактів. Використовує два основні режими роботи:

- режим редагування і побудови причинно-наслідкової мережі, а також заповнення таблиць умовної вірогідності, що є кількісним описом бази знань;
- режим розрахунку ймовірнісних оцінок для ухвалення рішення по всіх подіях, що входять в причинно-наслідкову мережу. Розрахунки можуть здійснюватися як на основі класичної теорії Байєса, так і на основі методів теорії можливостей.

Програмний комплекс Hugin має можливість зв'язку з основними найбільш поширеними програмними засобами фірми Microsoft. Дана система має всі основні функції будь-якої інформаційної системи, включаючи такі як: зберігання даних, діагностика помилок в роботі і т. д (рис. 7.25).

Рис. 7. 25. Приклад роботи програмного комплексу Hugin.

Байєсівські імовірнісні методи є істотним кроком вперед, в розвитку технологій інтелектуального аналіза даних. Вони дають зрозуміле пояснення своїх виводів, допускають логічну інтерпретацію і модифікацію структури стосунків між змінними задачі, а також дозволяють в явній формі врахувати апріорний досвід експертів відповідної предметної області. Завдяки вдалому представленню у вигляді графів, байєсівські мережі вельми зручні в призначених для користувача застосуваннях.

Байєсівська методологія, насправді, ширше, ніж сімейство способів операції з умовною вірогідністю в орієнтованих графах. Вона включає також моделі з симетричними зв'язками (випадкові поля і решітки), моделі динамічних процесів (марківські ланцюги), а також широкий клас моделей з прихованими змінними, що дозволяють вирішувати імовірнісні задачі класифікації, розпізнавання образів і прогнозування. В найближчому майбутньому передбачається значно розширити вживання байєсівських мереж довіри. Наприклад, на одному з сайтів пошукових систем конструюються байєсівські мережі для моделювання успішних запитів, що поступають від користувачів. Ці мережі можуть поповнювати реєстраційний файл пошукового сервера категоріями передбачуваних цілей інформації, що призначаються, для забезпечення можливості передбачення модифікацій запитів.

Аналіз ринку програмних засобів для використання байєсівських методів демонструє як наявність безкоштовних і Open-Source продуктів (GeNIe & SMILE, OpenBayes, RISO, BANSY3, SamIam) так і комерційних продуктів (AgenaRisk Bayesian network tool, Bayesian network application library, Bayesia, BNet, Dezide, MSBNx, Bayes Net Toolbox for MatLab, dVelox).

Алгоритми міркувань на основі прецедентів. Потужний імпульс технології міркувань на основі прецедентів дала розробка комп'ютерної системи SMART. Система SMART призначена для технічної підтримки замовників корпорації COMPAQ. Коли замовник стикається з деякою проблемою, подробиці передаються в систему. Виконується початковий пошук в бібліотеці прецедентів, аби знайти випадки з подібними ознаками. При недоліку інформації система ставить додаткові питання. Як тільки певний поріг досягнутий (скажімо, прецедент збігається не менше, чим на 80%), пропонується рішення від прецеденту. На додаток до цього, система може бути використана як інструмент навчання. Надалі COMPAQ розширила цю систему, просунувши її безпосередньо до покупців. Система QUICKSOURCE дозволяє користувачеві самому справлятися з проблемами і звертатися в центр підтримки в якості останнього притулку.

У системі KATE TOOLS компанії Acknosoft підтримується спрощений погляд на процес виводу. Вхідна інформація для KATE – це файл, який містить описи ознак і їх значення на спеціальній мові CASUAL. KATE може працювати із складними даними, представленими у вигляді структурованих об'єктів, відношень або навіть загальними знаннями про проблемної області. Але для виявлення схожості між прецедентами використовується одна проста метрика. Основний акцент робиться на відбір прецедентів за допомогою алгоритму «найближчого сусіда». KATE використовує версію алгоритму «найближчого сусіда» для обчислення метрики подібності. Близькість між двома випадками і , що мають ознак обчислюється за формулою

,

Система KATE не пропонує можливостей для автоматичної адаптації рішення. Перевірка коректності рішення неможлива, але є перевірка бази прецедентів на наявність контрприкладів. Все ж, KATE – це ефективна індустріальна система, яка дозволяє використовувати зважені ознаки при обчисленні метрики подібності, а також використовувати визначувану користувачем метрику. Її легко розширювати, тому що всі функції KATE доступні при підключенні супутніх динамічних бібліотек.

Інструментів Data Mining, що реалізовують метод найближчих сусідів і CBR-метод, не надто багато. Серед найбільш відомих: CBR Express і Case Point (Inference Corp.), Apriori (Answer Systems), DP Umbrella (VYCOR Corp.), KATE tools (Acknosoft, Франція), Pattern Recognition Workbench (Unica, США), а також деякі статистичні пакети, наприклад, Statistica.

CBR Express і CasePoint – продукти, призначені для розробки експертних систем, заснованих на прецедентах. CBR Express теж нагромаджує «досвід», забезпечуючи введення, супровід і динамічне додавання прецедентів, а також простий доступ до них за допомогою питань і відповідей. Обоє системи використовуються при автоматизації інформаційно-довідкових служб і «гарячих ліній», а також при створенні інтелектуальних програмних продуктів, систем доступу до інформації, систем публікації знань і так далі. При спілкуванні з системою спочатку вводиться простий запит, наприклад: «Мій комп'ютер не працює». Далі відбувається виділення ключових слів, пошук в базі прецедентів, і генерується перелік потенційних рішень. Користувачеві можуть бути також поставлені уточнюючі питання. Пропоновані варіанти вирішення проблеми можуть включати відео- або фотоматеріали. Технологія виводу по прецедентах є основою для практично безмежних застосувань, які нарощуються за рахунок постійного збору інформації (причому забезпечується поєднання структурованих і неструктурованих даних, включаючи мультимедіа). На думку компаній, що активно використовують

цю технологію, таких як Nippon Steel, Lockheed і деяких інших, створюється самонавчальна колективна пам'ять, виключно зручна для накопичення і передачі професійного досвіду.

На останок зупинимося на двох важливих практичних додатках технологій класифікації.

В останнє десятиліття кількість електронних документів різко зросла, у зв'язку з чим виникла необхідність вирішення різних задач для зручної роботи з ними. Класифікація текстів (text categorization) та сортування текстових документів по заздалегідь визначених категоріях - одна з таких задач.

Методи класифікації текстів лежать на стику двох областей - інформаційного пошуку (information retrieval) і машинного навчання (machine learning). Загальні частини два цих підходів - способи представлення документів і способи оцінки якості класифікації текстів, а відмінності полягають лише в способах власне пошуку. Згідно парадигмі машинного навчання, класифікуюче правило будується поступово на основі тренувальної колекції. Такий підхід забезпечує якість класифікації, порівнянну з якістю класифікації, вироблюваної людиною.

Сферами застосування цього підходу є фільтрація спаму, сортування новин, перевірка авторства, підбір ключових слів, складання інтернет-каталогів, контекстна реклама, зняття неоднозначності (автоматичні перекладачі), автоматичне анотування.

Текст представляється у вигляді мультимножини термів (слів). Кожному слову зіставлене деяке число – вага, яке є характеристикою зустрічаємості цього слова в тексті. Порядок слів, як правило, не враховується, враховується лише частота зустрічається слова в тексті і, можливо, інші ознаки, такі як «слово трапилося в заголовку», «слово виділене іншим кольором» і так далі. На підставі цих ознак кожному слову в тексті зіставляється його вага. Інколи проводиться нормалізація по документу для того, щоб сума квадратів всіх вагів в нім дорівнювала 1. Кожен текст - це вектор в багатовимірному просторі, координати - номери слів, значення координат -

значення вагів. Розмірність вектора - це кількість слів, які зустрічаються в документах. Через те, що враховуються всі слова, які будь-коли зустрілися в документах, вектора виходять з величезною кількістю координат, більшість з яких нулі.

Допустимо, що існує такий лінійно незалежний набір документів , що всі інші вектора документів виражаються як лінійна комбінація векторів цього набору. Тоді можна взяти множину за базис; у такому базисі в кожного документа буде вже координат. Таким чином, можна істотно скоротити розмірність.

Побудова класифікатора зазвичай полягає у визначенні функції , яка для кожного документа повертає значення приналежності (categorization status value) до . Побудову точного класифікатора можна робити двома способами: відразу будувати функцію або спочатку обчислити аналогічну ранжируванню функцію , а потім визначити поріг (threshold) такий, що і інтерпретується як , а інтерпретується як . Вибирати поріг можна декількома способами:

- пропорційний метод. Для кожної категорії на колекції обчислюється, яка доля документів їй належить. Порогове значення вибирається так, щоб на інших колекціях доля документів, віднесених до , була такою ж.
- метод найближчих категорій. Кожен документ вважається таким, що належить до найближчим категоріям і відповідно цього вибирається порогове значення.

Іншою важливою проблемою сучасної інформатики є оцінка релевантності. Релевантність як відповідність між пошуковим запитом і знайденою інформацією є одним з фундаментальних понять при пошуку інформації. Розробка будь-якого пошукового двигуна в мережі Інтернет також передбачає необхідність рішення задачі оцінки релевантності. Зростання об'єму інформації в мережі Інтернет останнім часом істотно утрудняє процес виявлення релевантних документів і фільтрацію не релевантних документів. Особливо це стосується пошуку в певної предметної області.

Пошукові двигуни загального призначення, по-перше, не надають користувачеві можливості звуження зони пошуку, а по-друге, через свою орієнтацію на роботу із слабо структурованими документами не в змозі виробляти оцінку релевантності з врахуванням специфіки конкретної області. Необхідність вирішення цих проблем привела до створення пошукових двигунів вузького призначення – вертикальних пошукових двигунів. Ці пошукові системи обмежуються пошуком в певної предметній області (вертикалі), намагаючись поліпшити якість надаваємих користувачеві результатів.

Для вирішення вказаної проблемі зручно використовувати байєсівські мережі довіри. Одна із задач, для якої успішно застосовуються байєсівські мережі - задачі класифікації. Задачу оцінки релевантності також можна розглядати як задачу класифікації. Дійсно, можна розглянути кожен документ як об’єкт, що належить до однієї з двох областей, що не перетинаються: релевантні документи і не релевантні документи. В такому разі, задача оцінки релевантності документа запиту представляється у вигляді задачі віднесення його до одного з двох класів. В цьому випадку, приналежність документа до першого класу дозволяє свідчити, що цей документ є релевантним запиту.

Така класифікація здійснюється на підставі розрахунку ймовірності приналежності документа до тієї або іншої категорії. Ця ж вірогідність виступає і мірою релевантності, що дозволяє відсікати документі з низькою релевантністю, сортувати множину отриманих результатів та надавати користувачеві можливість вибору порогу релевантності.

7.3. Класичні технології кластеризації в Data Mining

Задача кластеризації схожа із задачею класифікації, є її логічним продовженням, але її відмінність в тому, що класи вивчаємого набору

даних, заздалегідь не зумовлені. Синонімами терміну «кластеризація» є «автоматична класифікація», «навчання без вчителя» і «таксономія» [11, 13, 26].

Кластеризація призначена для розбиття сукупності об'єктів на однорідні групи (кластери або класи). Якщо дані вибірки представити як точки в ознаковому просторі, то задача кластеризації зводиться до визначення «згущувань точок». Мета кластеризації - пошук існуючих структур.

Кластеризація є описовою процедурою, вона не робить жодних статистичних висновків, але дає можливість провести розвідувальний аналіз і вивчити «структуру даних». Само поняття «кластер» визначене неоднозначно: у кожному дослідженні свої «кластери». Переводиться поняття кластер (cluster) як «скупчення», «гроздь». Кластер можна охарактеризувати як групу об'єктів, що мають загальні властивості. Характеристиками кластера можна назвати дві ознаки:

- внутрішня однорідність;
- зовнішня ізольованість.

Питання, що задається аналітиками при вирішенні багатьох задач, полягає в тому, як організувати дані в наглядні структури, тобто розвернути таксономії.

Результатом таксономії є деревоподібна ієрархічна структура. При цьому кожен об'єкт характеризується перерахуванням всіх кластерів, яким він належить, зазвичай від великого до дрібного. Візуально таксономія представляється у вигляді графіка, названого дендрограмою. Класичним прикладом таксономії на основі схожості є біноміальна номенклатура живих істот, запропонована Карлом Ліннеєм в середині XVIII століття. Аналогічні систематизації будуються в багатьох областях знання, аби упорядкувати інформацію про велику кількість об'єктів. Найбільше застосування кластеризація спочатку отримала в таких науках як біологія, антропологія, психологія. Для вирішення економічних задач кластеризація тривалий час мало використовувалася із-за специфіки економічних даних і явищ.

У 1925 р. гідробіолог П. В. Терентьєв розробив так званий «метод кореляційних плеяд», призначений для угрупування корелюючих ознак. Цей метод дав поштовх розвитку методів угрупування за допомогою графів. Термін «кластерний аналіз» вперше був запропонований Тріоном. На початку 50-х років з'явилися публікації Р. Люїса, Е. Фікса і Дж. Ходжеса по ієрархічних алгоритмах кластерного аналізу. Помітний поштовх розвитку робіт по кластерному аналізу дали роботи Р. Розенблатта по розпізнаючому пристрою (персептрону), які поклали початок розвитку теорії «розпізнавання образів без вчителя».

Важливим кроком до розробки методів кластеризації з'явилася книга «Принципи чисельної таксономії», опублікована в 1963г. двома біологами - Робертом Сокелом і Пітером Снітом. Автори цієї книги виходили з того, що для створення ефективних біологічних класифікацій процедура кластеризації повинна забезпечувати використання всіляких показників, що характеризують досліджувані організми, виробляти оцінку міри схожості між цими організмами і забезпечувати розміщення схожих організмів в одну і ту ж групу. При цьому сформовані групи мають бути досить «локальні», тобто схожість об'єктів усередині груп повинна перевершувати схожість груп між собою. Подальший аналіз виділених угрупувань, на думку авторів, може з'ясувати, чи відповідають ці групи різним біологічним видам. Іншими словами, Сокел і Сніт передбачали, що виявлення структури розподілу об'єктів в групи, допомагає встановити процес утворення цих структур. А відмінність і схожість організмів різних кластерів (груп) можуть служити базою для осмислення еволюційного процесу, що відбувався, і з'ясування його механізму.

В цей ж час була запропонована множина алгоритмів таких авторів, як Дж. Мак-Кин, Г. Болл і Д. Холл за методами середніх; Г. Ланса і В. Уїльямса, Н. Джардайна та ін. - за ієрархічними методами. Помітний внесок у розвиток методів кластерного аналізу внесли і вітчизняні учені - Е. М. Браверман, А. А. Дорофеюк, Л. А, Растригін, Ю. І. Журавльов та ін.

Зокрема, в 60-70 рр. великою популярністю користувалися багаточисельні алгоритми розроблені математиками Н. Г. Загоруйко і Г. С. Лобовим. Це такі широко відомі алгоритми, як FOREL, BIGFOR, KRAB, NTTP, DRET, TRF та ін. На їх основі був створений спеціалізований пакет програм ОТЕКС. По приблизних оцінках фахівців число публікацій по кластерному аналізу і його застосуванням в різних областях знання подвоюється кожні три роки.

Які ж причини настільки бурхливого інтересу до цього виду аналізу? Об'єктивно існують три основні причини цього явища. Це поява потужної обчислювальної техніки, без якої кластерний аналіз реальних даних практично не реалізовується. Друга причина полягає в тому, що сучасна наука все сильніше спирається в своїх побудовах на класифікацію. Причому цей процес усе більш поглиблюється, оскільки паралельно цьому йде все більша спеціалізація знань, яка неможлива без досить об'єктивної класифікації. Третя причина - поглиблення спеціальних знань неминуче приводить до збільшення кількості змінних, що враховуються при аналізі тих або інших об'єктів і явищ. Внаслідок цього суб'єктивна класифікація, яка раніше спиралася на досить малу кількість ознак, що враховувалися, часто виявляється вже ненадійною. А об'єктивна класифікація, зі все зростаючим набором характеристик об'єкту, вимагає використання складних алгоритмів кластеризації, які можуть бути реалізовані лише на базі сучасних комп'ютерів. Саме ці причини і породили «кластерний бум».

Формальна постановка задачі кластеризації (або навчання без вчителя) полягає в наступному. Є навчальна вибірка і функція відстані між об'єктами . Потрібно розбити вибірку на підмножини, що не перетинаються, які назвемо кластерами, так, щоб кожен кластер складався з об'єктів, близьких по метриці , а об'єкти різних кластерів істотно відрізнялися. При цьому кожному об'єкту приписується мітка (номер) кластера .

Алгоритм кластеризації - це функція a: X > Y, яка будь-якому об'єкту ставить у відповідність мітку кластера . Множина міток в деяких випадках відома

заздалегідь, проте частіше ставиться задача визначити оптимальне число кластерів, з точки зору того або іншого критерію якості кластеризації.

Потреба в обробці великих масивів даних в Data Mining привела до формулювання вимог, яким, по можливості, повинен задовольняти алгоритм кластеризації. Розглянемо їх:

- мінімальна можлива кількість проходів по базі даних;
- робота в обмеженому об'ємі оперативної пам'яті комп'ютера;
- роботу алгоритму можна перервати із збереженням проміжних результатів, аби продовжити обчислення пізніше;
- алгоритм повинен працювати, коли об'єкти з бази даних можуть витягуватися лише в режимі однонаправленого курсора (тобто в режимі навігації по записах).

Рішення задачі кластеризації принципове неоднозначно, і тому є декілька причин. По-перше, не існує однозначно найкращого критерію якості кластеризації. Відомий цілий ряд досить розумних критеріїв, а також ряд алгоритмів, що не мають чітко вираженого критерію, але що здійснюють досить розумну кластеризацію «по побудові». Всі вони можуть давати різні результати. По-друге, число кластерів, як правило, невідоме заздалегідь і встановлюється відповідно до деякого суб'єктивного критерію. По-третє, результат кластеризації істотно залежить від метрики , вибір якої, як правило, також суб'єктивний і визначається експертом.

Вибір відстані між об'єктами є вузловим моментом дослідження, від нього багато в чому залежить остаточний варіант розбиття об'єктів на класи при даному алгоритмі розбиття. Існує декілька методів визначення функції відстані.

Відстань Евкліда. Найбільш пряма дорога обчислення відстаней між об'єктами полягає в обчисленні відстаней Евкліда. Вона є геометричною відстанню в багатовимірному просторі і обчислюється таким чином

.

Відмітимо, що відстань Евкліда обчислюється по початкових, а не за стандартизованими даними. Це звичайний спосіб її обчислення, який має певні переваги (наприклад, відстань між двома об'єктами не змінюється при введенні в аналіз нового об'єкту, який може виявитися викидом). Проте, на відстані можуть сильно впливати відмінності між осями, по координатах яких обчислюються ці відстані.

Відстань міських кварталів (манхэттенська відстань). Ця відстань є просто середньою різниць по координатах. В більшості випадків ця міра відстані приводить до таких же результатів, як і звичайна відстань Евкліда. Проте відзначимо, що для цієї міри вплив окремих великих різниць (викидів) зменшується (оскільки вони не зводяться в квадрат). Манхеттенська відстань обчислюється за формулою

.

Відстань Чебишева. Ця відстань може виявитися корисною, коли бажають визначити два об'єкти як «різні», якщо вони розрізняються по якій-небудь одній координаті (яким-небудь одним виміром). Відстань Чебишева обчислюється за формулою

Степенна відстань (відстань Мінковського). Інколи виникає необхідність прогресивно збільшити або зменшити вагу, що відноситься до розмірності, для якої відповідні об'єкти сильно відрізняються. Це може бути досягнуто з використанням степенної відстані, яка обчислюється за формулою:

де і - параметри, що визначаються користувачем. Параметр відповідальний за поступове зважування різниць по окремих координатах, параметр відповідальний за прогресивне зважування великих відстаней між об'єктами. Якщо обоє параметра рівні двом, то ця відстань збігається з відстанню Евкліда.

Відсоток незгоди. Ця міра використовується в тих випадках, коли дані є категоріальними. Ця відстань обчислюється за формулою

.

Коли кожен об'єкт є окремим кластером, відстані між цими об'єктами визначаються вибраною мірою. Виникає наступне питання - як визначити відстані між кластерами? Існують різні правила, названі методами об'єднання або зв'язки для двох кластерів.

Метод ближнього сусіда або одиночний зв'язок. Тут відстань між двома кластерами визначається відстанню між двома найбільш близькими об'єктами (найближчими сусідами) в різних кластерах. Цей метод дозволяє виділяти кластери скільки завгодно складної форми за умови, що різні частини таких кластерів сполучені ланцюжками близьких один до одного елементів. В результаті роботи цього методу кластери представляються довгими «ланцюжками» або «волокнистими» кластерами, «зчепленими разом» лише окремими елементами, які випадково виявилися ближчими за останніх один до одного.

Метод найбільш віддалених сусідів або повний зв'язок. Тут відстані між кластерами визначаються найбільшою відстанню між будь-якими двома об'єктами в різних кластерах (тобто «найбільш віддаленими сусідами»). Метод добре використовувати, коли об'єкти дійсно походять з різних «гаїв». Якщо ж кластери мають в деякому роді подовжену форму або їх природний тип є «ланцюжковим», то цей метод не слід використовувати.

Метод Варда (Ward's method). В якості відстані між кластерами береться приріст суми квадратів відстаней об'єктів до центрів кластерів, отримуваний в результаті їх об'єднання. На відміну від інших методів кластерного аналізу для оцінки відстаней між кластерами, тут використовуються методи дисперсійного аналізу. На кожному кроці алгоритму об'єднуються такі два кластери, які приводять до мінімального збільшення цільової функції, тобто внутрішньогрупової суми квадратів. Цей метод направлений на об'єднання близько розташованих кластерів і «прагне» створювати кластери малого розміру.

Метод незваженого попарного середнього (unweighted pair-group method using arithmetic averages, UPGMA). В якості відстані між двома кластерами береться середня відстань між всіма парами об'єктів в них. Цей метод слід використовувати, якщо об'єкти дійсно походять з різних «гаїв», у випадках присутності кластерів типа «ланцюжка», при припущенні нерівних розмірів кластерів.

Метод зваженого попарного середнього (weighted pair-group method using arithmetic averages, WPGMA). Цей метод схожий на метод незваженого попарного середнього, різниця полягає лише в тому, що тут в якості вагового коефіцієнту використовується розмір кластера (число об'єктів, що містяться в кластері). Цей метод рекомендується використовувати саме за наявності припущення про кластери різних розмірів.

Кластеризація (навчання без вчителя) відрізняється від класифікації (навчання з вчителем) тим, що мітки вхідних об'єктів спочатку не задані, і навіть може бути невідома само множина . У цьому сенсі задача кластеризації ще в більшій мірі некоректно поставлене, чим задача класифікації (рис. 7.26).

Рис. 7.26. Порівняння задач класифікації і кластеризації.

Цілі кластеризації можуть бути різними залежно від особливостей конкретної прикладної задачі:

- зрозуміти структуру множини об'єктів , розбивши її на групи схожих об'єктів. Спростити подальшу обробку даних і ухвалення рішень, працюючи з кожним кластером окремо (стратегія «розділяй і володарюй»);
- скоротити об'єм зберігаємих даних в разі надвеликої вибірки , залишивши по одному найбільш типовому представникові від кожного кластера;
- виділити нетипові об'єкти, які не підходять ні до одного з кластерів. Цю задачу називають однокласовою класифікацією, виявленням нетиповості або новизни (novelty detection).

У першому випадку число кластерів прагнуть зробити поменше. У другому випадку важливіше забезпечити високу міру схожості об'єктів усередині кожного кластера, а кластерів може бути скільки завгодно. У третьому випадку найбільший інтерес представляють окремі об'єкти, що не вписуються ні в один з кластерів.

Кластери можуть бути такими, що не перетинаються, або ексклюзивними (non-overlapping, exclusive), і такими, що перетинаються, (overlapping) (мал. 7.27).

Рис. 7.27. Кластери, що перетинаються і не перетинаються.

Слід зазначити, що в результаті застосування різних методів кластерного аналізу можуть бути отримані кластери різної форми. Наприклад, можливі кластери типа «ланцюжка», тобто кластери представлені довгими «ланцюжками», кластери подовженої форми і так далі, а деякі методи можуть створювати кластери довільної форми.

На сьогоднішній день розроблена більше сотні різних алгоритмів кластеризації. Приведемо коротку характеристику підходів до кластеризації.

1. Алгоритми, засновані на розділенні даних (Partitioning Algorithms), в т.ч. ітеративні:

- розділення об'єктів на кластерів;
- ітеративний перерозподіл об'єктів для поліпшення кластеризації.

2. Ієрархічні алгоритми (Hierarchy Algorithms):

- агломерація (Agglomerative Nesting): кожен об'єкт спочатку є кластером, кластери, з'єднуючись один з одним, формують більший кластер і так далі;
- дивізімні методи (Divisive Analysis): характеризуються послідовним розділенням кластера, що складається зі всіх об'єктів, і відповідним збільшенням числа кластерів, що в результаті приводить до створення послідовність розщеплюючих груп.

3. Методи, засновані на концентрації об'єктів (Density-based methods):

- засновані на можливості з'єднання об'єктів;
- ігнорують шуми, знаходження кластерів довільної форми.

4. Грід-методи (Grid-based methods):
- квантування об'єктів в грід-структури.

5. Модельні методи (Model-based):
- використання моделі для знаходження кластерів, найбільш відповідних даним.

6. Методи за способом аналізу даних:
- чіткі;
- нечіткі.

7. Методи по кількості застосування алгоритмів кластеризації:
- з одноетапною кластеризацією;
- з багатоетапною кластеризацією.

8. Методи по можливості розширення об'єму оброблюваних даних:
- масштабовані;
- не масштабовані.

9. Методи за часом виконання кластеризації:
- потокові (on-line);
- не потокові (off-line).

Важливою проблемою кластеризації є оцінка її якості. Задачу кластеризації можна ставити як задачу дискретної оптимізації: необхідно так приписати номери кластерів об'єктам , аби значення вибраного функціонала якості прийняло найкраще значення. Існує багато різновидів функціоналів якості кластеризації, але немає «найправильнішого» функціонала. По суті справи, кожен метод кластеризації можна розглядати як точний або наближений алгоритм пошуку оптимуму деякого функціонала.

Середня внутрішньокластерна відстань має бути якомога менше

.

Середня міжкластерна відстань має бути якомога більше

.

Якщо алгоритм кластеризації обчислює центри кластерів , , то можна визначити обчислювально ефективніші функціонали.

На практиці обчислюють відношення пари функціоналів, аби врахувати як міжкластерні, так і внутрішньокластерні відстані

Оцінка якості кластеризації може бути проведена на основі наступних процедур:

- ручна перевірка;
- встановлення контрольних точок і перевірка на отриманих кластерах;
- визначення стабільності кластеризації шляхом додавання в модель нових змінних;
- створення і порівняння кластерів з використанням різних методів. Різні методи кластеризації можуть створювати різні кластери, і це є нормальним явищем. Проте створення схожих кластерів різними методами вказує на правильність кластеризації.

Кластерний аналіз застосовується в різних областях. Він корисний, коли потрібно класифікувати велику кількість інформації. Так, в медицині використовується кластеризація захворювань, лікування захворювань або їх симптомів, а також таксономія пацієнтів, препаратів і так далі. У археології встановлюються таксономії кам'яних споруд і древніх об'єктів. У менеджменті прикладом задачі кластеризації буде розбиття персоналу на різні групи, класифікація споживачів і постачальників, виявлення схожих виробничих ситуацій, при яких виникає брак. У соціології задача кластеризації - розбиття респондентів на однорідні групи. Загалом, всякий раз, коли необхідно класифікувати «гори» інформації до придатних для подальшої обробки груп, кластерний аналіз виявляється вельми корисним і ефективним.

Досить широко кластерний аналіз застосовується в маркетингових дослідженнях - як в теоретичних дослідженнях, так і практиці вирішення проблем групування різних об'єктів. При цьому вирішуються питання про

групи клієнтів, продуктів і так далі. Так, однією з найбільш важливих задач при використанні кластерного аналізу в маркетингових дослідженнях є аналіз поведінки споживача [48], а саме: групування споживачів в однорідні класи для здобуття максимально повного уявлення про поведінку клієнта з кожної групи і про чинники, що впливають на його поведінку. Ця проблема детально описана в роботах Клакстона, Фрая, Портіса, Кіля і Лейтона. Важливу задачу, яку може вирішити кластерний аналіз, є позиціювання, тобто визначення ніші, в якій слід позиціювати новий продукт, пропонований на ринку. В результаті вживання кластерного аналізу будується карта, по якій можна визначити рівень конкуренції в різних сегментах ринку і відповідні характеристики товару для можливості попадання в цей сегмент. За допомогою аналізу такої карти можливе визначення нових, незайнятих ніш на ринку, в яких можна пропонувати існуючі товари або розробляти нові. Кластерний аналіз також може бути зручний, наприклад, для аналізу клієнтів компанії. Для цього всі клієнти групуються в кластери, і для кожного кластера виробляється індивідуальна політика. Такий підхід дозволяє істотно скоротити об'єкти аналізу, і, в той же час, індивідуально підійти до кожної групи клієнтів.

Задачі кластерного аналізу можна об'єднати в наступні групи:

- розробка типології або класифікації;
- дослідження корисних концептуальних схем групування об'єктів;
- представлення гіпотез на основі дослідження даних;
- перевірка гіпотез або досліджень для визначення, чи дійсно типи (групи), виділені тим або іншим способом, присутні в наявних даних.

Як правило, при практичному використанні кластерного аналізу одночасно вирішується декілька з вказаних задач.

Розглянемо приклад процедури кластерного аналізу. Допустимо, ми маємо набір даних , що складається з 14-ти прикладів, в яких є по дві ознаки та . Дані по ним приведені в таблиці 7.1.

Табл.. 7.1

Набір даних

№ приклада	Ознака	Ознака
1	27	19
2	11	46
3	25	15
4	36	27
5	35	45
6	10	43
7	11	44
8	36	24
9	26	14
10	26	45
11	923	
12	33	15
13	27	16
14	10	48

Представимо змінні та у вигляді діаграми розсіювання (рис. 7. 28).

Рис. 7.28. Діаграма розсіювання змінних та .

На рисунку бачимо декілька груп «схожих» прикладів. Приклади (об'єкти), які по значеннях і «схожі» один на одного, належать до однієї групи (кластеру); об'єкти з різних кластерів не схожі один на одного. Критерієм для визначення схожості і відмінності кластерів є відстань між точками на діаграмі розсіювання. Цю схожість можна «виміряти», вона дорівнює відстані між точками на графіку.

Коли осей більше, ніж дві, відстань розраховується таким чином: сума квадратів різниці координат складається із стількох доданків, скільки осей (вимірів) присутньо в нашому просторі (рис. 7.29).

Рис. 7.29. Відстань між двома точками в просторі трьох вимірів.

У загальному випадку всі етапи кластерного аналізу взаємозв'язані, і рішення, прийняті на одному з них, визначають дії на подальших етапах. Основними етапами кластерного аналізу є:

1. Вибір метрики і методу стандартизації вихідних даних.
2. Визначення кількості кластерів (для ітеративного кластерного аналізу).
3. Визначення методу кластеризації (правила об'єднання або зв'язку). На думку багатьох фахівців, вибір методу кластеризації є вирішальним при визначенні форми і специфіки кластерів.
4. Аналіз результатів кластеризації. Цей етап передбачає вирішення таких питань: чи не є отримане розбиття на кластери випадковим; чи є розбиття надійним і стабільним на підвибірках даних; чи існує взаємозв'язок між результатами кластеризації і змінними, які не брали участь в процесі кластеризації; чи можна інтерпретувати отримані результати кластеризації.
5. Перевірка результатів кластеризації. Результати кластеризації також мають бути перевірені формальними і неформальними методами. Формальні методи залежать від того методу, який використовувався для кластеризації. Неформальні включають наступні процедури перевірки якості кластеризації:

- аналіз результатів кластеризації, отриманих на певних вибірках набору даних;
- крос-перевірка;
- проведення кластеризації при зміні порядку спостережень в наборі даних;
- проведення кластеризації при видаленні деяких спостережень;
- проведення кластеризації на невеликих вибірках.

Один з варіантів перевірки якості кластеризації - використання декількох методів і порівняння отриманих результатів. Відсутність подібності не означатиме некоректність результатів, але присутність схожих груп вважається ознакою якісної кластеризації.

Кластер має наступні математичні характеристики: центр, радіус, середньоквадратичне відхилення, розмір кластера. Центр кластера - це середнє геометричне місце точок в просторі змінних. Радіус кластера - максимальна відстань точок від центру кластера.

Як було відмічено раніше, кластери можуть бути такими, що перекриваються. В цьому випадку неможливо за допомогою математичних процедур однозначно віднести об'єкт до одного з двох кластерів. Такі об'єкти називають спірними. Спірний об'єкт - це об'єкт, який по мірі схожості може бути віднесений до декількох кластерів.

Розмір кластера може бути визначений або по радіусу кластера, або по середньоквадратичному відхиленню об'єктів для цього кластера. Об'єкт відноситься до кластера, якщо відстань від об'єкта до центра кластера менше радіуса кластера. Якщо ця умова виконується для двох і більше кластерів, об'єкт є спірним.

Вибір масштабу в кластерному аналізі має велике значення. Розглянемо приклад. Уявимо собі, що дані ознаки в наборі даних А на два порядки більше даних ознаки : значення змінної знаходяться в діапазоні від 100 до 700, а значення змінної - в діапазоні від 0 до 1. Тоді, при розрахунку величини відстані між точками, що відображають положення об'єктів в просторі їх властивостей, змінна, що має великі значення, тобто змінна , буде практично повністю домінувати над змінною з малими значеннями, тобто змінною . Таким чином із-за неоднорідності одиниць виміру ознак стає неможливо коректно розрахувати відстані між точками. Ця проблема вирішується за допомогою попередньої стандартизації змінних. Стандартизація (standardization) або нормування (normalization) приводить значення всіх перетворених змінних до єдиного діапазону значень шляхом

вираження через відношення цих значень до деякої величини, що відображає певні властивості конкретної ознаки.

Існують різні способи нормування вихідних даних. Два найбільш поширених способа:

- ділення даних на середньоквадратичне відхилення відповідних змінних;
- обчислення Z вкладу або стандартизованого вкладу.

Разом із стандартизацією змінних, існує варіант додання кожній з них певного коефіцієнта важливості, або ваги, який би відображав значущість відповідної змінної. Як ваги можуть виступати експертні оцінки, отримані в ході опиту експертів - фахівців предметної області. Отримані множення нормованих змінних на відповідних ваги дозволяють отримувати відстані між точками в багатовимірному просторі з врахуванням неоднакової ваги змінних. В ході експериментів можливе порівняння результатів, отриманих з врахуванням експертних оцінок і без них, і вибір кращого з них.

Як і будь-які інші методи, методи кластерного аналізу мають певні слабкі сторони, тобто деякі складнощі, проблеми і обмеження. При проведенні кластерного аналізу слід враховувати, що результати кластеризації залежать від критеріїв розбиття сукупності вихідних даних. При пониженні розмірності даних можуть виникнути певні спотворення, за рахунок узагальнень можуть загубитися деякі індивідуальні характеристики об'єктів.

Існує ряд складнощів, які слід продумати перед проведенням кластеризації.

1. Складність вибору характеристик, на основі яких проводиться кластеризація. Необдуманий вибір приводить до неадекватного розбиття на кластери і, як наслідок, - до невірного рішення задачі.
2. Складність вибору методу кластеризації. Цей вибір вимагає непоганого знання методів і передумов їх використання. Аби перевірити ефективність конкретного методу певної предметної області, доцільно застосувати наступну процедуру: розглядають декілька апріорі різних між собою груп і перемішують їх представників між собою випадковим чином. Далі

проводиться кластеризація для відновлення вхідного розбиття на кластери. Доля збігів об'єктів у виявлених і вхідних групах є показником ефективності роботи методу.

3. Проблема вибору числа кластерів. Якщо немає жодних відомостей відносно можливого числа кластерів, необхідно провести ряд експериментів і, в результаті перебору різного числа кластерів, вибрати оптимальне їх число.
4. Проблема інтерпретації результатів кластеризації. Форма кластерів в більшості випадків визначається вибором методу об'єднання. Проте слід враховувати, що конкретні методи прагнуть створювати кластери певних форм, навіть якщо в досліджуваному наборі даних кластерів насправді немає.

Перед проведенням кластеризації у аналітика може виникнути питання, якій групі методів кластерного аналізу віддати перевагу. Методи кластерного аналізу у загальному випадку можна розділити на дві групи: ієрархічні та неієрархічні. Вибираючи між ними, необхідно враховувати наступні їх особливості.

Неієрархічні методи виявляють вищу стійкість по відношенню до шумів і викидів, некоректного вибору метрики, включення незначимих змінних в набір, що бере участь в кластеризації. Ціною, яку доводиться платити за ці достоїнства методу, є слово «апріорі». Аналітик повинен заздалегідь визначити кількість кластерів, кількість ітерацій або правило зупинки, а також деякі інші параметри кластеризації.

Якщо немає припущень відносно числа кластерів, рекомендують використовувати ієрархічні алгоритми кластерного аналізу. Проте якщо об'єм вибірки не дозволяє це зробити, можлива дорога - проведення ряду експериментів з різною кількістю кластерів, наприклад, почати розбиття сукупності даних з двох груп і, поступово збільшуючи їх кількість, порівнювати результати. За рахунок такого «варіювання» результатів досягається чимала гнучкість кластеризації.

Ієрархічні методи, на відміну від неієрархічних, відмовляються від визначення числа кластерів, а будують повне дерево вкладених кластерів. Складнощі ієрархічних методів кластеризації: обмеження об'єму набору даних; вибір міри близькості; негнучкість отриманих класифікацій. Перевага цієї групи методів порівняно з неієрархічними методами - їх наочність і можливість отримати детальне уявлення про структуру даних. При використанні ієрархічних методів існує можливість досить легко ідентифікувати викиди в наборі даних і, в результаті, підвищити якість даних. Ця процедура лежить в основі двокрокового алгоритму кластеризації. Такий набір даних надалі може бути використаний для проведення неієрархічної кластеризації.

Існує ще один аспект, який полягає в можливості кластеризації всієї сукупності даних або ж її вибірки. Названий аспект істотний для обох груп методів, проте він критичніший для ієрархічних методів. Ієрархічні методи не можуть працювати з великими наборами даних, а використання деякої вибірки, тобто частини даних, могло б дозволити застосовувати ці методи.

Результати кластеризації можуть не мати достатнього статистичного обґрунтування. З іншого боку, при вирішенні задач кластеризації допустима нестатистична інтерпретація отриманих результатів, а також чимала різноманітність варіантів поняття кластера. Така нестатистична інтерпретація дає можливість аналітикові отримати результати кластеризації, які задовольняють його, що при використанні інших методів часто буває скрутним.

Розглянемо ієрархічні і неієрархічні методи більш детальніше.

Ієрархічні методи кластерного аналізу. При ієрархічній кластеризації виконується послідовне об'єднання менших кластерів у великі або розділення великих кластерів на менші.

Агломеративні методи AGNES (Agglomerative Nesting). Ця група методів характеризується послідовним об'єднанням елементів і відповідним зменшенням числа кластерів. На початку роботи алгоритму всі об'єкти є

окремими кластерами. На першому кроці найбільш схожі об'єкти об'єднуються в кластер. На подальших кроках об'єднання продовжується до тих пір, поки всі об'єкти не складатимуть один кластер.

Single Link, Complete Link, Group Average. Одні із перших алгоритмів кластеризації даних. Особливістю цих методів, є те, що вони розбивають об'єкти на кластери шляхом розбиття їх на ієрархічні групи. Основна суть цих методів полягає у виконанні наступних кроків:

- обчислення значень близькості між елементами і здобуття матриці близькості;
- визначення кожного елементу в свій окремий кластер;
- злиття в один кластер найбільш близьких пар елементів;
- оновлення матриці близькості шляхом видалення колонок і рядків для кластерів, які злилися з іншими і подальшого перерахунку матриці;
- перехід на крок 3 до тих пір, поки не спрацює зупинний критерій.

Поняттям, протилежним до поняття відстані між об'єктами та , є поняття близькості (схожість) між і . Точніше, міра близькості між об'єктами та - це дійсна функція з властивостями:

, якщо ,

, .

Пари значень мір близькості можна об'єднати в матрицю близькості

, для .

Величину називають коефіцієнтом близькості. Прикладом лінійною близькості є коефіцієнт кореляції.

Вказані алгоритми відрізняються між собою в 4-му кроці. І саме завдяки способам оновлення матриці близькості різні алгоритми мають різну точність. Перевірка точності алгоритмів була проведена на спеціальних тестових наборах і показала, що алгоритм Single Link має найменшу точність, а останні два - приблизно однакову, але більш високу, чим Single Link. В якості зупинного критерію вибирається максимальна кількість

об’єктів в кластері (рис. 7.30). Швидкість роботи алгоритмів Single Link і Group Average - , а Complete Link - , де - кількість елементів.

Перевагами методів є те, що алгоритми не потребують навчання та використовують матрицю близькості між елементами. Недоліками цих методів є: необхідно задавати поріг - максимальну кількість елементів в кластері; для здобуття добрих результатів кластеризації значення близькості між парами елементів повинні приходити в певному порядку, тобто робота алгоритму не детермінована; кластери не перетинаються.

Рис. 7.30. Результати роботи алгоритму Complete Link.

Алгоритм CURE (Clustering Using REpresentatives). Виконує ієрархічну кластеризацію з використанням набору визначальних точок для визначення об'єкту в кластер. Призначення: кластеризація дуже великих наборів числових даних. Обмеження: ефективний для даних низької розмірності, працює лише на числових даних. Достоїнства: виконує кластеризацію на високому рівні навіть за наявності викидів, виділяє кластери складної форми і різних розмірів, володіє лінійно залежними вимогами до місця зберігання даних і часову складність для даних високої розмірності. Недоліки: є необхідність в задані порогових значень і кількості кластерів. Робота алгоритму складається з наступних кроків:

Крок 1: Побудова дерева кластерів, яке складається з кожного рядка вхідного набору даних.

Крок 2: Формування «купи» в оперативній пам'яті, розрахунок відстані до найближчого кластера (рядка даних) для кожного кластера. При формуванні «купи» кластери сортуються за збільшенням дистанції від кластера до найближчого кластера. Відстань між кластерами визначається по двох найближчих елементах з сусідніх кластерів. Для визначення відстані між кластерами використовуються «манхэттенська», «Евклідова» метрики або схожі на них функції.

Крок 3: Злиття ближніх кластерів в один кластер. Новий кластер отримує всі точки вхідних даних, які опиняються в ньому. Виконується розрахунок відстані до інших кластерів для новоутвореного кластера. Для розрахунку відстані кластери діляться на дві групи: перша група – кластери, в яких найближчими кластерами вважаються кластери, що входять в новоутворений кластер, останні кластери – друга група. При цьому для кластерів з першої групи, якщо відстань до новоутвореного кластера менше ніж до попереднього найближчого кластера, то найближчий кластер міняється на новоутворений кластер. Інакше шукається новий найближчий кластер, але при цьому не беруться кластери, відстані до яких більше, ніж до новоутвореного кластера. Для кластерів другої групи виконується наступне: якщо відстань до новоутвореного кластера ближча, ніж попередній найближчий кластер, то найближчий кластер міняється. Інакше нічого не відбувається.

Крок 4: Перехід на крок 3, якщо не отримана необхідна кількість кластерів.

Дивізімні методи DIANA (Divisive Analysis). Ці методи є логічною протилежністю агломеративним методам. На початку роботи алгоритму всі об'єкти належать одному кластеру, який на подальших кроках ділиться на менші кластери, в результаті утворюється послідовність розщеплюючих груп.

Алгоритм BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies). У цьому алгоритмі передбачений двох етапний процес кластеризації. Призначення: кластеризація дуже великих наборів числових даних. Обмеження: робота з лише числовими даними. Достоїнства: двоступінчата кластеризація, кластеризація великих об'ємів даних, працює на обмеженому об'ємі пам'яті, є локальним алгоритмом, може працювати при одному скануванні вхідного набору даних, використовує той факт, що дані неоднаково розподілені по простору, і обробляє області з великою щільністю як єдиний кластер. Недоліки: робота з лише числовими даними,

добре виділяє лише кластери сферичної форми, є необхідність в задані порогових значень. Робота алгоритму здійснюється таким чином:

Фаза 1: Завантаження даних в пам'ять. Побудова початкового кластерного дерева за даними в пам'яті. Кластерне дерево – це зважено збалансоване дерево з двома параметрами: – коефіцієнт розгалуження, – порогова величина. Кожен не листовий вузол дерева має не більше ніж входжень вузлів (рис. 7.31).

Рис. 7.31. Побудова кластерного дерева.

Кожен листовий вузол має посилання на два сусідні вузли. Кластер, що складається з елементів листового вузла, повинен задовольняти наступній умові: діаметр або радіус отриманого кластера має бути не більше порогової величини .

Фаза 2: Стискування (ущільнення) даних. Стискування даних до прийнятних розмірів за допомогою перестроювання і зменшення кластерного дерева із збільшенням порогової величини .

Фаза 3: Глобальна кластеризація. Застосовується вибраний алгоритм кластеризації на листових компонентах кластерного дерева.

Фаза 4: Поліпшення кластерів. Використовує центри тяжесті кластерів, отримані у фазі 3, як основи. Перерозподіляє дані між «близькими» кластерами. Дана фаза гарантує попадання однакових даних в один кластер.

Алгоритм MST (Algorithm based on Minimum Spanning Trees). Призначення: кластеризація великих наборів довільних даних. Достоїнства: виділяє кластери довільної форми, вибирає з декількох оптимальних рішень найоптимальніше. Опис алгоритму:

Крок 1: Побудова мінімального остовного дерева.

Алгоритм Борувки:

1. Для кожної вершини графа знаходимо ребро з мінімальною вагою.

2. Додаємо знайдені ребра до остовного дерева, за умови їх безпеки.
3. Знаходимо і додаємо безпечні ребра для незв'язаних вершин до остовного дерева.

Алгоритм Крускала:

1. Обхід ребер за збільшенням їх ваги.
2. За умови безпеки ребра додаємо його до остовного дерева.

Алгоритм Пріма:

1. Вибір кореневої вершини.
2. Починаючи з кореня додаємо безпечні ребра до остовного дерева.

Крок 2: Розділення на кластери. Дуги з найбільшими вагами розділяють кластери.

Алгоритм Форель. Алгоритм є прикладом евристичного дивізімного алгоритму класифікації. В основі роботи алгоритму Форель лежить використання гіпотези компактності: близьким в змістовному сенсі об'єктам в геометричному просторі ознак відповідають відособлена множина точок, так звані «згустки». Мета роботи алгоритму Форель полягає в тому, щоб знайти таке розбиття множини об'єктів, аби величина відстані між ними була мінімальною.

Робота алгоритму полягає в переміщенні гіперсфери певного радіусу в геометричному просторі до здобуття стійкого центру тяжіння спостережень, що попали в цю гіперсферу. До початку роботи алгоритму ознаки об'єктів нормуються так, щоб їх значення знаходилися між нулем і одиницею. Процедура алгоритму Форель є такою, що сходиться за кінцеве число кроків в просторі Евкліда будь-якої розмірності при довільному розташуванні точок і будь-якому виборі гіперсфери (рис. 7.32).

Рис. 7.32. Пошук кластерів алгоритмом Форель.

Алгоритм Форель 2 є модифікацією алгоритму Форель і застосовується в тих випадках, коли необхідно отримати спочатку задану кількість кластерів.

Радіус сфери у міру потреби може змінюватися на задану величину, яка від ітерації до ітерації зменшуватиметься.

Принцип роботи описаних вище груп методів у вигляді дендрограмми показаний на рис. 7.33.

Ієрархічні методи кластеризації розрізняються правилами побудови кластерів. В якості правила виступають критерії, які використовуються при рішенні питання про «схожість» об'єктів при їх об'єднанні в групу (агломеративні методи) або розділення на групи (дивізімні методи). Ці методи кластерного аналізу застосовуються при невеликих об'ємах наборів даних..

Рис. 7.33. Дендрограмма агломеративних і дивізімних методів.

Неієрархічні методи кластерного аналізу. При великій кількості спостережень ієрархічні методи кластерного аналізу не придатні. У таких випадках використовують неієрархічні методи, засновані на розділенні, які є ітеративними методами дроблення вхідної сукупності. В процесі ділення нові кластери формуються до тих пір, поки не буде виконано правило зупинки. Така неієрархічна кластеризація полягає в розділенні набору даних на певну кількість окремих кластерів. Існує два підходи. Перший полягає у визначенні кордонів кластерів як найбільш щільних ділянок в багатовимірному просторі даних, тобто визначення кластера там, де є велике «згущення точок». Другий підхід полягає в мінімізації міри відмінності об'єктів.

Ітераційні алгоритми. Такі алгоритми засновані на оптимізації деякої цільової функції, що визначає оптимальне в певному значенні розбиття множини об'єктів на кластери. Вони носять ітеративний характер, і на кожній ітерації потрібно розраховувати матрицю відстаней між об'єктами. При

великому числі об'єктів це неефективно і вимагає серйозних обчислювальних ресурсів.

Алгоритм k-середніх (k-means). Найбільш поширений серед неієрархічних методів алгоритм -середніх, також названий швидким кластерним аналізом. На відміну від ієрархічних методів, які не вимагають попередніх припущень відносно числа кластерів, для можливості використання цього методу необхідно мати гіпотезу про найбільш вірогідну кількість кластерів. Алгоритм -середніх будує кластерів, розташованих на можливо великих відстанях один від одного. Основний тип задач, які вирішує алгоритм -середніх, - наявність припущень (гіпотез) відносно числа кластерів, при цьому вони мають бути різні настільки, наскільки це можливо. Вибір числа може базуватися на результатах попередніх досліджень, теоретичних міркуваннях або інтуїції. Загальна ідея алгоритму полягає в наступному: задане фіксоване число кластерів спостереження зіставляються кластерам так, що середні в кластері (для всіх змінних) максимально можливо відрізняються один від одного. Робота алгоритму полягає в наступному (рис. 7.34):

1. Первинний розподіл об'єктів по кластерах. Вибирається число , і на першому кроці ці точки вважаються «центрами» кластерів. Кожному кластеру відповідає один центр.
2. Ітеративний процес. Обчислюються центри кластерів, якими потім і далі вважаються покоординатні середні кластерів. Об'єкти знову перерозподіляються. Процес обчислення центрів і перерозподілу об'єктів продовжується до тих пір, поки не виконана одна з умов:

- кластерні центри стабілізувалися, тобто всі спостереження належать кластеру, якому належали до поточної ітерації;
- число ітерацій дорівнює максимальному числу ітерацій.

Рис. 7.34. Результат кластеризації алгоритмом -means.

Після здобуття результатів кластерного аналізу методом -середніх слід перевірити правильність кластеризації (тобто оцінити, наскільки кластери відрізняються один від одного). Для цього розраховуються середні значення для кожного кластера. При хорошій кластеризації мають бути отримані середні, що сильно відрізняються, для всіх вимірів або хоч би більшої їх частини.

Перевагами алгоритму -середніх є: простота та швидкість використання, зрозумілість і прозорість алгоритму. Недоліки алгоритму -середніх:

- алгоритм дуже чутливий до викидів, які можуть спотворювати середнє. Можливим вирішенням цієї проблеми є використання модифікації алгоритму - алгоритм -медіани;
- алгоритм може повільно працювати на великих базах даних. Можливим вирішенням даної проблеми є використання вибірки даних.

З метою підвищення ефективності роботи алгоритму розроблені цікаві його розширення для роботи з категорійними атрибутами (-modes) і змішаними атрибутами (-prototypes). Наприклад, в к-prototypes розрахунок відстаней між об'єктами здійснюється по-різному залежно від типа атрибуту.

Алгоритм PAM (partitioning around Medoids). PAM є модифікацією алгоритму -середніх алгоритмом -медіани (-medoids).Алгоритм менш чутливий до шумів і викидів даних, чим алгоритм -means, оскільки медіана менше схильна до впливів викидів. PAM ефективний для невеликих баз даних, але його не слід використовувати для великих наборів даних.

Алгоритм EM (Expectation - Maximization). В основі ідеї EM алгоритму лежить припущення, що досліджувана множина даних може бути змодельована за допомогою лінійної комбінації багатовимірних нормальних розподілів, а метою є оцінка параметрів розподілів, які максимізували логарифмічну функцію правдоподібності, використовувану як міра якості моделі. Іншими словами, передбачається, що дані в кожному кластері підкоряються певному закону розподілу, а саме, нормальному розподілу. З врахуванням цього припущення можна визначити параметри - математичне сподівання і

дисперсію, які відповідають закону розподілу елементів в кластері, щонайкраще «відповідному» до спостережуваних даних. Таким чином, ми передбачаємо, що будь-яке спостереження належить до всіх кластерів, але з різною ймовірністю. Тоді задача полягатиме в «підгонці» розподілів суміші до даних, а потім у визначенні ймовірності приналежності спостереження до кожного кластера. Вочевидь, що спостереження має бути віднесене до того кластера, для якого дана вірогідність вища.

Алгоритм EM заснований на обчисленні відстаней. Він може розглядатися як узагальнення кластеризації на основі аналізу суміші імовірнісних розподілів. В процесі роботи алгоритму відбувається ітеративне поліпшення рішення, а зупинка здійснюється в мить, коли досягається необхідний рівень точності моделі. Мірою в даному випадку є статистична величина, що монотонно збільшується, названа логарифмічною правдоподібністю. Метою алгоритму є оцінка середніх значень, коваріацій і вагів суміші для функції розподілу ймовірностей (рис. 7.35).

Рис. 7.35. Результати кластеризації алгоритмом EM.

Серед переваг EM алгоритму можна виділити наступні: потужна статистична основа, лінійне збільшення складності при зростанні об'єму даних, стійкість до шумів і пропусків в даних, можливість побудови бажаного числа кластерів. Проте алгоритм має і ряд недоліків. По-перше, припущення про нормальність всіх вимірів даних не завжди виконується. По-друге, при невдалій ініціалізації збіжність алгоритму може виявитися повільною. Окрім цього, алгоритм може зупинитися в локальному мінімумі і дати квазіоптимальне рішення.

Кластеризація категорійних даних: алгоритм CLOPE. Категорійні дані зустрічаються в будь-яких областях: виробництво, комерція, маркетинг, медицина. Вони включають і так звані транзакційні дані: чеки в супермаркетах, логи відвідин веб-ресурсів. Взагалі під категорійними

даними розуміють якісні характеристики об'єктів, виміряні в шкалі найменувань.

Застосовувати для кластеризації об'єктів з категорійними ознаками традиційні алгоритми неефективно, а часто – неможливо. Основні труднощі пов'язані з високою розмірністю і гігантським об'ємом, якими часто характеризуються такі бази даних. На сьогоднішній день запропоновано понад десяток методів для роботи з категорійними даними. Одним з ефективних вважається алгоритм LargeItem, який заснований на оптимізації деякого глобального критерію. Цей глобальний критерій використовує параметр підтримки. Взагалі, обчислення глобального критерію робить алгоритм кластеризації у багато разів швидше, ніж при використанні локального критерію при парному порівнянні об'єктів, тому «глобалізація» оціночної функції – один з шляхів здобуття масштабованих алгоритмів.

Алгоритм CLOPE запропонований в 2002 році групою китайських вчених. При цьому він забезпечує вищу продуктивність і кращу якість кластеризації порівняно з алгоритмом LargeItem і багатьма ієрархічними алгоритмами. Призначення: кластеризація величезних наборів категорійних даних. Переваги: висока масштабованість і швидкість роботи, якість кластеризації, що досягається використанням глобального критерію оптимізації на основі максимізації градієнта висоти гістограми кластера. Під час роботи алгоритм зберігає в пам'яті невелику кількість інформації по кожному кластеру і вимагає мінімальне число сканувань набору даних. CLOPE автоматично підбирає кількість кластерів, причому це регулюється одним єдиним параметром – коефіцієнтом відштовхування.

Алгоритм функціонує на основі слідуючих процедур. Хай є база транзакцій , що складається з множини транзакцій . Кожна транзакція є набір об'єктів . Множина кластерів є розбиття множини , таке, що = і , , . Кожен елемент називається кластером, а – кількість транзакцій, кількість

об'єктів в базі транзакцій і число кластерів відповідно. Кожен кластер має наступні характеристики:

- – множина унікальних об'єктів;
- – кількість входжень (частота) об'єкту в кластер ;

,

,

- функція вартості

,

де – кількість об'єктів в кластері, – кількість кластерів, – коефіцієнт відштовхування. За допомогою параметра регулюється рівень схожості транзакцій усередині кластера, і, як наслідок, фінальна кількість кластерів. Цей коефіцієнт підбирається користувачем. Чим більше , тим нижче рівень схожості і тим більше кластерів згенерується.

Формальна постановка задачі кластеризації алгоритмом CLOPE виглядає таким чином: для заданих та знайти розбиття : .

Нові алгоритми кластерного аналізу. Методи, які ми розглянули, є «класикою» кластерного аналізу. До останнього часу основним критерієм, по якому оцінювався алгоритм кластеризації, була якість кластеризації: вважалося, аби весь набір даних уміщався в оперативній пам'яті. Проте зараз, у зв'язку з появою надвеликих баз даних, з'явилися нові вимоги, яким повинен задовольняти алгоритм кластеризації. Основне з них - це масштабованість алгоритму. Відзначимо також інші властивості, яким повинен задовольняти алгоритм кластеризації: незалежність результатів від порядку вхідних даних; незалежність параметрів алгоритму від вхідних даних. Останнім часом ведуться активні розробки нових алгоритмів кластеризації, здатних обробляти надвеликі бази даних. У них основна увага приділяється масштабованості. До таких алгоритмів відноситься узагальнене представлення кластерів (summarized cluster representation), а також вибірка і використання структур даних, підтримуваних нижче лежачих СУБД.

Алгоритми нечіткої кластеризації - алгоритм Fuzzy C-means. Призначення: кластеризація великих наборів числових даних. Достоїнства: нечіткість при визначенні об'єкту в кластер дозволяє визначати об'єкти, які знаходяться на кордоні, в кластери. Недоліки: обчислювальна складність, задання кількості кластерів, виникає невизначеність з об'єктами, які віддалені від центрів всіх кластерів.

Принцип роботи алгоритму полягає в наступному. Хай нечіткі кластери задаються матрицею розбиття , , , , де - міра приналежності об'єкту до кластера , – кількість кластерів, – кількість елементів. При цьому

, , , .

Етап 1. Встановити параметри алгоритму: - кількість кластерів; - експоненціальна вага, що визначає нечіткість, розмазаність кластерів ( ), - параметр зупинки алгоритму.

Етап 2. Генерація випадковим чином матриці нечіткого розбиття з врахуванням вказаних умов.

Етап 3. Розрахунок центрів кластерів , .

Етап 4. Розрахунок відстані між об'єктами і центрами кластерів

, , .

Етап 5. Перерахунок елементів матриці розбиття з врахуванням наступних умов:

якщо : ,

якщо : , .

Етап 6. Перевірити умову , де - матриця нечіткого розбиття на попередній ітерації алгоритму. Якщо «Так», то перехід до етапу 7, інакше до етапу 3.

Етап 7. Кінець роботи алгоритму.

Алгоритм WaveCluster. WaveCluster є алгоритмом кластеризації на основі хвилевих перетворень. На початку роботи алгоритму дані узагальнюються шляхом накладання на простір даних багатовимірних грат. На подальших кроках алгоритму аналізуються не окремі точки, а узагальнені характеристики точок, що попали в одну чарунку грат. В результаті такого

узагальнення необхідна інформація уміщається в оперативній пам'яті. На подальших кроках для визначення кластерів алгоритм застосовує хвилеве перетворення до узагальнених даних. Головні особливості WaveCluster: складність реалізації, алгоритм може виявляти кластери довільних форм, алгоритм не чутливий до шумів, алгоритм застосовний лише до даних низької розмірності.

Алгоритм CLARA (Clustering LARge Applications). Алгоритм CLARA був розроблений Kaufmann і Rousseeuw в 1990 році для кластеризації даних у великих базах даних. Даний алгоритм виконується в статистичних аналітичних пакетах, наприклад, таких як S+.

Викладемо коротко суть алгоритму. Алгоритм CLARA витягує множину зразків з бази даних. Кластеризація застосовується до кожного із зразків, на виході алгоритму пропонується краща кластеризація. Для великих баз даних цей алгоритм ефективніший, ніж алгоритм PAM. Ефективність алгоритму залежить від вибраного як зразок набору даних. Хороша кластеризація на вибраному наборі може не дати хорошу кластеризацію на всій множині даних.

Алгоритми Clarans, CURE, DBScan. Алгоритм Clarans (Clustering Large Applications based upon RANdomized Search) формулює задачу кластеризації як випадковий пошук в графі. В результаті роботи цього алгоритму сукупність вузлів графа є розбиттям множини даних на число кластерів, визначеним користувачем. «Якість» отриманих кластерів визначається за допомогою критеріальної функції. Алгоритм Clarans сортує все можливе розбиття множини даних у пошуках прийнятного рішення. Пошук рішення зупиняється в тому вузлі, де досягається мінімум серед зумовленого числа локальних мінімумів.

Серед нових масштабованих алгоритмів також можна відзначити алгоритм CURE - алгоритм ієрархічної кластеризації, і алгоритм DBScan, де поняття кластера формулюється з використанням концепції щільності (density).

Основним недоліком алгоритмів Clarans, CURE, DBScan є та обставина, що вони вимагають задання деяких порогів щільності точок, а це не завжди прийнятно. Ці обмеження обумовлені тим, що описані алгоритми орієнтовані на надвеликі бази даних і не можуть користуватися великими обчислювальними ресурсами.

Над масштабованими методами зараз активно працюють багато дослідників, основне завдання яких - здолати недоліки алгоритмів, що існують на сьогоднішній день.

7.4. Програмне забезпечення задач кластеризації

Одним з основних підходів в «виявленні знань в даних» (Data Mining) є кластеризація. Кластерний аналіз дозволяє відкрити в даних раніше невідомі закономірності, які практично неможливо досліджувати іншими способами і представити їх в зручній для користувача формі. Методи кластерного аналізу використовуються як самостійні інструменти досліджень, так і у складі інших засобів Data Mining. До теперішнього часу розроблена велика кількість програмних продуктів, що застосовуються до даних різного типа. Розглянемо основні з них [17, 25, 72, 76, 79].

Система PolyAnalyst. В програмному комплексі реалізовано два модулі, які відповідають за проведення кластерного аналізу: Find Dependecies (FD) - N-мірний аналіз розподілів та Find Clusters (FC) - N-мірний кластеризатор.

Алгоритм Find Dependecies виявляє у вхідній таблиці групи записів, для яких характерна наявність функціонального зв'язку між цільовою змінною і незалежними змінними, оцінює міру (силу) цієї залежності в термінах стандартної помилки, визначає набір найбільш впливових чинників, відсіває точки, що відскочили. Цільова змінна для FD має бути числового типа, тоді як незалежні змінні можуть бути і числовими і категоріями і логічними (рис. 7.36).

Рис. 7.36. Кластеризація в модулі Find Dependecies.

Алгоритм працює дуже швидко і здатний обробляти великі об'єми даних. Його можна використовувати як препроцесор для інших алгоритмів, оскільки він зменшує простір пошуку, а також як фільтр точок, що відскочили, або в зворотній постановці, як детектор виключень. FD створює правило табличного вигляду, проте як і всі правила PolyAnalyst воно може бути обчислене для будь-якого запису таблиці.

Модуль Find Clusters застосовується тоді, коли треба виділити в деякій множині даних компактні типові підгрупи (кластери), що складаються з близьких по своїх характеристиках записів. Причому заздалегідь може бути невідомо які змінні потрібно використовувати для такого розбиття. Алгоритм FC сам визначає набір змінних, для яких розбиття найзначиміше. Результатом роботи алгоритму є опис областей (діапазонів значень змінних), що характеризують кожен виявлений кластер і розбиття досліджуваної таблиці на підмножини, відповідні кластерам. Якщо дані є досить однорідними по всіх своїх змінних і не містять «згущувань» точок в якихось областях, цей метод не дасть результатів (рис. 7.37).

Рис. 7.37. Кластеризація в модуліFind Clusters.

Треба відзначити, що мінімальне число кластерів, що виявляються, рівне двом - згущування точок лише в одному місці в даному алгоритмі не розглядається як кластер. Крім того, цей метод пред'являє вимоги до наявності достатньої кількості записів в досліджуваній таблиці, а саме, мінімальна кількість записів в таблиці, в якій може бути виявлено кластерів, рівне (2 -1)4.

Система «Багатовимірні розвідувальні технології аналізу STATISTICA».

У модулі «Кластерний аналіз» реалізований повний набір методів кластерного аналізу даних, включаючи методи -середніх, ієрархічної кластеризації і двовходового об'єднання. Дані можуть поступати як у ісходному вигляді, так і у вигляді матриці відстаней між об'єктами. Спостереження і змінні

можна кластеризувати, використовуючи різні міри відстані (евклідову, квадрат евклідова, міських кварталів (манхэттенське), Чебишева, степенне, відсоток незгоди і коефіцієнта кореляції Пірсона), а також різні правила об'єднання кластерів. Матриці відстаней можна зберігати для подальшого аналізу в інших модулях системи STATISTICA (рис. 7.38).

Рис. 7.38. Кластерний аналіз в пакеті STATISTICA.

При проведенні кластерного аналізу методом -середніх користувач має повний контроль над початковим розташуванням центрів кластерів. Можуть бути виконані надзвичайно великі плани аналізу: так наприклад, при ієрархічному аналізі можна працювати з матрицею з 90 тис. відстаней. Окрім стандартних результатів кластерного аналізу, в модулі доступний також всілякий набір описових статистик і розширених діагностичних методів (повна схема об'єднання з пороговими рівнями при ієрархічній кластеризації, таблиця дисперсійного аналізу при кластеризації методом -середніх). Інформація про приналежність об'єктів до кластерів може бути додана до файлу даних і використовуватися в подальшому аналізі. Графічні можливості модуля «Кластерний аналіз» включають дендрограмми, двовходові діаграми об'єднань, графічне представлення схеми об'єднання, діаграму середніх при кластеризації по методу -середніх і багато що інше.

Розглянемо процедуру рішення практичної задачі методом кластерного аналізу в системі STATISTICA. Задачею кластерного аналізу є організація спостережуваних даних в наочні структури. Для вирішення даної задачі в кластерному аналізі скористаємося наступними методами - Joining (tree clustering - ієрархічні агломеративні методи або деревовидна кластеризація), K - means clustering (метод -середніх), Two-way joining (двохвходове об'єднання).

Для цього за допомогою перемикача модулів STATISTICA відкриємо модуль Cluster Analysis (Кластерний Аналіз). На екрані з'явиться стартова панель модуля Clustering Method (методи кластерного аналізу), яка дозволяє вибрати необхідний метод кластерного аналізу. Розглянемо кожен з цих методів.

Joining (tree clustering) (ієрархічні агломеративні методи). Відкриємо файл даних. Після вибору Joining (tree clustering) з'являється вікно Cluster Analysis: Joing (Tree Clustering) (вікно введення режимів роботи для ієрархічних агломеративних методів) (рис. 7.39), в якому опція Variables дозволяє вибрати змінні, що беруть участь в класифікації. Виберемо всі змінні Select All .

Рис.7.39. Cluster Analysis - Joing (Tree Clustering).

Також можна задати Input (тип вхідної інформації) і Cluster (режим класифікації (по ознаках або об'єктах)). Можна вказати Amalgamation rule (правило об'єднання) і Distance measure (метрика відстаней). Опція Codes for grouping variable (коди для груп змінної) вказуватимуть кількість аналізованих груп об'єктів, а опція Missing data (пропущені змінні) дозволяє вибрати або порядкове видалення змінних із списку, або замінити їх на середні значення. Після задання всіх необхідних параметрів будуть виконані обчислення, а на екрані з'явиться вікно, що містить результати кластерного аналізу «Joining Results» (рис.7.40).

Рис.7.40. Вікно результатів кластерного аналізу «Joining Results».

Користувач може викликати на екран горизонтальну і вертикальну діаграму (Horizontal hierachical plot або Vertical icicle plot) (рис. 7.41).

Рис. 7.41. Vertical icicle plot.

Аби повернутися у вікно, що містить інші результати кластерного аналізу, необхідно використовувати опцію Continue. Опція Discriptive statistics містить такі важливі описові статистики, як середнє (means) і середньоквадратичне відхилення (standart deviations) для кожного спостереження.

K - means clustering (метод середніх). Із стартової панелі модуля Clustering Method (методи кластерного аналізу) виберемо K - means clustering (метод -середніх). Відкриємо файл даних. У вікні Cluster Analysis: K - means clustering (рис. 7.42) опція Variables дозволяє вибрати змінні, що беруть участь в кластеризації. Виберемо всі змінні Select All . Опція Cluster вказує як ведеться кластеризація: при запуску встановлений режим Variables (colums) - кластеризуються змінні на підставі їх спостережень, проте в переважній більшості випадків використовується режим Cases (rows) – класифікуються спостереження.

Рис.7.42. Cluster Analysis: K - means clustering.

Опція Number of iterations вказує кількість ітерацій в розрахунках кластерів. Як правило, встановлених за умовчанням 10 ітерацій цілком достатньо. Опція Missing data встановлює режим роботи з тими спостереженнями, в яких пропущені дані. Якщо встановити режим Subsituted by means (Замінювати на середнє), то замість пропущеного числа буде використано середнє по цій змінній (або спостереженню). Після проведення обчислень з'явиться нове вікно: «K - Means Clustering Results» (рис. 7. 43).

Рис. 7.43. K - Means Clustering Results.

У нижній частині вікна розташовані опції для виведення різної інформації по кластерах.

1. Analysis of Variance (аналіз дисперсії). Після застосування опції з'являється таблиця, в якій приведена міжгрупова і внутрішньогрупова дисперсії, в якій рядки - змінні (спостереження), а стовпці - показники для кожної змінної: дисперсія між кластерами, число мір свободи для міжкласової дисперсії, дисперсія всередині кластерів, число мір свободи для внутрікласової дисперсії, F - критерій, для перевірки гіпотези про нерівність дисперсій. Перевірка даної гіпотези схожа на перевірку гіпотези в дисперсійному аналізі, коли робиться припущення про те, що рівні чинника не впливають на результат.
2. Cluster Means & Euclidean Distances (середні значення в кластерах і відстань Евкліда). Виводяться дві таблиці. У першій вказані середні величини класу по всіх змінних (спостереженням). По вертикалі вказані номери класів, а по горизонталі змінні (спостереження). У другій таблиці приведені відстані між класами, а по вертикалі і по горизонталі вказані номери кластерів. Таким чином при пересіченні рядків і стовпців вказані відстані між відповідними класами.
3. Graph of means є графічним зображенням (рис. 7.44) тієї інформації, що міститься в таблиці, яка виводиться при застосуванні опції Analysis of Variance. На графіці показані середні значення змінних для кожного кластера.

Рис. 7.44. Graph of means.

4. Descriptive Statistics for each cluster (описова статистика для кожного кластера). Після використання цієї опції виводяться вікна, кількість яких дорівнює кількості кластерів. У кожному такому вікні в рядках вказані змінні (спостереження), а по горизонталі їх характеристики, розраховані для даного класу: середнє, незміщене середньоквадратичне відхилення, незміщена дисперсія.

5. Members for each cluster & distances. Виводиться стільки вікон, скільки задано класів. У кожному вікні вказується загальне число елементів, віднесених до цього кластера, у верхньому рядку вказаний номер спостереження (змінної), віднесеного до даного класу і відстань Евкліда від центру класу до цього спостереження (змінної). Центр класу - середні величини по всіх змінних (спостереженням) для цього класу.

У системі STATISTICA реалізовані також і інші методи кластеризації, наприклад Two-way joining, в якому кластеризуються випадки і змінні одночасно. На рис. 7.45 показаний результат кластеризації. Трудність з інтерпретацією отриманих результатів цим методом виникає внаслідок того, що схожість між різними кластерами може походити з деякої відмінності підмножин змінних. Тому отримані кластери є за своєю природою неоднорідними.

Рис. 7.45. Результат кластеризації Two-way joining методом.

Пакет програм SPSS (Statistical Package for Social Science). Пакет програм є одним з поширених, потужних і зручних інструментів статистичного аналізу. Пакет SPSS користується популярністю у економістів, соціологів, маркетологів, надає користувачеві широкі можливості по статистичній обробці емпіричних даних, по формуванню і модифікації баз даних, а також по створенню звітів, надаючи широкі можливості по представленню результатів статистичної обробки в текстовій, табличній і графічній формах. Пакет орієнтований, головним чином, на аналіз просторових даних і на кластерний аналіз. Інтерфейс програми інтуїтивно зрозумілий користувачеві і дозволяє застосувати різні варіанти статистичного аналізу.

Опишемо практичний підхід до виділення сегментів споживачів методом ієрархічного кластерного аналізу. Хай досліджується поведінка споживачів пельменів. В ході опиту респондентам задавалася ціла низка запитань, призначених для вирішення задач, що стоять перед дослідженням. Також

з'ясовувалися соціально-демографічні параметри респондентів: стать, вік і рівень доходів. В якості критерію сегментації виділимо 4 змінні: частота покупки, кратність покупки, наявність марочних переваг і орієнтація на ціну/якість. Як дескриптори сегментів використовуватимемо стать, вік і рівень доходів.

Ієрархічний кластерний аналіз проводиться в два етапи. Єдиним результатом першого етапу повинно стати число кластерів (цільових сегментів), на які слід розділити досліджувану вибірку респондентів. На другому етапі виконується власне кластеризація респондентів по тому числу кластерів, яке було визначено в ході першого етапу аналізу.

Процедура кластерного аналізу в SPSS запускається за допомогою меню: Analyze Classify Hierarchical Cluster (Аналіз Класифікація Ієрархічний кластерний аналіз). У діалоговому вікні (рис. 7.46) необхідно з лівого списку всіх наявних у файлі даних змінних вибрати змінні, що є критеріями сегментації.

Рис. 7.46. Діалогове вікно «Hierarchical Cluster Analysis».

За умовчанням SPSS виводить також спеціальну перевернуту гістограму, названу «Icicle» (сосульковидна діаграма). За задумом творців програми, вона допомагає визначити оптимальну кількість кластерів (виведення спеціальних видів діаграм здійснюється за допомогою опції «Plots» (діаграми)). Окрім «Icicle» SPSS дозволяє вибрати швидку лінійчату діаграму «Dendogram» (дендограмма). Теоретично при невеликій (до 50—100) кількості респондентів дана діаграма допомагає вибрати оптимальне рішення відносно необхідного числа кластерів. Проте практично у всіх прикладах з реальних маркетингових досліджень розмір вибірки перевищує це значення. Дендограмма в даному випадку стає абсолютно даремною, оскільки навіть при відносно невеликому числі спостережень є

дуже довгою послідовністю номерів рядків вихідного файлу даних, сполучених між собою горизонтальними і вертикальними лініями.

Після вказівки критеріїв сегментації слід вибрати метод проведення кластерного аналізу. Це дозволяє зробити спеціальне діалогове вікно Hierarchical Cluster Analysis: Method, що викликається опцією «Method». Серед всіх можливих варіантів статистичних методик, пропонованих SPSS, рекомендується вибирати або встановлений за умовчанням метод «Between-groups linkage» (зв'язок між групами), або процедуру Ward'а («Ward's method»). При цьому перший метод використовується найчастіше зважаючи на його універсальність і відносну простоту статистичної процедури, на якій він заснований. При цьому методі відстань між кластерами обчислюється як середнє значення відстаней між всіма можливими парами спостережень, причому в кожній ітерації бере участь одне спостереження з одного кластера, а інше - з іншого. Метод Ward'а складається з множини етапів і заснований на усереднюванні значень всіх змінних для кожного спостереження і подальшому підсумовуванні квадратів відстаней від обчислених середніх до кожного спостереження. Також слід вибрати метод для обчислення відстаней між спостереженнями (область «Measure» (шкала) в даному діалоговому вікні). Найбільш часто використовуваним методом визначення відстаней для інтервальних змінних є квадрат відстані («Squared Euclidean Distance») Евкліда, що встановлюється за умовчанням. Саме даний метод найкраще зарекомендував себе в маркетингових дослідженнях як найбільш точний і універсальний.

Після задання всіх необхідних параметрів виконуємо розрахунок першого етапу кластерного аналізу, результати якого з'являться у вікні SPSS Viewer (звіт SPSS). Єдиним значимим підсумком першого етапу аналізу буде таблиця «Average Linkage (Between Groups)» (усереднені зв'язки між групами), представлена на рис 7.47. По ній і повинне визначитися оптимальне число кластерів.

Рис. 7.47. Таблиця «Average Linkage (Between Groups)».

Перш за все спробуємо застосувати найбільш поширений, стандартний метод для визначення числа кластерів. Спочатку по таблиці «Average Linkage (Between Groups)» визначимо, на якому кроці процесу формування кластерів (колонка «Stage») відбувається перший порівняно великий стрибок коефіцієнта агломерації (колонка «Coefficients»). Даний стрибок означає, що до нього в кластери об'єднувалися спостереження, що знаходяться на досить малих відстанях одне від одного, а з цього етапу починає відбуватися об'єднання більш далеких спостережень. У нашому випадку коефіцієнти плавно зростають від 0 до 1,056, тобто різниця між коефіцієнтами на кроках з першого по 286 включно була вельми мала (наприклад, між 286 і 285 кроками — всього 0,033). Проте починаючи з 287 кроку відбувається перший істотний стрибок коефіцієнта: з 1,056 до 3,690 (на 2,634). Таким чином, ми визначили крок, на якому відбувається перший стрибок коефіцієнта - 287. Тепер, аби визначити оптимальну кількість кластерів, необхідно відняти отримане значення із загального числа спостережень (розміру вибірки). Загальний розмір вибірки в нашому випадку складає 300 споживачів пельменів, отже, розрахункова оптимальна кількість кластерів складає: 300 – 287 = 13. Отримано досить велике число кластерів, яке надалі складно інтерпретуватиме. Тому необхідно досліджувати отримані кластери і визначити, які з них є значимими, а які слід спробувати скоротити. Ця задача вирішується на другому етапі кластерного аналізу.

Знов відкриємо головне діалогове вікно процедури кластерного аналізу (меню: Analyze Classify Hierarchical Cluster). У полі для аналізованих змінних вже є необхідні нам чотири параметри. Діалогове вікно, що відкрилося, дозволяє створити у вхідному файлі даних нову змінну, що розподіляє всіх респондентів на цільові групи. Виберемо параметр «Single Solution» (єдине

рішення) і вкажемо у відповідному полі необхідне нам число кластерів - 13. Тепер слід знов запустити процедуру кластерного аналізу. В результаті у вхідному файлі даних SPSS буде створена нова змінна з назвою «clu13_1».

Аби дослідити, наскільки вірно ми визначили оптимальне число кластерів, побудуємо лінійний розподіл змінної «clu13_1» (меню: Analyze Descriptive Statistics Frequencies (Аналіз Описова статистика Лінійні розподіли)). Як видно з рис 7.48, в кластерах з 7 по 13 число спостережень вагається від 1 до 2. Подібна ситуація зустрічається практично завжди, тому число кластерів, визначене на першому етапі аналізу, майже ніколи не буває істинно оптимальним. Тому разом з вищеописаним універсальним методом визначення оптимальної кількості кластерів існує також додаткове обмеження: розмір кластерів має бути статистично значимим і практично прийнятним. Наприклад, при нашому розмірі вибірки таке критичне значення можна встановити хоч би на рівні 10 респондентів на один кластер. Оскільки даній умові відповідають 6 кластерів, нам необхідно перерахувати процедуру кластерного аналізу із збереженням 6-кластерного рішення (буде створена нова змінна «clu6_1»).

Рис. 7.48. Лінійний розподіл для 13-кластерного рішення.

Побудувавши лінійний розподіл по знов створеній змінній «clu6_1», ми побачимо, що лише в трьох кластерах число респондентів більше 10. Отже, нам необхідно знов перебудувати кластерну модель, тепер для 3-кластерного рішення. Після цього знову побудуємо розподіл по змінній «clu3_1». У загальному випадку дану процедуру слід продовжувати до тих пір, поки не вийде рішення, в якому на кожен кластер доводитиметься статистично значиме число респондентів. У нашому випадку 3-кластерне рішення виявилося оптимальним.

Приступимо до завершуючого етапу кластерного аналізу: інтерпретації отриманих цільових груп (сегментів). Опис отриманих сегментів проводиться в два етапи: опис з точки зору критеріїв сегментації і опис з точки зору дескрипторів сегментів. Сегменти, виділені в результаті кластерного аналізу, характеризуються однорідністю значень критеріїв сегментації усередині кожного кластера і відмінністю між кластерами. Тому, по-перше, слід визначити, якими конкретно значеннями змінних, вибраних як критерії сегментації, характеризуються отримані кластери. Для цього найчастіше будують перехресний розподіл, в якому по стовпцях розташовується кластеризуюча змінна (у нашому випадку це «clu3_1»), а по рядках - критерії сегментації. Таким чином, можна бачити, в який кластер потрапляють респонденти з тим або іншим значенням критерію сегментації. Наприклад, в нашому випадку ми отримали 3 сегменти, які (за допомогою перехресного розподілу) описуються таким чином.

Сегмент 1. Часті покупці пельменів (1 раз на тиждень і частіше), які за один прихід в магазин купують невелику кількість продукту (до 1 кг); при цьому вони не звертає уваги на марку і орієнтуються в основному на ціну.

Сегмент 2. Відносно рідкі покупці пельменів (рідше за 1 раз в тиждень), які за один прихід в магазин купують значну кількість продукту (більше 1 кг); при цьому вони не звертає уваги на марку і орієнтуються в основному на ціну.

Сегмент 3. Відносно рідкі покупці пельменів (рідше за 1 раз в тиждень), які за один прихід в магазин купують значну кількість продукту (більше 1 кг); при цьому вони звертають увагу на марку і орієнтуються в основному на якість.

Отже, після побудови перехресного розподілу стає очевидною різниця в ключових характеристиках сегментів. Виділеним сегментам стає можливим дати вербальні назви. Крім того, з процентного співвідношення отриманих сегментів можна оцінити частку ринку, що займає кожен з них, і виявити найпривабливіші цільові групи. Другим, завершальним етапом в

інтерпретації результатів кластерного аналізу є поглиблений опис отриманих сегментів за допомогою дескрипторних змінних. Таким чином, сегменти все більше знаходять «людське обличчя». Опис сегментів дескрипторними змінними також проводиться за допомогою побудови перехресних розподілів способом, аналогічним описаному вище. В результаті виходить повна картина сегментації ринку. З такими даними можна аргументовано вибирати найпривабливіші цільові сегменти і розробляти стратегію позиціювання для кожного з них.

Кластеризація в програмному комплексі «1С: Предприятие 8.0». Метою кластеризації в програмному комплексі є виділення з множини об'єктів однієї природи деякої кількості відносно однорідних груп - сегментів або кластерів. Об'єкти розподіляються по групах так, щоб внутрішньогрупові відмінності були мінімальними, а міжгрупові - максимальними (рис. 7.49). Методи кластеризації дозволяють перейти від пооб’єктного до групового представлення сукупності довільних об'єктів, що істотно спрощує операцію ними.

Рис. 7.49. Аналіз даних методом кластеризації.

Можливі сценарії вживання кластеризації на практиці:

1. Сегментація клієнтів по певній сукупності параметрів дозволяє виділити серед них стійкі групи, що мають схожі купівельні переваги, рівні продажів і платоспроможності, що істотно спрощує управління взаєминами з клієнтами.
2. При класифікації товарів часто використовуються досить умовні принципи класифікації. Виділення сегментів на основі групи формальних критеріїв дозволяє визначити дійсно однорідні групи товарів. В умовах широкої і досить різнорідної номенклатури товарів управління асортиментом на рівні сегментів, в порівнянні з управлінням на рівні номенклатури, істотно

підвищує ефективність просування, ціноутворення, мерчендайзинга, управління ланцюжками постачань.

3. Сегментація менеджерів дозволяє ефективніше спланувати організаційні зміни, поліпшити мотиваційні схеми, скоректувати вимоги до найманого персоналу, що кінець кінцем дозволяє підвищити керованість компанії і стабільність бізнесу в цілому.

Результатами аналізу за допомогою кластеризації є:

- центри кластерів, що є сукупністю усереднених значень вхідних колонок в кожному кластері;
- таблиця міжкластерних відстаней (між центрами кластерів), що визначають міру відмінності між ними;
- значення прогнозних колонок для кожного кластера;
- рейтинг чинників і дерево умов, що визначили розподіл об'єктів на кластери.

Алгоритми кластеризації дозволяють не лише провести кластерний аналіз об'єктів на множині заданих атрибутів, але і спрогнозувати значення одного або декількох з них для актуальної вибірки на підставі віднесення об'єктів цієї вибірки до того або іншого кластера.

В програмному комплексі на основі застосування кластерного аналізу реалізован типовий бізнес - сценарій «Управління взаємовідношеннями з клієнтами».

Сценарій - "Планування рекламної кампанії". Планерування майбутньої рекламної кампанії розглядається з точки зору оптимізації розподілу виділеного бюджету по рекламних каналах виходячи з регіонального, продуктового, клієнтського і інших показників цільового сегменту, а також ефективності рекламних каналів у вказаних розрізах в деякому, попередньому плановому періоді.

Прогнозні атрибути - долі відгуків на рекламний канал умовно однорідних сегментів, виділених алгоритмом.

Обчислювані колонки: долі рекламних каналів в бюджеті рекламної кампанії з врахуванням вірогідної долі відгуків і ефективності (у сенсі результуючої виручки) кожного рекламного каналу.

На останок розглянемо найбільш суттєві практичні напрями застосування технології кластерного аналізу.

Кластеризація текстової інформації. Величезні об'єми інформації в мережі Internet приводять до того, що кількість об'єктів, що видаються по запиту користувача, дуже велика. Це утрудняє процес огляду результатів і вибору відповідних матеріалів з множини знайдених. Проте, в більшості випадків величезні об'єми інформації можна зробити доступними для сприйняття, якщо уміти розбивати джерела інформації (наприклад, WEB-сторінки) на тематичні групи. Тоді, користувач відразу може відкидати множину документів з мало релевантних груп. Такий процес угрупування даних здійснюється за допомогою кластеризації корпусу текстів. На даний момент існує декілька методів, що здійснюють кластеризацію документів:

- LSA/LSI – Latent Semantic Analysis/Indexing. Шляхом факторного аналізу множини документів виявляються латентні (приховані) чинники, які надалі є основою для утворення кластерів документів;
- STC – Suffix Tree Clustering. Кластери утворюються у вузлах спеціального вигляду дерева – суффіксного дерева, яке будується із слів і фраз вхідних документів;
- Single Link, Complete Link, Group Average – ці методи розбивають множину документів на кластери, розташовані в деревовидній структурі, – dendrogramm, яка отримується за допомогою ієрархічної агломеративної кластеризації;
- K-means. Кластери представлені у вигляді центроїдів, що є «центром маси» всіх документів, що входять в кластер;
- Scatter/Gather. Представляється як ітеративний процес, що спочатку розбиває (scatter) множину документів на групи і представленні потім цих

груп користувачеві (gather) для подальшого аналізу. Далі процес повторюється знову над конкретними групами.

На фазі розбиття метод може використовувати два алгоритми: Buckshot і Fractionation. Алгоритм Buckshot швидший і годиться для швидкої рекластеризації при виконанні ітерацій в Scatter/Gather. Fractionation же є точнішим і повільнішим алгоритмом і використовується в Scatter/Gather для попереднього розбиття на групи множини документів і виконується в режимі off-line. Обоє алгоритму відносяться до алгоритмів восходящої (bottom-up) кластеризації. Система Scatter/Gather існує у вигляді комерційного продукту і використовується для кластеризації різних текстових ресурсів (рис. 7.50).

Рис.7.50. Результат запиту, розподілений по кластерах, в системі Scatter/Gather.

Кластерний аналіз в маркетингових дослідженнях. Кластерний аналіз все частіше знаходить застосування в маркетингових дослідженнях. Серед найбільш популярних напрямів маркетингу виділяють такі:

1. Сегментація. Кластерний аналіз застосовується для вирішення широкого спектру задач, але найчастіше йдеться саме про задачу сегментації. Всі дослідження, присвячені проблемі сегментації, безвідносно того, який використовується метод, мають на меті ідентифікувати стійкі групи (люди, ринки, організації), кожна з яких об'єднує в собі об'єкти з схожими характеристиками.
2. Аналіз поведінки споживача. Другим, але не менш важливі напрямом використання апарату кластерного аналізу, є побудова однорідних груп споживачів з метою отримати максимально повне уявлення про те, як поводиться клієнт з кожного сегменту, які ознаки визначають його поведінку.
3. Позиціонування. Кластерний аналіз застосовується також для того, щоб визначити, в якій ніші краще позиціонувати продукт, що виводиться на

ринок. Кластерний аналіз дозволяє побудувати карту, на основі якої можна буде визначити рівень конкуренції в різних сегментах і характеристики, якими повинен володіти товар для того, щоб попасти в цільовий сегмент. Така карта дозволяє, наприклад, виявити нові ринки, для яких можна розробляти і просувати свої рішення.

4. Вибір тестових ринків. Багато маркетологів застосовують кластерний аналіз для того, щоб, визначити, які ринки (магазини, продукти, ...) можна об'єднати в одну групу за релевантними характеристиками. Річ у тому, що, висунувши припущення про існування певної закономірності необхідно запропонувати новий, не використаний в аналізі, ринок, на якому вона має бути перевірена, перш ніж застосовувати на практиці.

Кластеризація в фінансовій діяльності. Величезна кількість інвестиційних інструментів, що надається сучасним фінансовим ринком, заставляє корпоративних інвесторів з кожним днем аналізувати все більшу кількість фінансової інформації. Часом успіх інвестування залежить від об'єму аналізованих фінансових даних, часу, витраченого на аналіз, і вигляду, в якому представлені результати. Більше, швидше, зручніше - ось основні вимоги, що пред'являються постійно зміняющимся фінансовим ринком до методів аналізу фінансових даних. Основу для того, щоб упевнено відповідати на виклики ринку, дають методи кластеризації. Процедура кластеризації вирішує питання про схожість фінансових активів, що характеризуються значеннями багатьох параметрів, на основі формальних математичних критеріїв. Це дозволяє замінити тривалий і трудомісткий процес вивчення і порівняння активів швидшим обчислювальним алгоритмом. Крім того, будучи засобом аналізу багатовимірних даних, кластеризація дозволяє виділити активи з близькими значеннями всіх параметрів.

Використання кластеризації виправдане скрізь, де потрібний аналіз багатовимірних фінансових даних. Один з прикладів - вибір фінансових інструментів відповідно до багатофакторної моделі оцінки прибутковості

активів. Допустимо, що для формування диверсифікованого портфеля акцій потрібно проаналізувати чутливість 150 компаній NASDAQ до чотирьох чинників. Мірою чутливості до кожного чинника є показники багатофакторної моделі. Таким чином, кожна компанія характеризується набором з чотирьох значень . Виходить, що необхідно розглянути більше 600 чисел, а потім якимсь чином відранжирувати компанії. Метод звичайного сортування не дасть жодних результатів - якщо сортування виконується по показнику , що характеризує чутливість до першого чинника, то після виділення активів з близьким значенням може виявитися, що ці компанії істотно відрізняються по значеннях та .

Як видно з рим. 7.51, в результаті кластеризації все 150 компаній розподілено по трьох групах. У кожній групі автоматично зібрані компанії з близькими значеннями показників . Замість вивчення 600 значень тепер для здобуття повного уявлення про ситуацію, що склалася в даний момент на ринку, досить проаналізувати показники лише по трьох компаніях (центрам груп). Показники центру групи найбільш близькі до середніх по всій групі.

Рис. 7.51. Результати кластеризації компаній.

Аналіз стовпців в таблиці показує, що перша група найбільш чутлива до другого чинника, але значно менш чутлива до всіх інших. Друга - найбільш чутлива до всіх чинників, за винятком другого. По відношенню до нього спостерігається середня чутливість. Остання група об'єднує компанії, що демонструють середню чутливість до 1-го, 3-го і 4-го чинника і мінімальну, - до 2-го. Таким чином, якщо портфельний менеджер ставить перед собою завдання підняти очікувану прибутковість портфеля без збільшення ризику, пов'язаного з чинником , наприклад, цінами на нафту, і підвищеною чутливістю до чинника , наприклад, рівню довіри інвесторів,

то далі з одновимірного масиву першої групи легко вибираються компанії з необхідною нормою прибутковості.

Тест.

1. Під поняттям «класифікація» Ви розумієте:

а) системний розподіл предметів за якими-небудь істотними ознаками для зручності їх дослідження; розподіл вихідних понять і об'єднання їх в певному порядку;

б) системне об'єднання предметів за якими-небудь істотними ознаками для зручності їх дослідження; розподілу вихідних понять і розташування їх в певному порядку;

в) системний розподіл предметів за якими-небудь істотними ознаками для зручності їх дослідження; групування вихідних понять і розташування їх в певному порядку.

2. При проведенні класифікації мають бути присутні ознаки, що характеризують …

а) предметну область, до якої належить подія або об'єкт;

б) групу, до якої належить подія або об'єкт;

в) цільову функцію, яка характеризує події або об'єкт.

3.Задача класифікації, по Вашому розумінню, полягає в …

а) передбаченні категоріальної залежної змінної на основі вибірки безперервних і категоріальних змінних;

б) розрахунку категоріальної залежної змінної на основі вибірки неперервних і категоріальних змінних;

в) виявленні категоріальної залежної змінної на основі вибірки неперервних і категоріальних змінних.

4. Ви вважаєте, що метою процесу класифікації є …

а) побудова графічного зображення, яке використовує прогнозуючі атрибути як вхідні параметри і отримує значення залежного атрибуту;

б) побудова моделі, яка використовує прогнозуючі атрибути як вхідні параметри і отримує значення залежного атрибуту;

в) побудова алгоритму, який використовує прогнозуючі атрибути як вхідні параметри і отримує значення залежного атрибуту.

5. В чому, на Вашу думку, полягає процес класифікації:

а) в розбитті множини об'єктів на класи по різним критеріям;

б) в розбитті множини об'єктів на класи по певному критерію;

в) в автоматичному розбитті множини об'єктів на класи.

6. Часто в процесі класифікації застосовуються класифікатори, під якими розуміють:

а) деяку сутність, що визначає, якому із зумовлених класів належить об'єкт по зовнішньому вигляді;

б) деяку сутність, що визначає, якому із зумовлених класів належить об'єкт по вектору кількості об’єктів;

в) деяку сутність, що визначає, якому із зумовлених класів належить об'єкт по вектору ознак.

7. На які множини розбивають набір даних при класифікації?

а) навчальну і корисну;

б) функціональну і тестову;

в) навчальну і тестову;

г) функціональну і виконавчу.

8. Коли Ви конструюєте модель класифікації, то розумієте, що це …

а) опис множини зумовлених класів;

б) опис цільової функції;

в) опис множини обмежень класів.

9. Вибір методу класифікації Ви здійснюєте, керуючись наступними характеристиками:

а) швидкість, точність, інтерпретуємість, надійність;

б) швидкість, робастність, алгоритмічність, надійність;

в) швидкість, функціональність, інтерпретуємість, надійність;

г) швидкість, робастність, інтерпретуємість, стійкість;

д) швидкість, робастність, інтерпретуємість, надійність.

10. Процедуру крос-перевірки Ви застосовується для …

а) оцінки надійності класифікації на даних з тестової множини;

б) оцінки точності класифікації на даних з тестової множини;

в) оцінки швидкості класифікації на даних з тестової множини.

11. Відмінність задач кластеризації від задач класифікації полягає в тому, що …

а) класи вивчаємого набору даних, заздалегідь не зумовлені;

б) класи вивчаємого набору даних, заздалегідь зумовлені;

в) класи вивчаємого набору даних не відомі.

12. Технологія кластеризації призначена для …

а) розбиття сукупності об'єктів на різнорідні групи;

б) розбиття сукупності об'єктів на однорідні групи;

в) розбиття сукупності об'єктів на однакові групи.

13. На Ваш погляд, метою кластеризації є …

а) пошук однакових множин;

б) пошук функціональних залежностей;

в) пошук існуючих структур.

14. Кластер можна охарактеризувати як …

а) групу об'єктів, що не мають загальних властивостей;

б) групу об'єктів, що мають спеціальні властивості;

в) групу об'єктів, що мають загальні властивості.

15. Характеристиками кластера Ви вважаєте слідуючи ознаки:

а) внутрішня різнорідність та зовнішня ізольованість;

б) внутрішня функціональність та зовнішня ізольованість;

в) внутрішня однорідність та зовнішня ізольованість;

г) внутрішня однорідність та зовнішня не ізольованість.

16. Під поняттям «дендрограма» зазвичай розуміють:

а) візуальне представлення результатів таксономії у вигляді деревоподібної ієрархічної структури;

б) візуальне представлення результатів таксономії у вигляді графічної ієрархічної структури;

в) візуальне представлення результатів таксономії у вигляді багаторівневої ієрархічної структури.

17. Важливість вибору відстані між об'єктами полягає в тому, що від неї …

а) залежить остаточний варіант об’єднання об'єктів при даному алгоритмі розбиття;

б) залежить остаточний варіант розбиття об'єктів на класи при даному алгоритмі розбиття;

в) залежить остаточний варіант формального опису об'єктів в класах при будь-якому алгоритмі розбиття.

18. Застосовуючи поняття «об'єднання або зв'язки для двох кластерів», Ви вважаєте, що це є …

а) правила побудови кластеру;

б) правила розподілу об’єктів в кластерах;

в) правила визначення відстані між кластерами.

19. Під оцінкою якості кластеризації Ви розумієте …

а) можливість так розподілити кластери, аби значення вибраного функціонала якості прийняло найкраще значення;

б) можливість так приписати номери кластерів об'єктам, аби значення вибраного функціонала якості прийняло найкраще значення;

в) можливість так приписати номери кластерів об'єктам, аби значення вибраного функціонала якості прийняло значення нескінченності.

20. Вибір методу кластеризації Ви здійснюєте, керуючись наступними математичними характеристиками:

а) центр, радіус, середньоквадратичне відхилення, розмір кластера;

б) швидкість, радіус, середньоквадратичне відхилення, розмір кластера;

в) центр, радіус, стійкість, розмір кластера.

Розділ 8.

ЛАБОРАТОРНИЙ ПРАКТИКУМ

8.1. Лабораторна робота №1 «Створення і наповнення сховища даних».

Завдання до лабораторної роботи.

1. Відповідно до заданого варіанту підготувати необхідні дані у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.

2. Створити персональне сховище даних і організувати доступ до нього відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 1».

3. Виконати завантаження даних з таблиць MS Excel за допомогою Майстра імпорту відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 2».

4. Організувати завантаження даних за допомогою Майстра експорту в сховище даних відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 3».

5. Здійснити наступний імпорт даних з сховища: кількість відвантаженого товару в розрізі дат та товарів по вибраному Вами клієнтові, залишивши одну властивість товару (вибір властивості довільний). Виконання завдання здійснюється відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 4».

Варіанти завдань:

N варіанту	Галузь
1	Торгівля продовольчими товарами
2	Торгівля побутовими товарами
3	Торгівля речовими товарами

4	Торгівля побутовою технікою
5	Торгівля автомобілями
6	Торгівля нерухомістю
7	Будівництво
8	Реалізація медичних препаратів
9	Реалізація комп'ютерної техніки
10	Реалізація техніки зв'язку
11	Складський облік
12	Ремонт техніки
13	Ремонт житла

Теоретична частина.

В якості програмного засобу для проведення інтелектуального аналізу даних будемо використовувати пакет програм Deductor, який є технологічною платформою для створення закінчених аналітичних рішень. У ньому зосереджені найсучасніші методи витягання, маніпулювання, візуалізації даних, кластеризації, прогнозування і багато інших технологій інтелектуального аналізу даних. Deductor Academic призначений лише для освітніх цілей, безкоштовна версія пакету представлена за адресою http://www.basegroup.ru/download/deductor/deductor_setup_5.2/.

Пакет Deductor складається з 2-х частин - *багатовимірного сховища даних* Deductor Warehouse та *аналітичного додатка* Deductor Studio.

Deductor Warehouse - багатовимірне сховище даних, що акумулює всю необхідну для аналізу предметної області інформацію. Використання єдиного сховища дозволяє забезпечити несуперечність даних і централізоване зберігання, а також автоматично забезпечує всю необхідну підтримку процесу аналізу даних.

Deductor Studio - програма, що реалізовує функції імпорту, обробки, візуалізації і експорту даних. Deductor Studio може функціонувати і без сховища даних, отримуючи інформацію з будь-яких інших джерел, але найбільш оптимальним є їх спільне використання. У Deductor Studio включений повний набір механізмів, що дозволяє отримати інформацію з довільного

джерела даних, провести весь цикл обробки (очищення, трансформацію даних, побудову моделей), відображувати отримані результати найбільш зручним чином (OLAP, таблиці, діаграми, дерева і так далі) і експортувати результати в найбільш поширені формати.

Сховище даних Deductor Warehouse. При роботі зі сховищем даних від користувача не вимагається знання структури зберігання даних і мови запитів. Користувач оперує звичними термінами бізнес-середовища - відвантаження, товар, клієнт. Вся інформація в сховищі зберігається в структурах типа «зірка», де в центрі розташовані таблиці фактів, а «променями» є виміри. Така архітектура сховища найбільш адекватна задачам аналізу даних. Кожна «зірка» називається *процесом* і описує певну дію, наприклад, продажу товару, відвантаження, поступлення грошових коштів та інше. У Deductor Warehouse може одночасно зберігатися множина процесів, що мають загальні виміри, наприклад, «Товар», що фігурує в «Поступленні» і в «Відвантаженні».

Виміри можуть бути як простими списками, наприклад, дата. Так і містити додаткові стовпці, названі *властивостями*. Наприклад, вимір «Товар» може складатися з «Найменування товару» - власне вимір (первинний ключ) і «Ваги», «Об'єму» та інше - властивості даного виміру.

Загрузка даних в Deductor Warehouse виконується за допомогою Deductor Studio, причому дану операцію можна виробити з будь-якими даними, імпортованими або обробленими програмою. Це забезпечує широкі можливості - до завантаження можна провести весь цикл передобробки і очищення даних, наприклад видалити аномальні значення, заповнити пропуски і завантажити в сховище очищені і необхідним чином трансформовані дані.

Порядок виконання роботи.

ЧАСТИНА 1. Створення файлу сховища і організація доступу до нього.

1. Сховище даних Deductor Warehouse створюється на основі бази даних Interbase\Firebird, тому для створення сховища необхідно, аби в системі була

доступна одна з баз даних вказаного типа. Для цього в меню «*Пуск – Програми»* знаходимо *Firebird* і запускаємо *Firebird Server*.

2. Для створення і підключення сховища даних необхідно в меню «*Вид*» вибрати команду «*Источники данных*». В результаті буде відкрита панель «*Источники данных*».

Панель «*Источники данных*» дозволяє здійснювати швидкий вибір, налаштування і підключення нових джерел даних - баз даних різних типів і сховищ даних. Для вже підключених джерел тут можна отримувати інформацію про їх поточні параметри, а також при необхідності змінювати ці параметри.

Джерела даних представлені у вигляді деревовидного ієрархічного списку. Він містить дві основних гілки: бази даних і сховища даних. Зліва від заголовка обох гілок розташований значок «+», який означає що гілка містить підгілки - бази даних або сховища. Клацання по значку «+» розвертає гілку, роблячи видимими всі її бази даних і сховища. Якщо гілка повністю розгорнута, то зліва від її імені з'являється значок «-« який, навпаки, дозволяє скрутити її.

При роботі із списком джерел даних можуть бути виконані наступні операції, доступні за допомогою кнопок на панелі інструментів або через контекстне меню, що викликається клацанням правою кнопкою миші по відповідному вузлу:

- додати нову базу даних - дозволяє підключити до системи нову базу даних і налаштувати її параметри;
- додати нове сховище даних - дозволяє підключити до системи нове сховище даних і налаштувати його параметри;
- видалити вузол - видаляє вузол списку джерел даних і всі його підвузли.

Окрім цього, через контекстне меню можуть бути доступна команда «*Показать*», яка відображає список параметрів виділеного підключення.

3. Викликати контекстне меню клацанням правої кнопки миші в будь-якому місці панелі «*Источники данных*» і вибрати команду «*Создать локальное ХД*». В результаті буде відкрито вікно «*Укажите файл БД для хранилища*». У цьому

вікні в списку «*Папка*» вибрати папку, в якій має бути створений новий файл сховища, а в полі «*Имя файла*» вказати ім'я, яке буде привласнено файлу сховища (за умовчанням пропонується Warehouse.gdb).

4. Підключити новий файл сховища до системи. Для цього викликати контекстне меню для пункту «*Базы данных*» панелі «*Источники данных*» і вибрати команду «*Добавить базу данных*» (або скористатися кнопкою ) і у вікні, що відкрилося, «*Виберите тип базы данных*» вибрати базу Interbase/Firebird, тобто платформу, яка використовується в даний час в Deductor для організації сховища даних. Для цього потрібно виділити тип бази в списку і клацнути по кнопці «*Выбрать*». В результаті відкриється список параметрів, які необхідно налаштувати для підключення сховища даних:

- *«Имя»* - вказується унікальний ідентифікатор по якому виконується звернення до бази. Він має бути англомовним. Відображаємий далі рядок «*База данных*», служить не для налаштування яких-небудь параметрів. Просто вона дозволяє скрутити або розвернути список властивостей бази даних. У лівій частині рядки стоїть значок, який має вигляд «+», якщо список властивостей бази даних згорнутий і «-», якщо розгорнутий.
- «*База данных*» - тут необхідно вказати ім'я файлу бази даних Interbase\Firebird (gdb-файла), який потрібно підключити і повний шлях до нього. Можна ввести їх вручну з клавіатури, але зручніше скористатися кнопкою вибору  в правій частині рядка. Клацання по ній відкриє вікно «*Подключение к БД IB\FB*».
- «*Пользователь*» - вказати ім'я користувача по якому буде виконуватися доступ до сховища даних.
- «*Пароль*» - вказати пароль доступу до сховища для вказаного вище імені користувача.
- «*Показывать системные таблицы*» - даний прапорець дозволяє або забороняє відображення системних, тобто використовуваних в службових цілях, таблиць сховища даних.

5. Перевірка доступу. Після налаштування всіх необхідних параметрів можна перевірити доступ до нової бази даних. Для цього слід скористатися кнопкою  , внаслідок чого буде зроблена спроба з'єднання з базою даних. Якщо з'єднання буде успішним, то з'явиться повідомлення «*Соединение успешно протестировано*» і база даних буде готова до роботи. Інакше буде видано повідомлення про помилку внаслідок якої з'єднання неможливе. В цьому випадку потрібно перевірити параметри підключення, при необхідності внести до них зміни і протестувати підключення ще раз.

Після виконання вказаних дій, вибраний файл сховища відображуватиме в якості вузла гілки «*База данных*» панелі «*Источники данных*». Тепер її можна буде зареєструвати як сховище даних. Для цього потрібно викликати контекстне меню для пункту «*Хранилища данных*» і вибрати команду «*Добавить хранилище данных*» або скористатися кнопкою  . У відкрившимуся вікні «*Выберите базу данных*» виділити базу даних Interbase\Firebird, яку ви хочете використовувати в якості сховища і клацнути по кнопці «*Выбрать*». В результаті нове сховище з'явиться як вузол гілки «*Хранилища данных*» панелі «*Источники данных*», одночасно буде відкритий список параметрів сховища:

- *«Имя»* - вказується ім'я по якому сховище даних ідентифікуватиметься в системі. Воно має бути унікальним і англомовним.
- «*Название подключения*» - відображується найменування підключення, відповідне базі даних, вибраній як сховище даних (змінити його тут не можна).
- «*Тип базы данных*» - вказується тип бази даних, використовуваної як сховище даних (Interbase\Firebird).

ЧАСТИНА 2. Завантаження даних в сховище.

1. Сформуємо в MS Excel (зовнішнє джерело) інформацію про відвантаження товару, представлену таблицею та інформацію про товар, також представлену відповідною таблицею.

ПРИКЛАД.

Таблица. Отгрузки товара.

Номер товара	Дата	Клиент	Количество	Сумма к оплате	Наценка
1	01.03.04	ОАО «ССК»	1000	7500	1000
2	10.03.04	ООО «ДСК»	1500	15000	3000
3	12.03.04	ООО «ДСК»	900	9000	2000
3	12.03.04	ОАО «ССК»	2000	20000	4400
4	15.03.04	ООО «ДСК»	500	6000	1000

Номер товара	Группа	Цвет	Ширина	Высота	Длина
1	Силикатный	Белый	120	85	250
2	Силикатный	Тонированный	60	88	190
3	Керамический	Красный	120	65	250
4	Керамический	Песочный	120	65	250

2. Завантаження даних в сховище виконується з їх проміжним імпортом в Deductor Studio. Для цього необхідно виконати наступні дії:

- за допомогою «*Мастер импорта*» завантажити вибірку даних в Deductor Studio;
- запустити «*Мастер экспорта*» і в списку форматів вибрати «*Deductor Warehouse – процесс*» або «*Deductor Warehouse – измерение*». У першому випадку буде завантажена вся вибірка у вигляді процесу («зірки»), а в другому - значення лише одного або декількох вимірів, вибраних користувачем. При завантаженні вимірів все відбувається аналогічно, як і для процесу, але крок «*Выбор процесса*» пропускається.

«*Мастер импорта»* призначений для автоматизації імпорту даних з будь-якого джерела, передбаченого в системі. Аби викликати «*Мастер импорта»*, досить скористатися кнопкою  у верхній частині панелі «*Сценарии*» або вибрати відповідну команду з контекстного меню, що

викликається клацанням правої кнопки миші в будь-якому місці панелі«*Сценарии*».

На першому кроці застосування «*Мастер импорта»* відкривається список всіх передбачених в системі типів джерел даних. Серед них слід вибрати джерела потрібного типа і, для переходу на наступний крок, клацнути по кнопці «*Далее*». Число кроків «*Мастер импорта»*, а також набір параметрів, що налаштовуються, різний для різних типів джерел. На кожному кроці «*Мастер импорта»* доступні кнопки «*Далее*» і «*Назад*», які відповідно дозволяють перейти до наступного кроку або повернутися на попередній крок для внесення змін до раніше налагоджених параметрів.

Імпортуємо дані, представлені таблицями MS Excel в програму. Для цього необхідно викликати двічі «*Мастер импорта»* і завантажити дані спочатку про відвантаження, потім про товар (порядок значення не має).

3. Налаштування параметрів імпорту даних. На даному кроці користувач повинен в полі «*База данных*» ввести ім'я файлу книги MS Excel і шлях до нього. Зручніше не вводити його вручну, а скористатися кнопкою вибору  в правій частині поля. В результаті відкриється стандартне вікно «*Открытие файла*» Windows, в якому слід знайти і відкрити потрібний файл. Як тільки натискуватиме кнопка «*Открыть*», це вікно закриється, а вибраний файл і шлях до нього, з'являться в полі «*База данных*».

Імпортувати дані з вибраного файлу можна двома способами - з окремої таблиці на аркуші книги MS Excel або за допомогою SQL-запросу. Аби вибрати один із способів, потрібно активізувати відповідний пункт. Якщо вибраний пункт «*Таблица в базе данных*», то в розташованому нижче полі, за допомогою кнопки  відкривається список, в якому можна вибрати один з листів книги або таблицю бази даних (під таблицею Excel розуміється іменована область аркуша книги), звідки слід імпортувати дані.

4. Налаштування параметрів стовпців. На даному кроці потрібно налаштувати відповідні параметри стовпців даних, що імпортуються, вказавши відповідні значення в полях:

- «*Имя столбца*» - відображується ім'я стовпця, тобто його ідентифікатор, що використовується в базі даних. Змінити ім'я стовпця тут не можна.
- «*Название столбца*» - вказується назва (мітка), під якою даний стовпець буде видний в таблиці, крос-таблиці або на діаграмі після імпорту. Бажано, аби воно відображало зміст стовпця.
- «*Размер*» - вказується ширина стовпця в символах.
- «*Тип данных*» - вказується тип даних, що містяться в стовпці. Він також задається в базі даних і змінити його тут не можна.
- «*Назначение*» - визначає порядок використання стовпця при подальшій обробці імпортованих даних. Найчастіше використовується «і» – інформаційний.

5. Вибір способу відображення. Користувач повинен вибрати, в якому вигляді відображатимуться імпортовані дані. Для цього досить помітити потрібні види відображення прапорцями і клацнути по кнопці «*Далее*». Для вибірки даних, отриманих в результаті імпорту з сховища, доступні наступні види відображення: «*Таблица, Статистика, Диаграмма, Гистограмма*». Вибрати і виконати відображення даних *самостійно*.

ЧАСТИНА 3. Використання майстра експорту.

1. Запустити майстер експорту і в списку форматів вибрати «*Deductor Warehouse – процесс*» або «*Deductor Warehouse – измерение*». У першому випадку буде завантажена вся вибірка у вигляді процесу («зірки»), а в другому - значення лише одного або декількох вимірів, вибраних користувачем.

Процеси. Для експорту даних в процес сховища даних потрібно виконати наступні дії. У списку доступних сховищ даних вибрати те, в яке потрібно виконати завантаження. Потім відбувається вибір процесу, в який завантажуються дані. Якщо вибірка, що завантажується, повинна бути додана до вже існуючого процесу, то досить виділити цей процес. Якщо ж вибірка завантажується в сховище як новий процес, потрібно встановити прапорець

«*Новый процесс*», а в полі праворуч від його вказати ім'я нового процесу. За бажанням в полі «*Описание*» можна ввести короткий опис процесу в довільній формі. Далі необхідно вибрати виміри і факти процесу, які мають бути завантажені в сховище. У списку «*Поле в источнике данных*» представлені всі поля вибірки, що завантажується. Для кожного з них в стовпці «*Назначение поля*» слід вказати варіант його використання при завантаженні:

- факт - значення поля будуть завантажені як значення фактів (у центр «зірки»);
- вимір - значення поля будуть завантажені як значення вимірів (у промені «зірки»);
- невживане - поле завантажене не буде.

У списку «*Поле в хранилище*» можна вибрати поле сховища, під яким дане поле джерела буде збережено в сховищі. наприклад, якщо в сховищі вже були завантажені виміри, то потрібно вказати, що дане поле джерела даних буде збережено в конкретному вимірі. Якщо залишити пропоноване за умовчанням значення «*Новое*», то поле в сховищі буде представлено під тим же ім'ям, що і у вхідній вибірці.

Для завантаження процесу необхідно вибрати хоч би одне поле виміру і одне поле фактів. Установка прапорця «*Создать вспомогательную таблицу*» дозволяє створити для завантажуваного процесу в сховищі допоміжну таблицю, яка дозволить прискорити доступ до даних в сховищі.

Виміри. Завантажувати вимір має сенс, якщо воно має властивості. Наприклад, у виміру «Товар» можуть бути властивості «Вага», «Колір», «Розмір» (рис. 8.1). Його потрібно завантажити окремо. При завантаженні процесу вимір з властивостями завантажується по його ідентифікатору. При завантаженні виміру завантажуються також і його властивості. Вимір «Дата» жодних властивостей не має. Тому він завантажується лише разом із завантаженням процесу.

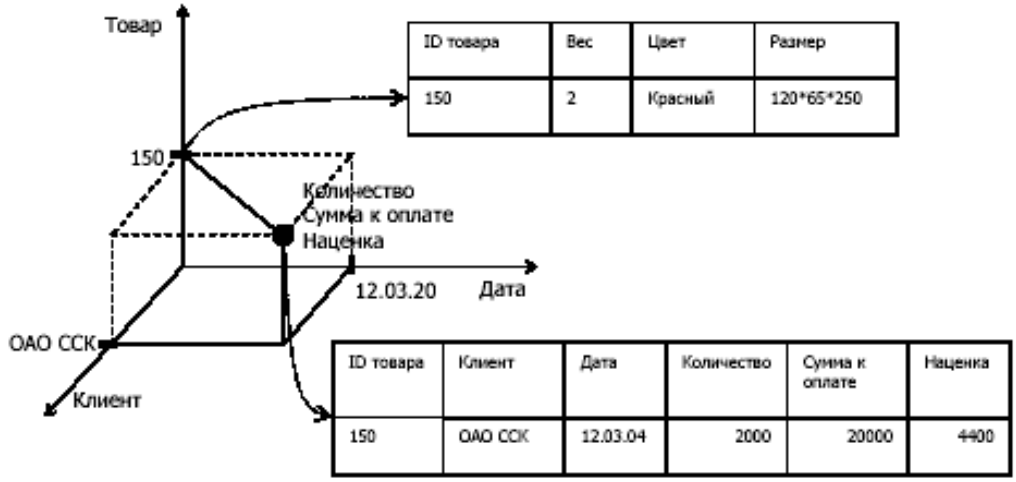


Рис. 8.1. Завантаження виміру в сховище даних.

Для експорту даних у вимір сховища даних потрібно виконати наступні дії. У списку доступних сховищ даних вибрати те, в яке потрібно виконати завантаження. Далі необхідно вказати вимір, що завантажується, і його властивості. У списку «*Поле в хранилище*» представлені всі поля завантажуваної вибірки. Для кожного з них в стовпці «*Назначение поля*» слід вказати варіант його використання при завантаженні:

- вимір – ідентифікатор завантажуваного виміру;
- властивість – поле буде завантажено як значення властивості виміру;
- невживане - поле завантажене не буде.

Призначення «вимір» в даному випадку можна вказати лише для одного поля джерела. Всі значення виміру мають бути унікальними і однозначно ідентифікувати вимір. У списку «*Поле в хранилище*» можна вибрати поле сховища, під яким дане поле джерела буде збережено в сховищі. Тобто, якщо вже були раніше завантажені виміри, то для завантаження поля з джерела в існуючий вимір або властивість, потрібно тут його вибрати. Якщо залишити пропоноване за умовчанням значення «*Новое*», то поле в сховищі буде представлено під тим же ім'ям, що і у вхідній вибірці. Для завантаження виміру необхідно вибрати хоч би одне поле виміру. Поля з властивостями можуть бути відсутніми.

2. Вивантажимо дані в сховищі. Спочатку потрібно визначитися, що є процесом, фактами і вимірами. *Процес* – це відвантаження товарів. Він представлений першою таблицею. *Виміри* – це товар, дата і клієнт. *Факти* – це кількість, сума до оплати і націнка. Вимір «Товар» містить властивості, тому його потрібно завантажувати в сховищі окремо. З нього і почнемо. Виберемо вузол з таблицею про товар і викличемо майстер експорту. У цьому майстрові потрібно вибрати пункт «*Deductor Warehouse – измерение*». Потім потрібно задати значення полям таблиці, як показано на рис. 8.2.

Поле в источнике данных	Назначение поля	Поле в хранилище
Номер товара	Измерение	Новый
Группа	Свойство	Новый
Цвет	Свойство	Новый
Ширина	Свойство	Новый
Высота	Свойство	Новый
Длина	Свойство	Новый

Рис. 8.2. Задання значення полям таблиці вимірів.

Номер товару є виміром і однозначно визначає товар. Група, колір, ширина, висота і довжина – властивості товару. Вони можуть бути різними у різних товарів. Крім того, їх згодом можна буде змінити.

У даному прикладі більше немає вимірів з властивостями. Тому можна приступати до вивантаження процесу. Виберемо вузол з таблицею відвантажень і викличемо майстер експорту. У майстрові потрібно вибрати пункт «*Deductor Warehouse – процесс*». Для нашого нового процесу необхідно задати ім'я. Назвемо його «Відвантаження». Далі слід вказати що є вимірами і фактами, як показано на рис. 8.3.

Поле в источнике данных	Назначение поля	Поле в хранилище
Номер товара	Измерение	Номер товара
Дата	Измерение	Новый
Клиент	Измерение	Новый
Количество	Факт	Новый
Сумма к оплате	Факт	Новый
Наценка	Факт	Новый

Рис. 8.3. Опис процесу в майстрові експорту.

Вимір «Товар» вже є в сховищі. Тому в стовпці «*Поле в хранилище*» напроти рядка «Номер товара» вибрано вимір сховища «Номер товара». Таким чином, буде створена структура сховища і завантажені дані про відвантаження товару.

ЧАСТИНА 4. Імпорт даних зі сховища.

1. Імпорт даних зі сховища виконується за допомогою майстра імпорту, в якому необхідно вибрати тип джерела даних Deductor Warehouse.

Процеси. Спершу потрібний вибрати сховище даних (ХД), з якого потрібно виконати імпорт. У вікні вибору ХД представлено дві колонки «*Название*» і «*Описание*». У колонці «*Название*» вказується ім'я сховища, під яким воно зареєстроване в системі, а в списку «*Описание*» (це необов'язковий параметр) - коротка характеристика змісту сховища, що вводиться користувачем при його створенні.

Потім потрібно вибрати процес, з якого потрібно імпортувати дані. Інформація, що відноситься до якого-небудь об'єкту або бізнес-процесу, представлена в сховищі у вигляді «зірки», де в центрі розташовані таблиці фактів, а «променями» є виміри. Кожна така структура називається *процесом*. У загальному випадку, таких процесів в сховищі міститься декілька. Тому кожного разу при зверненні до сховища необхідно вибрати процес, з якого повинні імпортуватися дані. У списку «*Название*» представлені найменування процесів, що містяться в даному сховищі, а в списку «*Описание*» - коротка характеристика процесу (це необов'язковий параметр). І назва, і опис вводяться користувачем при створенні процесу. Далі потрібно визначити, які виміри, властивості і факти з вибраного на попередньому кроці процесу мають бути імпортовані. Це необхідно тому, що процес може містити багато вимірів і фактів, а користувачеві будуть потрібні лише деякі з них.. Всі виміри і факти вибраного процесу представлені у вигляді дерева, де виміри утворюють одну гілку, а факти - іншу. Виміру позначені значком  , а факти -  . Зліва від кожного виміру і факту розташований прапорець. Установка прапорця дозволяє

вибрати відповідний вимір або факт для імпорту, а скидання прапорця, навпаки, виключає вимір або факт з числа тих, що імпортуються. Окрім цього, з виміром можуть бути зв'язані одне або декілька властивостей. В цьому випадку властивості утворюють підгілку гілки виміри, де кожна властивість позначена значком  . Зліва від кожної властивості також розташований прапорець, який дозволяє вибирати або забороняти імпорт властивості.

Далі потрібно визначити *зріз даних* – тобто вибрати значення вимірів, вибраних на попередньому кроці, які будуть імпортовані. Цей крок бажаний тому, що кількість значень виміру може бути дуже великою, а завантаження всіх значень недоцільне. Тому вибір лише тих значень виміру, які представляють інтерес в даному випадку, допоможе заощадити час завантаження даних, і не захаращуватиме отриману вибірку. Аби задати параметри зрізу для даного виміру необхідно виділити його в полі «*Измерение*», вибрати умову і клацнути по кнопці «*Выбрать*». В результаті відкриється діалогове вікно «*Выбор среза*». У лівій частині вікна буде представлений список всіх значень даного виміру. Аби вибрати значення досить виділити його клацанням миші і клацнути по кнопці  або просто натискувати «Enter», після чого, вибране значення з'явиться в правому полі і, таким чином, буде вибране для побудови зрізу. При необхідності можна відмінити вибір значення, для чого виділити його в правому полі і клацнути по кнопці  . Аби очистити весь список вибраних значень слід використовувати кнопку  . Після закриття вікна «*Выбор среза*», список вибраних значень з'явиться в полі «*Выбранные значения измерения*».

Список умов містить декілька значень:

- «*Нет*» – фільтрація по вказаному виміру не виконується;
- *«Входит в список»* – вибираються значення, які входять в список вибраних;
- *«Не входит в список»* - вибираються значення, які не входять в список вибраних;

- *«В интервале»* – вказується лише для вимірів типа «Дата». Вибираються записи, для яких вимір лежить в заданому діапазоні дат;
- «*Вне интервала*» – вказується лише для вимірів типа «Дата». Вибираються записи, для яких вимір не входить в заданий діапазон дат;
- «*За последний период*» - вказується лише для вимірів типа «Дата». Вибираються записи, для яких вимір лежить у вказаний проміжок часу, передуючий поточній даті.

2. Імпортуємо з сховища дані про кількість відвантаженого товару в розрізі дат і товарів по клієнтові ВАТ «ССК», залишивши властивість товару «Колір». Викличемо майстер імпорту і виберемо джерело «Deductor Warehouse». Далі виберемо наше сховище із списку і процес «Отгрузки». Відзначимо виміри і факти, що імпортуються зі сховища, як показано на рис. 8.4.

Рис. 8.4. Вибір процесу «Отгрузки».

Вкажемо фільтр по клієнтові (рис. 8.5).

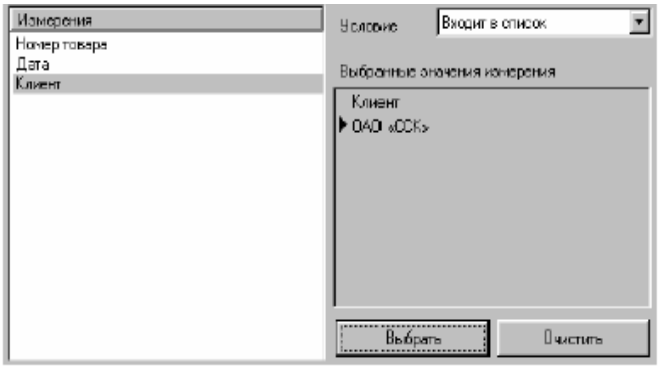


Рис. 8.5. Фільтр по клієнту ВАТ «ССК».

Таким чином, буде отримана таблиця наступного змісту (рис. 8.6). Тобто отримали кількість в розрізі товару, дати і конкретного клієнта.

Номер товара	Группа	Цвет	Дата	Клиент	Количество
1	Силикатный	Белый	01.03.2004	ОАО «ССК»	1000
3	Керамический	Красный	12.03.2004	ОАО «ССК»	2000

Рис. 8.6. Результат зрізу даних.

8.2. Лабораторна робота №2 «Створення і використання OLAP-кубів».

Завдання до лабораторної роботи.

1. Створити OLAP–куб і отримати крос-таблицю для варіанту сховища даних, розробленого в попередній лабораторній роботі відповідно до заданого варіанту.

2. Виконати всі операції з крос-таблицею, відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 1».

3. Здійснити агрегацію даних відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 2».

4. Виконати фільтрацію даних крос-таблиці відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 3».

Теоретична частина.

Куб є одним з поширених методів комплексного багатовимірного аналізу даних, що отримали назву *OLAP (On-Line Analyzing Process)*. У його основі лежить представлення даних у вигляді багатовимірних кубів, названих також *OLAP-кубами* або *гіперкубами*. По осях багатовимірної системи координат відкладаються ті або інші параметри бізнес-процесу, що аналізується. Наприклад, для продажів це може бути товар, регіон, тип покупця. Зазвичай як один з вимірів використовується час. По осях (вимірам) багатовимірної системи координат знаходяться дані, які кількісно

характеризують процес - факти. Це можуть бути об'єми продажів в штуках або в грошовому вираженні, залишки на складі, витрати, суми і тому подібне. Користувач, що аналізує інформацію, може виконувати перетин куба по різних напрямах, отримувати звідні (наприклад, по роках) або, навпаки, детальні (по тижнях) дані і здійснювати інші операції необхідні для ефективного аналізу.

Результатом візуалізації даних, організованих в OLAP-куб є *крос-таблиця* - зручний засіб візуалізації багатовимірних даних і здобуття необхідних форм звітів. Крос-таблиця будується на основі багатовимірного представлення у вигляді OLAP-куба і містить виміри і факти, визначені при побудові куба. Основною особливістю крос-таблиці є те, що *її структура не є жорстко визначеною*. Маніпулюючи за допомогою миші заголовками вимірів, користувач може добитися, аби крос-таблиця виглядала найбільш інформативно. У системі Deductor Studio передбачена можливість будувати на основі крос-таблиці крос-діаграму, основна особливість якої полягає в тому, що вона автоматично перебудовуватиметься відповідно до будь-яких змін крос-таблиці.

Порядок виконання роботи.

ЧАСТИНА 1. Побудова крос-таблиць.

1. Розміщення вимірів. Крос-таблиця є розміщенням багатовимірних даних на площині у вигляді звідної таблиці. Отже, перш, ніж будувати цю таблицю необхідно вказати виміри і факти. Наприклад, виміри – це «Місяць, Найменування товару, Група товару», а факт – «Кількість проданого товару». Виміри можуть бути розміщені в рядках і стовпцях крос-таблиці. При запуску «*Мастер настройки отображения*» спочатку весь список вибраних вимірів відображується у вікні «*Доступные измерения*». Натискуючи кнопки «>« та «>>« справа і знизу від цього вікна можна розміщувати вибрані виміри в рядках і стовпцях таблиці (рис. 8.7).

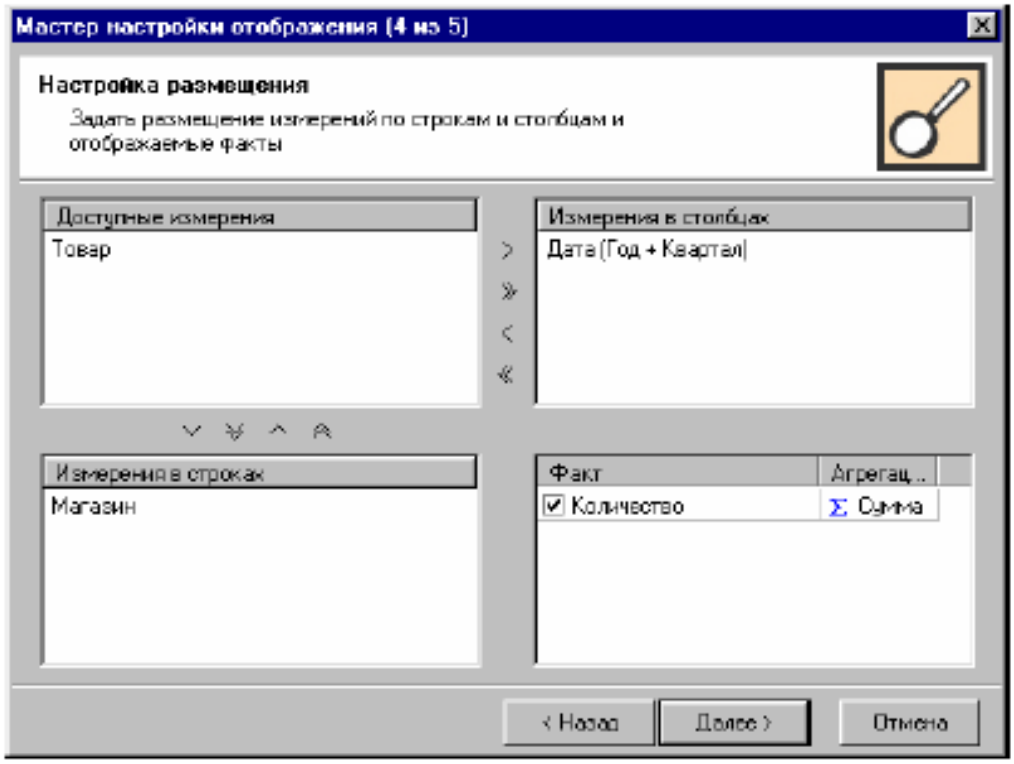


Рис. 8.7. Розміщення вимірів в таблиці.

Побудована крос-таблиця для раніше розглянутого прикладу виглядатиме таким чином (рис. 8.8).

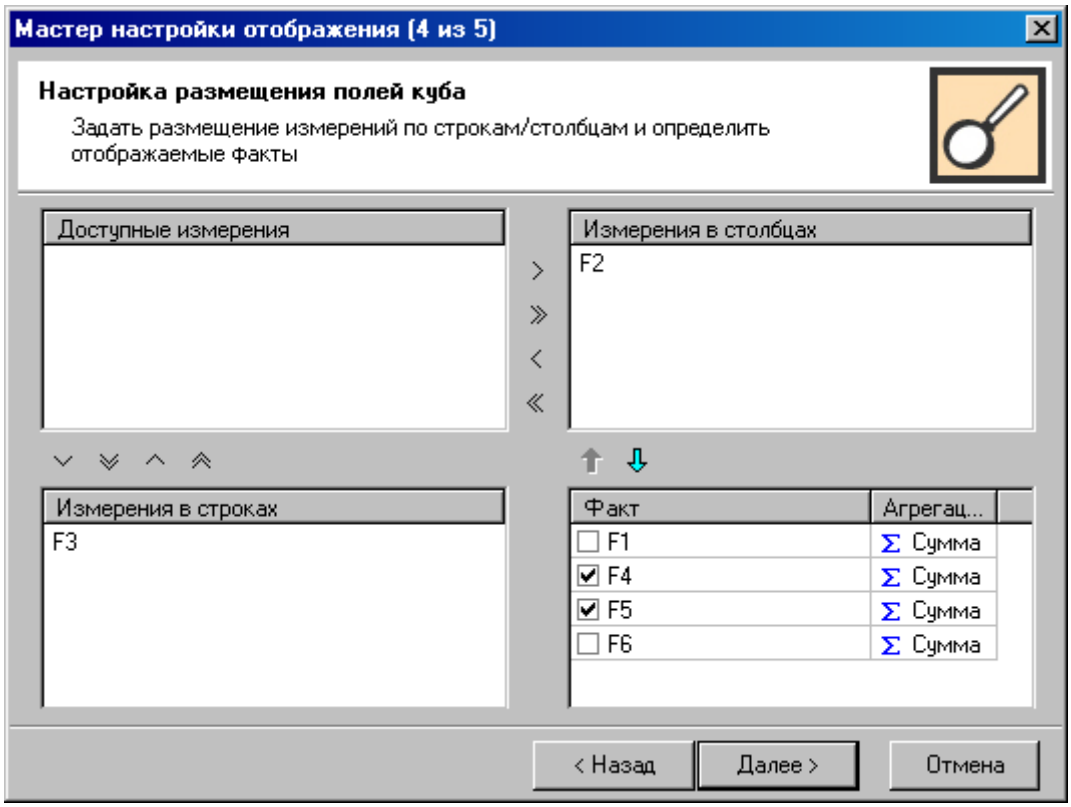


Рис. 8.8. Побудування крос-таблиці.

Тут можна так само вибрати, які факти відображувати в крос-таблиці на пересіченні вимірів і яку функцію застосовувати при їх агрегації (об'єднанні). У даному прикладі факт «Кількість» і «Сума до оплати» підсумовуватиметься. Побудована крос-таблиця для даного прикладу виглядатиме таким чином (рис. 8.9).

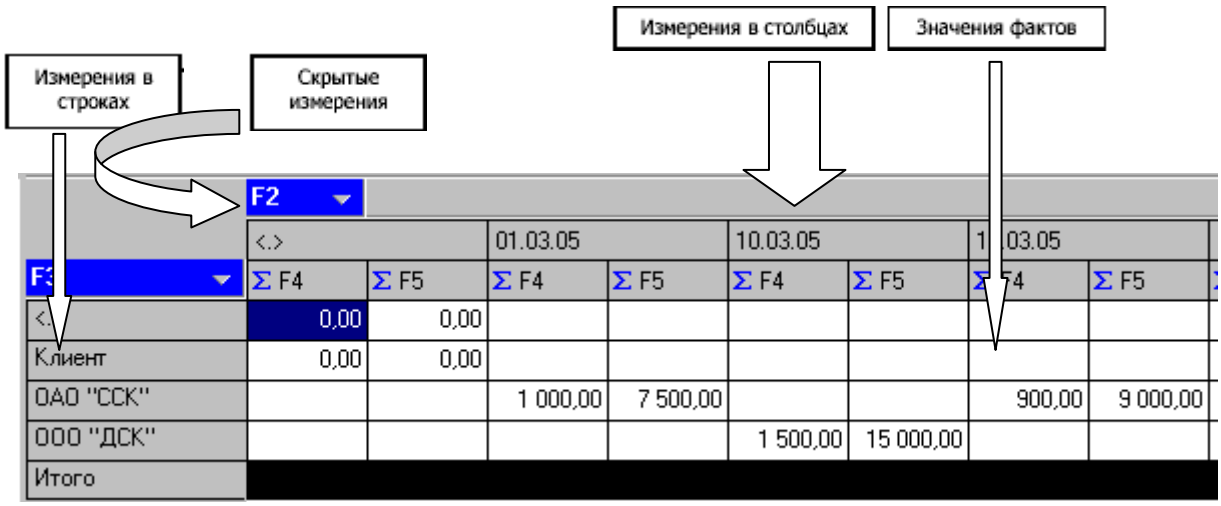
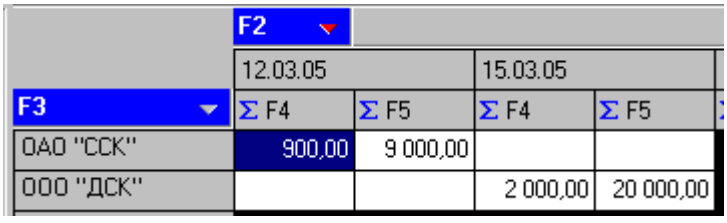


Рис. 8.9. Крос-таблиця для наведеного прикладу.

2. Виміри в крос-таблиці зображаються спеціальними полями. Сині поля показують виміри, що беруть участь в побудові таблиці. Зеленими полями відображуються приховані виміри, що не беруть участь в побудові таблиці. Є можливість перебудовувати таблицю за допомогою миші «на льоту». Зробити це можна, якщо перетягувати поля із заголовками вимірів. Приведемо різні варіанти зміни таблиці в такий спосіб.

- Зробити вимір, що бере участь в побудові таблиці прихованим. Для цього потрібно перетягнути поле із заголовком виміру в рядок з прихованими вимірами або в полі F2, що розкривається, прибрати відмітки. Наприклад, нас цікавлять продажі лише за 12.03.05 і 15.03.05. Результат



- Зробити прихований вимір таким, що бере участь в побудові таблиці. Його можна додати до вимірів в рядках або стовпцях за допомогою перетягання мишею.
- Поміняти два виміри місцями. Змінювати розташування вимірів можна, використовуючи *операцію транспонування таблиці*. В результаті транспонування дані, що раніше відображалися в рядках, відображатимуться в стовпцях, а дані в стовпцях перетворяться в рядки. Транспонування у багатьох випадках дозволяє оперативно зробити таблицю зручнішою для сприйняття. Аби застосувати операцію транспонування слід скористатися кнопкою  «*Транспонировать таблицу*» на панелі інструментів.

У наведених вище прикладах крос-таблиця будується по всіх значеннях вимірів. Проте інколи виникає необхідність побудувати таблицю в розрізі лише деяких значень, наприклад, за першу і третю декади. Включати або виключати значення вимірів в таблицю можна, натискує на трикутник в полі заголовка виміру. Наприклад, якщо натискувати на трикутник в полі заголовка виміру «Дата» відкриється список його значень.

ЧАСТИНА 2. Способи агрегації і відображення фактів.

1. Передбачено декілька способів об'єднання фактів в крос-таблиці:

- сума – обчислюється сума об'єднуваних фактів;
- мінімум – серед всіх об'єднуваних фактів в таблиці відображається лише мінімальний;
- максимум - серед всіх об'єднуваних фактів в таблиці відображається лише максимальний;
- середнє – обчислюється середнє значення об'єднуваних фактів;
- кількість – в крос-таблиці відображуватиметься кількість об'єднаних фактів.

2. Для зміни способу агрегації фактів потрібно викликати вікно «*Настройка размещения*», натискує кнопку «*Изменение размещения*» на панелі

інструментів. Також існує можливість відображати факти не лише їх значеннями, але і у відсотках по рядках або стовпцях крос-таблиці. Для застосування такого способу відображення потрібно натискувати кнопку «*Отображать факты как*» на панелі інструментів.

У крос-таблиці факти відображуються з двома знаками після коми і вирівняні по правому краю чарунки. Є можливість змінити форматування елементів таблиці, викликавши вікно «*Настройка форматов отображения данных*». Викликати його можна, натискує кнопку «*Настройка отображения фактов*» (рис. 8.10).

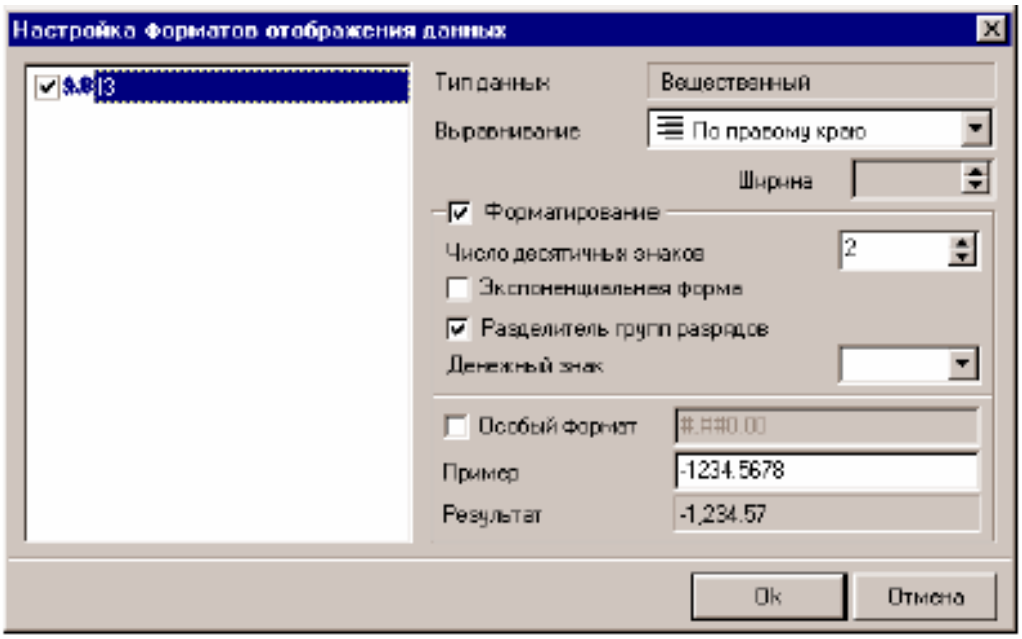


Рис. 8.10. Вікно «*Настройка форматов отображения данных*».

Призначення полів наступне:

- «*Выравнивание*» – визначає вирівнювання значень у чарунках. Може набувати значень: «*По левому краю*», «*По центру*», «*По правому краю*»;
- «*Форматирование*» – застосувати особливе форматування або залишити все як є;
- «*Число десятичных знаков*» – визначає число знаків після коми;
- «*Експоненциальная форма*» – якщо прапорець встановлений, то число відображатиметься в експоненціальній формі. Наприклад, число 153.47 виглядатиме як 1.5347E+2;

- «*Разделитель групп разрядов*» – відображати або не відображати роздільник розрядів. Тобто число може виглядати так 1289 або так 1,289.

ЧАСТИНА 3. *Селектор – фільтрація даних крос-таблиці.*

1. Селектор є потужним засобом фільтрації даних в крос-таблиці. Фільтрація може виконуватися двома способами:

- по значеннях фактів;
- по значеннях вимірів, шляхом безпосереднього вибору значень із списку, або відбору їх по умові. Фільтрація виконується окремо по кожному виміру.

Аби приступити до роботи з селектором досить на панелі інструментів натискувати на кнопку  «*Селектор*», після чого буде відкрито вікно селектора. Зліва відображуються всі виміри крос-таблиці і поле «*Факты*», що означає фільтрацію по фактах. Справа знаходяться елементи:

- «*Измерение*». Фільтрація передбачає, що в таблиці залишиться лише частина значень деякого виміру. Це поле якраз і задає вимір, значення якого будуть відфільтровані.
- «*Факт*». У крос-таблиці може міститися один та більше фактів. Фільтрація відбуватиметься по значеннях вибраного тут факту.
- «*Агрегация*». Можна вибрати функцію агрегації, відповідно до якої слід виконати відбір записів. В результаті будуть вибрані лише ті записи, агреговані значення яких задовольняють вибраній умові.
- *«Условие*». Умова відбору записів по значеннях вибраного факту.

Поле умова може приймати наступні значення:

- «*Первые N*». Значення виміру сортуються в порядку убування факту і вибираються перші N значень вимірів. Таким чином, можна, наприклад, знаходити лідерів продажів – перші 10 товарів, що найбільше продаються, або перші 5 найбільше вдалих днів.

- «*Последние N*». Значення виміру сортуються в порядку убування факту і вибираються останні N значень вимірів. Наприклад, 10 найменш популярних товарів.
- «*Доля от общего*». Значення виміру сортуються в порядку убування факту. У цій послідовності вибирається стільки перших значень виміру, щоб вони в сумі давали задану долю від загальної суми. Наприклад, можна відібрати клієнтів, що приносять 80% прибутки, – група А по АВС класифікації.
- «*Диапазон*». Результатом відбору будуть записи, для яких значення відповідного факту лежить в заданому діапазоні.
- «*Більше*». Будуть відібрані записи, значення відповідного факту, для яких буде більше вказаного значення.
- «*Меньше*». Будуть відібрані записи, значення відповідного факту, для яких буде менше вказаного значення.
- «*Равно*». Будуть відібрані записи, значення відповідного факту, для яких буде рівно вказаному значенню.

Наведемо приклад. Хай нам потрібно визначити товари, що мають найбільший попит. Вхідна крос-таблиця містить 15 товарів (рис. 8.11).

Товар	Σ Количество
Товар 1	55.00
Товар 10	167.00
Товар 11	110.00
Товар 12	133.00
Товар 13	162.00
Товар 14	145.00
Товар 15	54.00
Товар 2	191.00
Товар 3	125.00
Товар 4	120.00
Товар 5	145.00
Товар 6	163.00
Товар 7	199.00
Товар 8	147.00
Товар 9	154.00
Итого	2,070.00

Рис. 8.11. Вхідна крос-таблиця товарів.

Застосуємо до неї селектор (рис. 8.12).

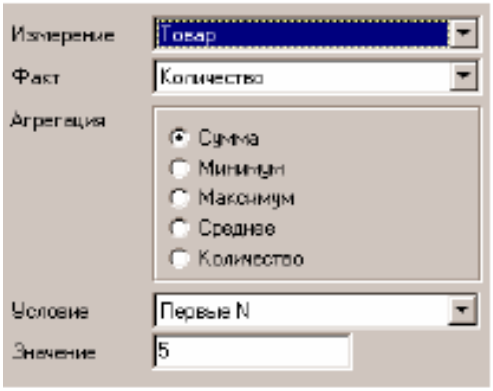


Рис. 8.12. Селекторний аналіз даних.

В результаті отримаємо 5 найбільш популярних товарів. Таку вибірку можна отримати по будь-якому факту. У даному прикладі – це кількість. Тому ми отримали товари, що найбільше продавалися. Якщо відфільтрувати по націнці, то отримаємо найбільш прибутковий товар.

2. Крос-діаграма. Крос-діаграма є діаграмою заданого типа, побудованою на основі крос-таблиці. Основна відмінність крос-діаграми від звичайної діаграми в тому, що вона однозначно відповідає поточному стану крос-таблиці і при будь-яких її змінах змінюється відповідно. Наведемо приклад крос-діаграми для наступної крос-таблиці (рис. 8.13).

	Дата (Год + Квартал)				
Магазин	2004-Q1	2004-Q2	2004-Q3	2004-Q4	Итого
Магазин 1	364.00	326.00	256.00	187.00	1,133.00
Магазин 2	251.00	255.00	267.00	164.00	937.00
Итого	615.00	581.00	523.00	351.00	2,070.00

Рис. 8.13. Крос-таблиця роботи магазинів.

Крос-діаграма для неї має наступний вигляд (рис. 8.14).

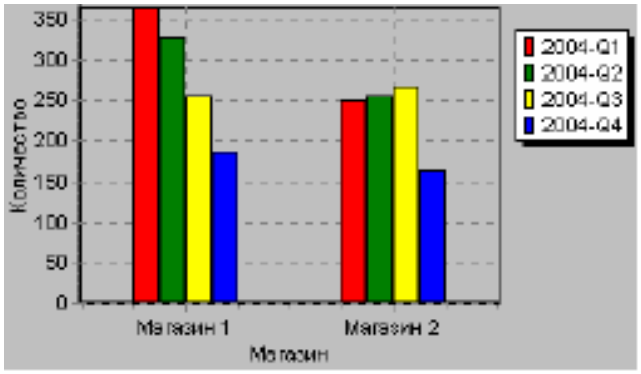


Рис. 8.14. Крос-діаграма роботи магазинів.

На цій діаграмі можна спостерігати поквартальну тенденцію продажів в різних магазинах. До крос-діаграми, так само як і до крос-таблиці можна застосовувати транспонування. Результат транспонування приведеної вище діаграми буде наступний (рис. 8.15).

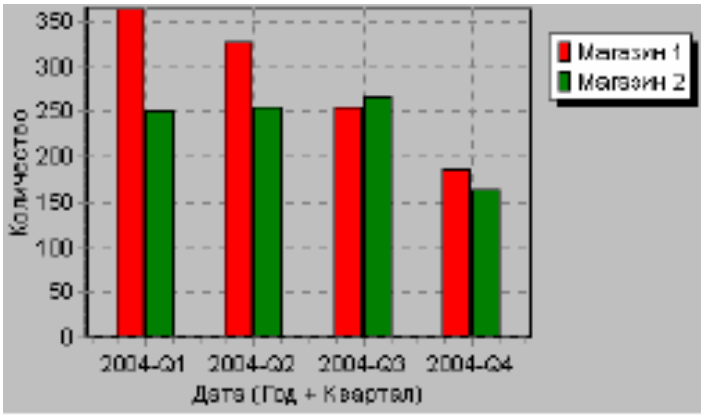


Рис. 8.15. Транспортування крос-діаграми.

Крос-діаграма будується по одному з фактів крос-таблиці. Вибрати факт, що цікавить, можна, натискує кнопку  на панелі інструментів.

При побудові діаграми вводяться обмеження числа серій і числа точок в кожній серії. Дане обмеження викликане, з одного боку, великими обчислювальними витратами при побудові діаграми, а з іншої - складністю сприйняття великих діаграм. Кнопка  на панелі інструментів матиме синій колір, якщо обмеження не перевищені, і червоний  інакше. Це попереджає про те, що на крос-діаграмі користувач бачить не всю інформацію. Клацання по кнопці виводить вікно «*Сведения об ограничениях*», в якому представлена інформація:

- скільки серій і точок, фактично відображається на крос-діаграмі;
- максимально можливе число серій і точок (за умовчанням 50 і 100 відповідно) для крос-діаграми;
- кількість серій і точок, мінімально необхідне для того, щоб діаграма відображалася повністю.

Таким чином, якщо фактична кількість серій і точок відповідає мінімально необхідному для повного відображення крос-діаграми, то

відображується вся інформація. Крос-діаграма володіє цікавою можливістю – побудовою тренду. У багатьох випадках тренд дозволяє побачити тенденції, які зазвичай приховані із-за великого розкиду значень, наявності відхилень, не типових для процесу, що відображається, і так далі. Лінія тренду отримується шляхом згладжування рядів даних, на основі яких побудована крос-діаграма, за допомогою виділення і відсікання великих відхилень, які в більшості випадків заважають оцінити загальний характер процесу. Кнопка  «*Тренд*» на панелі інструментів дозволяє включити відображення тренду. Як тільки режим відображення кнопки буде включений, на панелі інструментів вікна крос-діаграми стануть доступні налаштування «*Гладкость линии*» і «*Мера огрублення*». Гладкість лінії визначає міру згладжування вхідного ряду значень, а міра огрублення визначає «масштаб» деталей, що відсіваються. Чим вище значення цих налаштувань, тим більше гладкою буде лінія тренду. Комбінуючи ці налаштування можна добитися найкращого результату. Проте слід врахувати, що для процесів, які швидко змінюються, ці значення мають бути більше, а для повільних процесів - менше. Якщо задати їх дуже великими, то вхідний процес буде згладжений і огрублений в такій мірі, що буде втрачена і корисна інформація, а лінія тренду звиродніє в сходинку або пряму лінію.

8.3. Лабораторна робота №3 «Аналітичні рішення за допомогою нейронних мереж».

Завдання до лабораторної роботи.

1. Відповідно до заданого варіанту підготувати навчальну вибірку у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.

2. Провести нормалізацію полів, створення структури і навчання нейромережі в пакеті Deductor відповідно до методики, представленої в розділі «Порядок виконання роботи».

3. Отримати аналітичне рішення і провести декілька експериментів, використовуючи візуалізацію «Що - якщо» відповідно до методики, представленої в розділі «Порядок виконання роботи».

4. Виконати моделювання і економічну оцінку результатів шляхом використання діаграм відповідно до методики, представленої в розділі «Порядок виконання роботи».

6. Здійснити наступний імпорт даних з сховища: кількість відвантаженого товару в розрізі дат та товарів по вибраному Вами клієнтові, залишивши одну властивість товару (вибір властивості довільний). Виконання завдання здійснюється відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 4».

Варіанти завдань:

N варіанту	Процес
1	Діагностика захворювань
2	Діагностика комп’ютерів
3	Діагностика технічних систем
4	Діагностика комп’ютерних мереж
5	Прогнозування курсу валют
6	Прогнозування вартості нерухомості
7	Прогнозування курсу акцій
8	Оцінка реалізація медичних препаратів
9	Прогнозування рівня інфляції
10	Оцінка фінансового стану фірми
11	Оцінка кредитоспроможності фірми
12	Прогнозування рівня цін на продовольчі товари
13	Прогнозування рівня цін на промислові товари

Теоретична частина.

Нейронні мережі (НМ) є обчислювальними структурами, що моделюють прості біологічні процеси, аналогічні процесам, які відбуваються в

людському мозку. НМ – це розподілені і паралельні системи, здібні до адаптивного навчання шляхом реакції на позитивні і негативні дії. У основі побудови НМ лежить елементарний перетворювач, названий *штучним нейроном* або просто *нейроном* по аналогії з його біологічним прототипом.

Структуру нейромережі можна описати таким чином. Нейромережа складається з декількох шарів: вхідного, внутрішнього (прихованого) і вихідного шарів. Вхідний шар реалізує зв'язок з вхідними даними, вихідний – з вихідними. Внутрішніх шарів може бути від одного і більше. У кожному шарі міститься декілька нейронів. Між нейронами є зв'язки, названі *вагами* (рис.8.16).

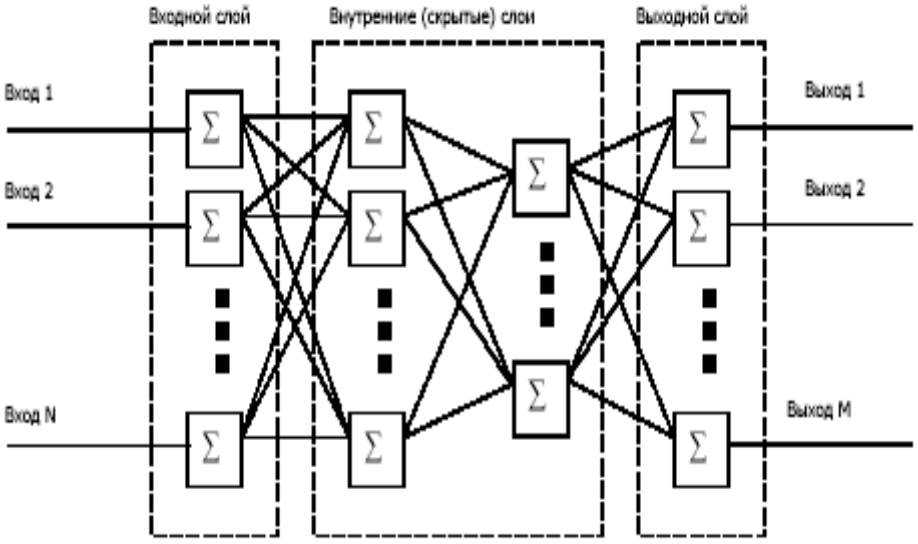


Рис. 8.16. Структура нейромережі.

Призначення і підготовка навчальної вибірки. Нейромережа здатна імітувати який-небудь процес. Будь-яка зміна входів нейромережі веде до зміни її виходів. Причому виходи нейромережі однозначно залежать від її входів. Але перш ніж використовувати нейромережу її необхідно навчити. Задача навчання тут рівносильна задачі апроксимації функції, тобто відновлення функції по окремо взятих її точках – таблично заданій функції. Таким чином, для навчання *потрібно підготувати таблицю* з вхідними значеннями і відповідними їм вихідними значеннями, тобто підготувати *навчальну вибірку*. По такій таблиці нейромережа сама знаходить залежності вихідних полів від вхідних. Далі ці

залежності можна використовувати, подаючи на вхід нейромережі деякі значення. На виході будуть відновлені залежні від них значення. Причому на вхід можна подавати значення, на яких нейромережа не навчалася.

Після навчання на вхід нейромережі необхідно подавати значення з діапазону, на якому вона навчалася. Наприклад, якщо при навчанні нейромережі на один з її входів подавалися значення від 0 до 100, то надалі слід на цей вхід подавати значення з діапазону від 0 до 100. Допускається подавати значення, які лежать поряд з діапазоном. На самому початку роботи з нейромережею потрібно визначитися, що є її входами, а що – виходами. Передбачається, що у нас вже є таблиця з навчальною вибіркою.

Нормалізація значень полів. Після того, як вказані вхідні і вихідні поля, слід нормалізувати дані в навчальній вибірці. Метою нормалізації значень полів є перетворення даних до вигляду, найбільш відповідного для обробки засобами Deductor Studio. Для таких обробників як нейронна мережа, дерево рішень, лінійна модель прогнозування дані, що поступають на вхід, повинні мати *числовий тип*, а їх значення мають бути розподілені в певному діапазоні.

Нормалізатор може перетворити дискретні дані до набору унікальних індексів або значень в діапазоні [0...1]. Для нейромережі доступні наступні види нормалізації полів.

1. Лінійна нормалізація. Використовується лише для безперервних числових полів. Дозволяє привести числа до діапазону [min…max], тобто мінімальному числу з вихідного діапазону відповідатиме min, а максимальному – max. Останні значення розподіляться між min та max.

Унікальні значення. Використовується для дискретних значень. Такими є рядки, числа або дати, задані дискретно. Аби привести безперервні числа в дискретні можна, наприклад, скористатися обробкою «квантування». Так слід поступати з величинами, для яких можна задати відношення порядку, тобто, якщо для двох будь-яких дискретних значень можна вказати, яке більше, а яке менше. Тоді всі значення необхідно розташувати в порядку зростання. Далі вони нумеруються по порядку і значення замінюються їх порядковим номером.

3. Бітова маска. Використовується для дискретних значень. Цей вигляд нормалізації слід використовувати для величин, які можна лише порівнювати на рівність або нерівність, але не можна сказати, яке більше, а яке менше. Всі значення замінюються порядковими номерами, а номер розглядається в двійковому вигляді або у вигляді маски з нулів і одиниць. Тоді кожна позиція маски розглядається як окреме поле, що містить нуль або одиницю. До такого поля можна застосувати лінійну нормалізацію, тобто замінити нуль на деяке мінімальне значення, а одиницю – на деяке максимальне значення. Після такої нормалізації на вхід нейромережі подаватиметься не одне це поле, а стільки полів, скільки розрядів в масці.

Існує налаштування нормалізації полів за умовчанням. Тобто, запустивши обробник, в даному випадку нейронну мережу, цей етап можна пропустити. В цьому випадку нормалізація буде виконана автоматично залежно від типів і видів полів:

1. вигляд дискретний – нормалізація списком унікальних значень;
2. вигляд безперервний
    - тип цілий або дата – лінійна нормалізація в діапазоні [-1..1];
    - тип вещественний - лінійна нормалізація в діапазоні [0..1].

Налаштування навчальної вибірки. Після нормалізації полів слід налаштувати навчальну вибірку. Навчальну вибірку розбивають на дві множини – *навчальну* і *тестову*.

Навчальна множина - включає записи (приклади), які використовуватимуться безпосередньо для навчання мережі, тобто міститимуть вхідні і бажані вихідні (цільові) значення.

Тестова множина - також включає записи (приклади), що містять вхідні і бажані вихідні (цільові) значення, але використовується не для навчання мережі, а для перевірки результатів навчання.

Розбивати навчальну вибірку на ці множини можна двома способами: або по порядку, або випадково. Якщо розбиття відбувається по порядку, то тестова множина вибирається або з початку, або з кінця навчальної вибірки.

Далі задаються параметри, що визначають структуру нейронної мережі, - кількість прихованих шарів і нейронів в них, а також активаційна функція нейронів.

Зауваження. До вибору кількості прихованих шарів і кількості нейронів для кожного прихованого шару потрібно підходити обережно. Хоча до цих пір не вироблені чіткі критерії вибору, дати деякі загальні рекомендації все ж можливо. Вважається, що задачу будь-якої складності можна вирішити за допомогою двошарової нейромережі, тому конфігурація з кількістю прихованих шарів, що перевищують 2, навряд чи виправдана. Для вирішення багатьох задач сповна підійде одношарова нейронна мережа. При виборі кількості нейронів слід керуватися наступним правилом: «*кількість зв'язків між нейронами має бути значно менше кількості прикладів в навчальній множині*». Кількість зв'язків розраховується як зв'язок кожного нейрона зі всіма нейронами сусідніх шарів, включаючи зв'язки на вхідному і вихідному шарах. Дуже велика кількість нейронів може привести до так званого «перенавчання» мережі, коли вона видає добрі результати на прикладах, що входять в навчальну вибірку, але практично не працює на інших прикладах.

Навчання нейромережі. Після налаштування конфігурації мережі слід вибрати алгоритм її навчання.

Метод зворотного поширення помилки – ітеративний градієнтний алгоритм навчання, який використовується з метою мінімізації середньоквадратичного відхилення поточних значень виходів мережі від потрібних. Однією з найважливіших властивостей алгоритму зворотного поширення помилки є *висока стійкість*, а отже, *надійність*. Хоча нейронні мережі, що використовують алгоритм зворотного поширення помилки, будучи потужним інструментом пошуку закономірностей, прогнозування і якісного аналізу, набули широкого поширення, їм властиві деякі недоліки. До них відноситься невисока швидкість збіжності (велике число необхідних ітерацій), що робить процес навчання дуже довгим і непридатним використання даного алгоритму для широкого круга задач, які вимагають швидкого рішення. Інші

алгоритми навчання НМ, хоча і працюють швидше, в більшості випадків, володіють меншою стійкістю.

Для алгоритму зворотного поширення помилки потрібно вказати два параметри:

- швидкість навчання - визначає величину кроку при ітераційній корекції вагів в нейронній мережі (рекомендується в інтервалі 0…1). При великій величині кроку, збіжність буде швидшою, але є небезпека «перестрибнути» через рішення. З іншого боку, при малій величині кроку, навчання зажадає дуже багатьох ітерацій. На практиці величина кроку береться пропорційній крутості схилу, так, що алгоритм сповільнюється поблизу мінімуму. Правильний вибір швидкості навчання залежить від конкретної задачі і зазвичай шукається дослідним шляхом;
- момент - задається в інтервалі 0…1. Рекомендується значення 0.9 ±0.1.

Метод Resilent Propogation (Rprop) - еластичне поширення. Алгоритм використовує так зване «навчання по епохах», коли корекція вагів відбувається після пред'явлення мережі всіх прикладів з навчальної вибірки. Перевага даного методу полягає в тому, що він забезпечує збіжність, а, отже, і навчання мережі в 4-5 разів швидше, ніж алгоритм зворотного поширення. Для алгоритму Resilient Propagation вказуються параметри:

- крок спуску - коефіцієнт збільшення швидкості навчання, який визначає крок збільшення швидкості навчання при недосягненні алгоритмом оптимального результату;
- крок підйому - коефіцієнт зменшення швидкості навчання. Задається крок зменшення швидкості навчання в разі пропуску алгоритмом оптимального результату.

Далі необхідно задати умови, при виконанні якого навчання буде припинено:

- вважати приклад розпізнаним, якщо помилка менша - критерієм останову в даному випадку є умова, що розузгодження між еталонним і реальним виходом мережі стає менше заданого значення;

- після досягнення епохи - установка даного режиму дозволяє задати число епох (циклів навчання) після досягнення якого навчання зупиняється незалежно від величини помилки. Якщо прапорець скинутий, то навчання продовжуватиметься, поки помилка не стане менше заданого значення, але при цьому є вірогідність зациклення, коли помилка ніколи не буде досягнута. Тому, все-таки, бажано встановити прапорець по «*Достижению епохи*», аби алгоритм був зупинений в будь-якому разі.

Тепер все готово до процесу навчання мережі. На початку всі ваги нейромережі ініціалізувалися випадковими значеннями. Після навчання ці ваги набувають певних значень. Навчання може з великою часткою вірогідності вважатися успішним, якщо відсоток розпізнаних прикладів на навчальній і тестовій множині досить великий (близький до 100%).

Порядок виконання роботи.

1. Створення навчальної вибірки. Розглянемо приклад побудови системи оцінки кредитоспроможності фізичної особи. Передбачимо, що експерти визначили основні чинники, що визначають кредитоспроможність. Ними виявилися: вік, освіта, площа квартири, наявність автомобіля, тривалість мешкання в даному регіоні. В організації була накопичена статистика повернень або не повернень взятих кредитів. Ця статистика представлена таблицею в MS Excel.

Сумма кредита	Возраст	Образование	Площадь квартиры	Автомобиль	Срок проживания	Давать кредит
7000	37	Специальное	37	отечественная	22	Да
7500	38	Среднее	29	импортная	12	Да
14500	60	Высшее	34	Нет автомобиля	30	Нет
15000	28	Специальное	14	отечественная	21	Да
32000	59	Специальное	53	отечественная	29	Да
11500	25	Специальное	28	отечественная	9	Да
5000	57	Специальное	18	отечественная	34	Да
61500	29	Высшее	26	Нет автомобиля	18	Нет
13500	37	Специальное	46	отечественная	28	Нет
25000	36	Специальное	20	Нет автомобиля	21	Нет
25500	68	Высшее	45	отечественная	30	Нет
...	...	...	...	...	...	...

Використовуючи «*Мастер импорта*» завантажимо файл даних з MS Excel в Deductor (рис. 8.17).

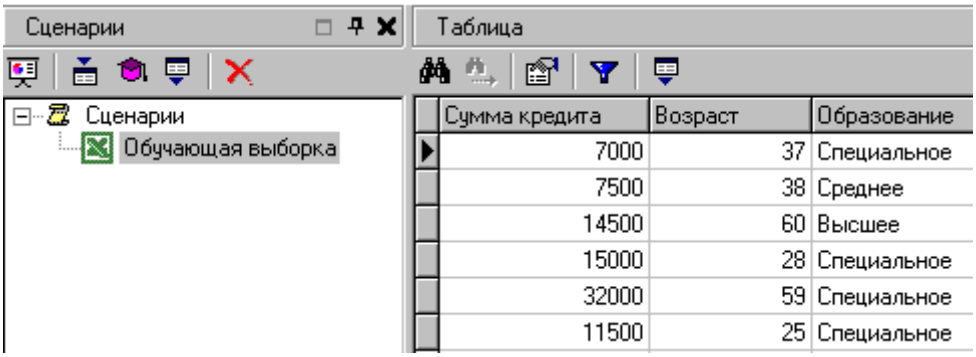


Рис. 8.17. Завантаження даних.

2. Нормалізація полів. Поля «Сума кредиту», «Вік», «Площа квартири» і «Тривалість мешкання» – безперервні значення, які перетворимо до інтервалу [–1…1]. «Освіта» представлена трьома унікальними значеннями, які можна порівнювати на більше або менше, а точніше краще або гірше. Тобто «Освіту» можна упорядкувати так: середнє, спеціальне, вище. Значення поля з наявністю автомобіля упорядкувати не можна. Його потрібно перетворити до бітової маски. Для кодування трьох значень потрібний два біта. Отже, це поле буде розбито на два (рис. 8.18).

Наличие автомобиля	Первый бит маски	Второй бит маски
Импортная	0	0
Отечественная	0	1
нет автомобиля	1	0

Рис. 8.18. Розбиття поля «Наявність автомобіля».

Скориставшись «*Мастером обработки*» (кнопка на панелі ) в 1-му вікні вибираємо спосіб обробки – «*Нейросеть*». Потім виконуємо налаштування призначень стовпців і в цьому ж вікні, ставимо відмітку в полі – «*Настройка нормализации в ручную*». У 4-му вікні майстра проводимо налаштування нормалізації стовпців відповідно до приведених вище міркувань.

3. Навчальну вибірку розіб'ємо на навчальну і тестову множину так, як програма пропонує це зробити за умовчанням, тобто випадкові 95 відсотків записів будуть в навчальній безлічі, останні 5 відсотків – в тестовому. (Крок 4 – Майстер обробки).

4. Конфігурація мережі. У вхідному шарі – 7 нейронів, тобто по одному нейрону на один вхід (у навчальній вибірці 6 стовпців, але стовпець «Автомобіль» представлений бітовою маскою з двох біт, для кожного з яких створений новий вхід). Зробимо один прихований шар з двома нейронами. У вихідному шарі буде один нейрон, на виході якого буде рішення про видачу кредиту (Крок 5 – Майстер обробки).

5. Виберемо алгоритм навчання мережі – Resilent Propagation з налаштуваннями за умовчанням. Умову закінчення навчання залишимо без зміни (Крок 6, 7, 8 – Майстер обробки).

6. Навчену таким чином нейромережу можна використовувати для ухвалення рішення про видачу кредиту фізичній особі. Це можна зробити, використовуючи аналіз «Що-якщо». Для його включення потрібно вибрати візуалізацію «Що-якщо». Тоді відкриється форма представлена на рис. 8.19 (Крок 9, 10 – Майстер обробки).

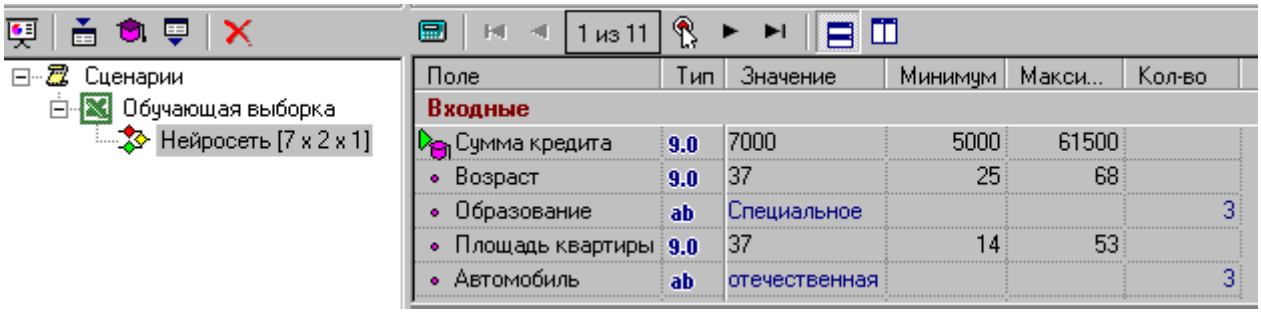


Рис. 8.19. Застосування візуалізатора «Що-якщо».

Після зміни в цій таблиці вхідних полів система сама приймає рішення про видачу кредиту і в полі «Давати кредит» проставляє або «Так», або «Ні». Стовпці «Мінімум» і «Максимум» визначають діапазон значень, на яких навчалася нейромережа. Слід дотримуватися цих обмежень, хоча і можливо взяти значення що трохи виходять за кордони діапазону.

7. Окрім такої таблиці аналіз «Що-якщо» містить діаграму, на якій відображується залежність вихідного поля від одного з вхідних полів при фіксованих значеннях останніх полів. Наприклад, потрібно взнати, на яку суму кредиту може розраховувати чоловік що володіє певними характеристиками. Це можна визначити по діаграмі (рис. 8.20).

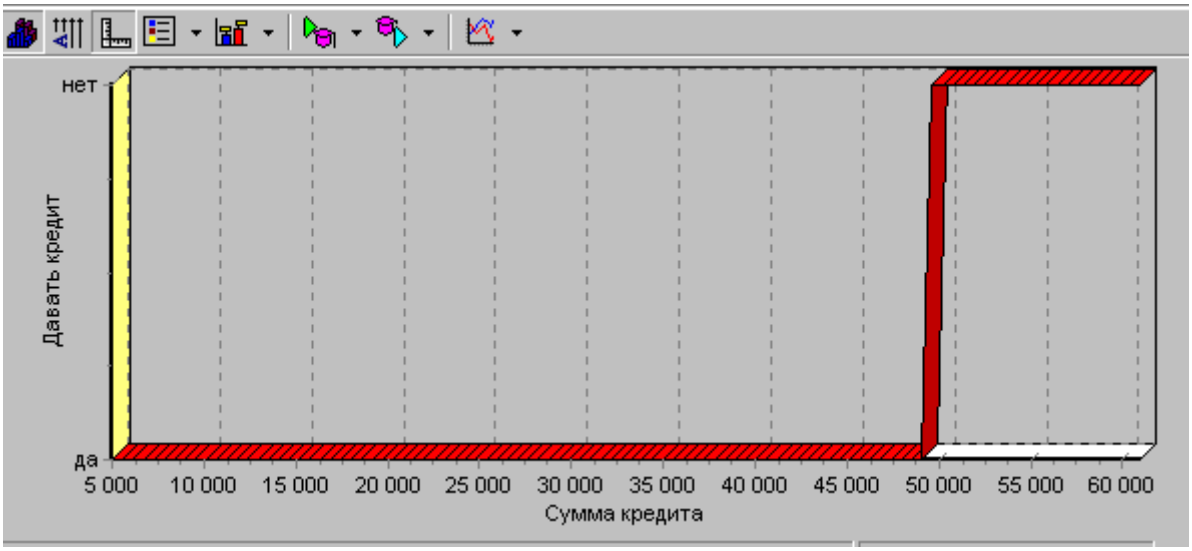


Рис. 8.20. Діаграма результатів аналізу.

8.4. Лабораторна робота №4 «Аналітичні рішення за допомогою дерева рішень».

Завдання до лабораторної роботи.

1. Відповідно до заданого варіанту підготувати навчальну вибірку у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.

2. Здійснити побудову дерева рішень і виконати його аналіз в пакеті Deductor відповідно до методики, представленої в розділі «Порядок виконання роботи».

3. Отримати правила рішень і провести декілька експериментів відповідно до методики, представленої в розділі «Порядок виконання роботи».

Варіанти завдань:

N варіанту	Процес
1	Діагностика захворювань
2	Діагностика комп’ютерів
3	Діагностика технічних систем
4	Діагностика комп’ютерних мереж
5	Прогнозування курсу валют
6	Прогнозування вартості нерухомості
7	Прогнозування курсу акцій
8	Реакція ринку на підвищення податків
9	Прогнозування рівня інфляції
10	Оцінка фінансового стану фірми
11	Оцінка кредитоспроможності фірми
12	Прогнозування рівня цін на продовольчі товари
13	Прогнозування рівня цін на промислові товари

Теоретична частина.

Дерева рішень (decision trees) є одним з найбільш популярних підходів до вирішення задач інтелектуального аналізу даних. Вони створюють ієрархічну структуру класифікуючих правил типа «якщо … то» (if-then), що має вигляд дерева. Аби прийняти рішення, до якого класу слідує віднести деякий об'єкт або ситуацію, потрібно відповісти на питання, що стоять у вузлах цього дерева, починаючи з його кореня. Питання мають вигляд «значення параметра А більше В?». Якщо відповідь позитивна, здійснюється перехід до правого вузла наступного рівня; потім знову слідує питання, пов'язане з відповідним вузлом і так далі. Наведений приклад ілюструє роботу так званих бінарних дерев рішень, в кожному вузлі яких, галуження виконується по двох напрямах (тобто на питання, задане у вузлі, є лише два варіанти відповідей, наприклад «Так» чи «Ні»). Проте, в загальному випадку, відповідей, а, отже, гілок, що виходять з вузла, може бути більше. Дерево рішень складається з вузлів – де виконується перевірка умови, і листя – кінцевих вузлів дерева, вказуючих на клас (вузлів рішення) (рис. 8.21).

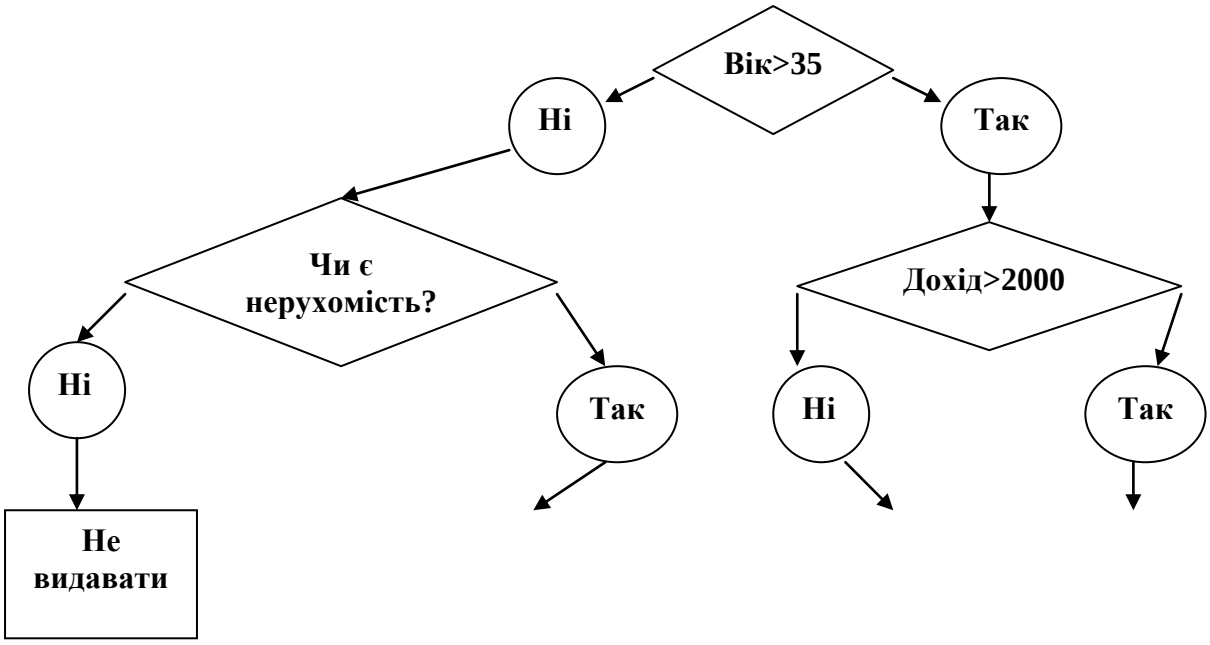


Рис. 8.21. Бінарне дерево рішень.

Сфера застосування дерев рішень в даний час вельми широка, але всі задачі, що вирішуються цим апаратом, можуть бути об'єднані в три класи.

1. Опис даних. Дерева рішень дозволяють зберігати інформацію про дані в компактній формі. Замість громіздких масивів даних можна зберігати дерево рішень, яке містить точний опис об'єктів.

2. Класифікація. Дерева рішень відмінно справляються із задачами класифікації, тобто віднесення об'єктів до одного із заздалегідь відомих класів.

3. Регресія. Якщо цільова змінна має безперервні значення, дерева рішень дозволяють встановити залежність цільової змінної від незалежних (вхідних) змінних. Наприклад, до цього класу відносяться задачі чисельного прогнозування (передбачення значень цільовій змінній).

Підготовка навчальної вибірки.

Для побудови дерева рішень готується навчальна вибірка так, як це описано для нейромережі. Різниця полягає в тому, що вихідне поле для дерева рішень може бути лише одне - *дискретне*.

Для полів, що подаються на входи і вихід дерева рішень, також задається нормалізація. Можна задати або лінійну нормалізацію, або нормалізацію унікальними значеннями. Налаштування навчальної вибірки виконується аналогічно, як і для нейромережі. Параметри навчання дерева рішень наступні:

- мінімальна кількість прикладів, при якій буде створений новий вузол. Задається мінімальна кількість прикладів, яка можлива у вузлі. Якщо прикладів, які потрапляють в даний вузол, буде менше заданого - вузол вважається листом (тобто подальше галуження припиняється). Чим більше цей параметр, тим менш гіллястим виходить дерево;
- будувати дерево з достовірнішими правилами в збиток складності. Включає спеціальний алгоритм, який, ускладнюючи структуру дерева, збільшує достовірність результатів класифікації. При цьому дерево виходить, як правило, більш гіллястим;
- рівень довіри, що використовується при відсіканні вузлів дерева. Значення цього параметра задається у відсотках і повинне лежати в межах від 0 до 100. Чим більше рівень довіри, тим більше гіллястим виходить дерево, і, відповідно, чим менше рівень довіри, тим більше вузлів буде відсічено при його побудові.

Якість побудованого дерева після навчання можна оцінити по декількох параметрах. По-перше, це число розпізнаних прикладів в навчальному і тестовому наборах даних. Чим вище це число, тим якісніше побудоване дерево. По-друге, це кількість вузлів в дереві. При дуже великому їх числі дерево стає важким для сприйняття. Це також означає дуже слабку залежність вихідного поля від вхідних полів.

Кожне правило характеризується *підтримкою* і *достовірністю*. Підтримка – загальна кількість прикладів класифікованих даним вузлом дерева. Достовірність – кількість правильно класифікованих даним вузлом прикладів.

Порядок виконання роботи.

1. Продовжимо приклад, розглянутий в попередній лабораторній роботі, присвяченій нейронним мережам. Всіх кредиторів можна розділити на два класи – кредитоспроможних і некредитоспроможних. Вочевидь, існують деякі правила віднесення кредиторів до того або іншого класу. Але при чималому числі їх характеристик майже неможливо побудувати ці правила. Це дозволяють зробити дерева рішень.

Підготуємо і завантажимо навчальну вибірку згідно варіанту за допомогою «*Мастера импорта*» і викличемо «*Мастер обработки*». У вікні, що з'явилося, виберемо режим – «*Дерево решений*». Кроки 1 – 4 «*Мастера обработки*» виконуються аналогічно попередній лабораторній роботі.

2. При побудові правил задамо мінімальну кількість прикладів, при якій буде створений новий вузол, рівним 1. Будуватимемо дерево з достовірнішими правилами в збиток складності (у вікні «*Параметры отсечения*» задамо 100) – крок 5 «*Мастера обработки*».

3. На кроці 6 запускаємо процес побудови дерева рішень і на кроці 7 вибираємо способи відображення даних – «*Дерево решений*» і «*Правила*».

4. Отримане дерево рішень містить 3 вузли і 7 правил (рис. 8.22).

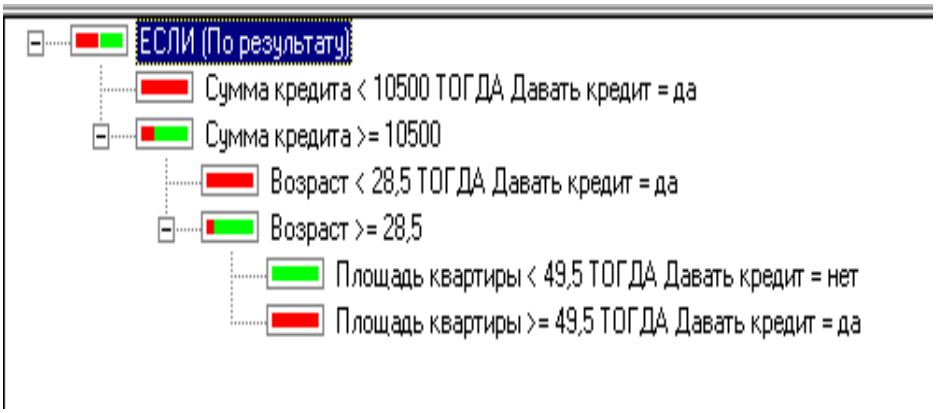


Рис. 8.22. Результат побудови дерева рішень в Deductor.

Таке дерево містить в собі правила, слідуючи яким можна віднести кредитора в одну з груп ризику і зробити висновок про видачу кредиту. Правила читаються

з вузлів, розташованих правіше. Побудовані правила можна також проглянути у вигляді списку правил (рис. 8.23).

Итого правил: 4

N	Условие	Следствие (Давать кредит)	Поддержка		Достоверность	
			%	Кол-во	%	Кол-во
1	Сумма кредита < 10500	да	30,00	3	100,00	3
2	Сумма кредита >= 10500 И Возраст < 28,5	да	10,00	1	100,00	1
3	Сумма кредита >= 10500 И Возраст >= 28,5 И Площадь квартиры < 49,5	нет	50,00	5	100,00	5
4	Сумма кредита >= 10500 И Возраст >= 28,5 И Площадь квартиры >= 49,5	да	10,00	1	100,00	1

Рис. 8.23. Список правил.

8.5. Лабораторна робота №5 «Аналітичні рішення на основі самоорганізуючих карт Кохонена».

Завдання до лабораторної роботи.

1. Відповідно до заданого варіанту підготувати необхідні дані у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.

2. Виконати завантаження даних з таблиць MS Excel за допомогою Майстра імпорту відповідно до методики, представленої в розділі «Порядок виконання роботи».

3. Отримати самоорганізуючі карти Кохонена відповідно до методики, представленої в розділі «Порядок виконання роботи».

4. Виконати економічний аналіз різних ситуацій, змінюючи основні дані на картах як у бік збільшення, так і у бік їх зменшення відповідно до методики, представленої в розділі «Порядок виконання роботи».

Варіанти завдань:

N варіанту	Галузь
1	Оцінка ефективності вкладень в рекламу
2	Оцінка ефективності вкладень в будівництво
3	Оцінка ефективності вкладень в курорти
4	Оцінка ефективності вкладень в промисловість
5	Аналіз ринку ЗМІ
6	Аналіз ринку нерухомістю
7	Аналіз ринку книжкової продукції
8	Аналіз ринку медичних препаратів
9	Аналіз ринку комп'ютерної техніки
10	Аналіз ринку техніки зв'язку
11	Створення профілів клієнтів (розділення на групи) по суспільних інтересах
12	Створення профілів клієнтів (розділення на групи) по спортивних інтересах
13	Створення профілів клієнтів (розділення на групи) по наукових інтересах

Теоретична частина.

Самоорганізуючі карти можуть використовуватися для вирішення таких задач як моделювання, прогнозування, пошук закономірностей у великих масивах даних, виявлення наборів незалежних ознак і стискування інформації.

Алгоритм функціонування самоорганізуючих карт (Self Organizing Maps - SOM) є одним з варіантів кластеризації багатовимірних векторів - алгоритм проектування із збереженням топологічної подібності. Прикладом таких алгоритмів може служити алгоритм k-найближчих середніх (c-means). *Важливою відмінністю алгоритму SOM є те, що в ньому всі нейрони (вузли, центри класів) впорядковані в деяку структуру (зазвичай двовимірну сітку).* При цьому, в ході навчання модифікується не лише нейрон-переможець (нейрон карти, який найбільшою мірою відповідає вектору входів і визначає до якого класу відноситься приклад), але і його сусіди, хоча і у меншій мірі. За рахунок цього SOM можна вважати одним з методів проектування багатовимірного простору в простір з нижчою розмірністю. При використанні

цього алгоритму, вектора, близькі у вихідному просторі, виявляються поруч і на отриманій карті.

SOM здійснює використання впорядкованої структури нейронів. Зазвичай використовуються одно- і двовимірні сітки. При цьому кожен нейрон є n-мірним вектором-стовпцем, де n визначається розмірністю вхідного простору (розмірністю вхідних векторів). Вживання одно- і двовимірних сіток пов'язане з тим, що виникають проблеми при відображенні просторових структур більшої розмірності (при цьому знову виникають проблеми з пониженням розмірності до двовимірної, уявної на моніторі). Зазвичай, нейрони розташовуються у вузлах двовимірної сітки з прямокутними або шестикутними чарунками. При цьому, як було сказане вище, нейрони також взаємодіють один з одним. Величина цієї взаємодії визначається відстанню між нейронами на карті. При реалізації алгоритму SOM заздалегідь задається конфігурація сітки (прямокутна або шестикутна), а також кількість нейронів в мережі. Деякі джерела рекомендують використовувати максимально можливу кількість нейронів в карті. При цьому початковий радіус навчання (neighborhood в англомовній літературі) в значній мірі впливає на здатність узагальнення за допомогою отриманої карти. У разі, коли кількість вузлів карти перевищує кількість прикладів в навчальній вибірці, успіх використання алгоритму у великій мірі залежить від відповідного вибору початкового радіусу навчання. Проте, у разі, коли розмір карти складає десятки тисяч нейронів, час, потрібний на навчання карти, зазвичай буває дуже велике для вирішення практичних задач. Таким чином, необхідно досягати допустимий компроміс при виборі кількості вузлів.

Підготовка навчальної вибірки. *Відмінність* в підготовці навчальної вибірки в цьому алгоритмі полягає в тому, що *вихідні поля в такій вибірці можуть бути відсутніми зовсім*. Навіть якщо в навчальній вибірці будуть присутні вихідні поля, вони не братимуть участь при навчанні нейромережі. Проте вони братимуть участь при відображенні карт. Нормалізація полів тут

така ж, як для дерев рішень. Навчальна вибірка налаштовується так само, як і для нейромережі і дерева рішень.

Навчання. Перед початком навчання карти необхідне проініціалізувати вагові коефіцієнти нейронів. Вдало вибраний спосіб ініціалізації може істотно прискорити навчання і привести до здобуття якісніших результатів. Існують три способи ініціації початкових вагів:

- ініціалізація випадковими значеннями, коли всім вагам даються малі випадкові величини;
- ініціалізація прикладами, коли в якості початкових значень задаються значення випадково вибраних прикладів з навчальної вибірки;
- лінійна ініціалізація. В цьому випадку ваги ініціюються значеннями векторів, лінійно впорядкованих уздовж лінійного підпростору, який проходить між двома головними власними векторами вхідного набору даних.

Візуалізація. Отриману в результаті навчання карту можна представити у вигляді прошаркового пирога, кожен шар якого є розфарбовуванням, породженим однією з компонент вхідних даних. Отриманий набір розфарбовувань може використовуватися для аналізу закономірностей, які є між компонентами набору даних. Після формування карти, отримується набір вузлів, який можна відображати у вигляді двовимірної картинки. При цьому кожному вузлу карти можна поставити у відповідність ділянку на малюнку (чотири або шестикутну), координати якої визначаються координатами відповідного вузла в гратах. Тепер для візуалізації залишається лише визначити колір чарунок цієї картинки. Для цього і використовуються значення компонент. Найпростіший варіант - використання градацій сірого. В цьому випадку чарунки, відповідні вузлам карти, в які попали елементи з мінімальними значеннями компонента або не попало взагалі жодного запису, будуть змальовані чорним кольором, а чарунки, в які попали записи з максимальними значеннями такого компонента, відповідатимуть чарункам білого кольору. В принципі можна використовувати будь-яку градієнтну палітру для розфарбовування. Отримані розфарбовування в сукупності

утворюють атлас, що відображає розташування компонент, зв'язки між ними, а також відносне розташування різних значень компонент.

Порядок виконання роботи.

1. Продовжимо приклад з кредитуванням фізичних осіб. За допомогою карт Кохонена можна розглянути залежності між різними характеристиками кредиторів і виділити сегменти кредиторів, об'єднавши їх по схожих ознаках. Навчальна вибірка буде така ж, що і для нейромереж і дерев рішень. Але поле «Автомобіль» використовувати не будемо. Це недоцільно, оскільки дані, що подаються на вхід карт Кохонена мають бути такими, аби між ними можна було обчислити відстань або була можливість розташувати їх в порядку зростання або убування. Значення ж «вітчизняна», «імпортна» і «немає автомобіля» порівнювати між собою не можна (рис. 8.24).

Сумма кредита	Возраст	Образование	Площадь	Срок проживания	Давать кредит
7000	37	Специальное	37	22	да
7500	38	Среднее	29	12	да
14500	60	Высшее	34	30	нет
15000	28	Специальное	14	21	да
32000	59	Специальное	53	29	да
11500	25	Специальное	28	9	да
5000	57	Специальное	18	34	да
61500	29	Высшее	26	18	нет
13500	37	Специальное	46	28	нет
25000	36	Специальное	20	21	нет
25500	68	Высшее	45	30	нет

Рис. 8.24. Навчальна вибірка для отримання карт Кохонена.

2. Застосуємо «*Мастер обработки*» і в першому вікні вибираємо «*Карта Кохонена*». Нормалізація полів буде така ж, як і для дерев рішень. Розбиття навчальної вибірки на дві множини залишимо за умовчанням, так само, як і останні налаштування.

3. Кроки 5 – 8 виконуються за умовчанням. На кроці 9 вибираємо режим «*Карты Кохонена*». Крок 10 дозволяє вибрати в лівому вікні допустимі відображення карт. Поставимо відмітки у вхідних і вихідному стовпцях.

4. Результати роботи алгоритму відображуються на картах. Кожному вхідному полю відповідає своя карта. Наприклад, в один і той же кластер були згруповані молоді кредитори з маленьким терміном мешкання в даній місцевості, спеціальною або середньою освітою, середньою житлоплощею, що беруть високі суми кредиту (рис. 8.25).

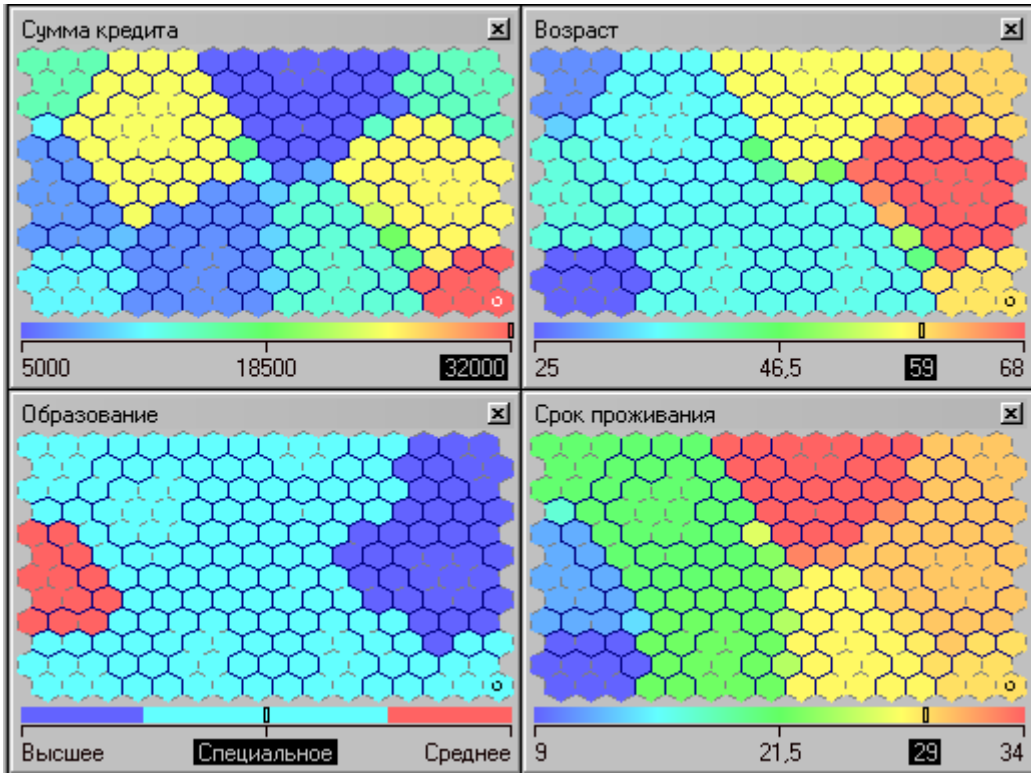


Рис. 8.25. Отримана карта Кохонена.

8.6. Лабораторна робота №6 «Аналітичні рішення на основі асоціативних правил».

Завдання до лабораторної роботи.

1. Відповідно до заданого варіанту підготувати необхідні дані у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.

2. Виконати завантаження даних з таблиць MS Excel за допомогою Майстра імпорту відповідно до методики, представленої в розділі «Порядок виконання роботи».

3. Отримати аналітичне рішення на основі асоціативних правил відповідно до методики, представленої в розділі «Порядок виконання роботи».

4. Виконати економічний аналіз різних ситуацій, застосовуючи основні візуалізатора, відповідно до методики, представленої в розділі «Порядок виконання роботи».

Варіанти завдань:

N варіанту	Галузь
1	Продажі продовольчих товарів (шаблон покупок)
2	Продажі промислових товарів (шаблон покупок)
3	Продажі будівельних товарів (шаблон покупок)
4	Продажі комп’ютерної техніки (шаблон покупок)
5	Зручне розміщення товарів на прилавках супермаркетів
6	Зручне розміщення товарів на прилавках промислових магазинів
7	Зручне розміщення товарів на вітринах книжкових електронних магазинів
8	Зручне розміщення товарів на вітринах електронних магазинів загального призначення
9	Зручне розміщення товарів на вітринах комп’ютерних електронних магазинів
10	Стимулювання продажів промислових товарів
11	Стимулювання продажів продукції ЗМІ
12	Стимулювання продажів продовольчих товарів
13	Стимулювання продажів комп’ютерної техніки

Теоретична частина.

Асоціативні правила дозволяють знаходити закономірності між зв'язаними подіями. Прикладом такого правила, служить твердження, що покупець, який купує «Хліб», придбає і «Молоко» з вірогідністю 75%. Вперше

ця задача була запропонована для пошуку асоціативних правил при знаходженні типових шаблонів покупок, що здійснюються в супермаркетах, тому інколи її ще називають *аналізом ринкової корзини* (market basket analysis).

Транзакція – це множина подій, відбувшихся одночасно. Хай є база даних, що складається з купівельних транзакцій. Кожна транзакція – це набір товарів, куплених покупцем за один візит. Таку транзакцію ще називають *ринковою корзиною*. Після визначення поняття транзакція, можна перейти до визначення поняття асоціативного правила.

Хай є список транзакцій. Необхідно знайти закономірності між цими подіями. Як в умові, так і внаслідку правила повинні знаходитися елементи транзакцій. Хай $I = \{i_1, i_2, ..., i_n\}$ – множина елементів, що входять в транзакції. *Асоціативним правилом називається імплікація* $X \Rightarrow Y$, *де* $X \subset I, Y \subset I$ *та* $X \cap Y = \varnothing$. Правило $X \Rightarrow Y$ має *підтримку* s (support), якщо s% транзакцій з D, містять $X \cup Y$, $\sup p(X \Rightarrow Y) = \sup p(X \cup Y)$. *Достовірність* правила показує, яка вірогідність того, що з X виходить Y. Правило $X \Rightarrow Y$ справедливо з *достовірністю* (confidence) c, якщо c% транзакцій з D, що містять X, також містять Y, $conf\ (X \Rightarrow Y) = \sup p(X \cup Y) / \sup p(X)$.

Покажемо це на конкретному прикладі: «75% транзакцій, що містять хліб, також містять молоко. 3% від загального числа всіх транзакцій містять оба товари». 75% – це достовірність (confidence) правила, 3% це підтримка (support), або «Якщо «Хліб», то «Молоко»» з вірогідністю 75%. Іншими словами, метою аналізу є встановлення наступних залежностей: якщо в транзакції зустрівся деякий набір елементів X, то на підставі цього можна зробити висновок про те, що інший набір елементів Y також же повинен з'явитися в цій транзакції. Встановлення таких залежностей дає нам можливість знаходити дуже прості і інтуїтивно зрозумілі правила.

Алгоритми пошуку асоціативних правил призначені для знаходження всіх правил $X \Rightarrow Y$, причому підтримка і достовірність цих правил мають бути вище за деякі наперед задані пороги, названі відповідно *мінімальною підтримкою* (minsupport) і *мінімальною достовірністю* (minconfidence).

Значення для цих параметрів вибираються так, щоб обмежити кількість знайдених правил. Якщо підтримка має велике значення, то алгоритми знаходитимуть правила, добре відомі аналітикам або настільки очевидні, що немає жодного сенсу проводити такий аналіз. З іншого боку, низьке значення підтримки веде до генерації величезної кількості правив, що, звичайно, вимагає істотних обчислювальних ресурсів. Проте, більшість цікавих правил знаходяться саме при низькому значенні порогу підтримки. Хоча дуже низьке значення підтримки веде до генерації статистично необґрунтованих правил.

Пошук асоціативних правил зовсім не тривіальна задача, як може здатися на перший погляд. Одна з проблем – алгоритмічна складність при знаходженні часто зустрічаючих наборів елементів, оскільки із зростанням числа елементів експоненціально зростає число потенційних наборів елементів. Звичайні асоціативні правила – це правила, в яких як в умові, так і в слідстві присутні лише елементи транзакцій і при обчисленні яких використовується лише інформація про те, чи присутній елемент в транзакції чи ні. Фактично всі наведені вище приклади відносяться до звичайних асоціативних правил. Для пошуку звичайних асоціативних правил в програмі служить технологія «*Ассоциативные правила*».

Налаштування. По-перше необхідний вказати, що є ідентифікатором (ID) транзакції, а що елементом транзакції. Наприклад, ідентифікатор транзакції – це номер чека або код накладної. А елемент – це найменування товару в чеці. Потім слідує налаштування параметрів пошуку правил. Застосовуються слідуючи параметри:

- мінімальна і максимальна підтримка. Асоціативні правила шукаються лише в деякій множині всіх транзакцій. Для того, щоб транзакція увійшла до цієї множини вона повинна зустрітися у вхідній вибірці декілька разів, більше мінімальної підтримки і менше максимальною. Наприклад, мінімальна підтримка дорівнює 1%, а максимальна – 20%. Кількість елементів «Хліб» і «Молоко» стовпця «Товар» з однаковим значенням стовпця «Номер чека»

зустрічаються в 5% всіх транзакцій (номерів чека). Тоді ці два рядки увійдуть до шуканої множини;

- мінімальна і максимальна достовірність. Це процентне відношення кількості транзакцій, що містять всі елементи, які входять в правило, до кількості транзакцій, що містять елементи, які входять в умову. Якщо транзакція – це замовлення, а елемент – товар, то достовірність характеризує, наскільки часто купуються товари, що входять в слідство, якщо замовлення містить товари, що увійшли до всього правила.

Порядок виконання роботи.

1. Розглянемо механізм пошуку асоціативних правил на прикладі даних про продажі товарів в деякій торгівельній точці. Дані знаходяться у файлі «Supermarket.xls», підготовленому за допомогою MS Excel (*НЕ МЕНШЕ 30 ТРАНЗАКЦІЙ*) (рис. 8.26). У таблиці представлена інформація по покупках продуктів декількох груп. Вона має всього два поля «НОМЕР ЧЕКА» і «ТОВАР». Необхідно вирішити задачу аналізу споживчої корзини з метою подальшого вживання результатів для стимулювання продажів.

Номер чека	Товар
160698	КЕТЧУПЫ, СОУСЫ, АДЖИКА
160698	МАКАРОННЫЕ ИЗДЕЛИЯ
160698	ЧАЙ
160747	МАКАРОННЫЕ ИЗДЕЛИЯ
160747	МЕД
160747	ЧАЙ
161217	КЕТЧУПЫ, СОУСЫ, АДЖИКА
161217	МАКАРОННЫЕ ИЗДЕЛИЯ
161217	СЫРЫ

Рис. 8.26. Дані файлу «Supermarket.xls».

2. За допомогою «*Мастера импорта*» імпортуємо дані з файлу в пакет Deductor і проглянемо їх у вигляді таблиці.

3. Для пошуку асоціативних правил запустимо майстер обробки. У ньому виберемо тип обробки «*Ассоциативные правила*». На другому кроці майстра необхідно вказати, який стовпець є *ідентифікатором* транзакції («Номер чека»), а який є *елементом* транзакції («Товар»).

4. Наступний крок дозволяє налаштувати параметри побудови асоціативних правил: мінімальну і максимальну підтримку, мінімальну і максимальну достовірність, а також максимальну потужність множини. Виходячи з характеру наявних даних, слід вказати кордони підтримки – 13% і 80%, та достовірності 60% і 90%.

5. Наступний крок дозволяє запустити процес пошуку асоціативних правил. На екрані відображається інформація про кількість множин, кількість знайдених правил, а також гістограма розподілу знайденої множини, що часто зустрічається по потужності.

6. Після завершення процесу пошуку отримані результати можна проглянути, використовуючи спеціальну візуалізацію: «Популярні набори», «Правила», «Дерево правил», «Що-якщо».

«Популярні набори» - це множина, що складається з одного і більше елементи, які найчастіше зустрічаються в транзакціях одночасно. На скільки часто зустрічається множина у вхідному наборі транзакцій можна судити по підтримці. Дана візуалізація відображує множину у вигляді списку (рис. 8.27).

N	Множество	Поддержка	
		%	Кол-во
1	КЕТЧУПЫ, СОУСЫ, АДЖИКА	66,67	2
5	КЕТЧУПЫ, СОУСЫ, АДЖИКА И СЫРЫ	33,33	1
6	КЕТЧУПЫ, СОУСЫ, АДЖИКА И ЧАЙ	33,33	1
2	МЕД	33,33	1
7	МЕД И ЧАЙ	33,33	1
3	СЫРЫ	33,33	1
4	ЧАЙ	66,67	2

Рис. 8.27. Візуалізатор «Популярні набори».

Само назва візуалізації говорить про те, як застосувати результати на практиці. Набори товарів, що вийшли, найчастіше купують в даній торгівельній точці, отже можна приймати рішення про постачання товарів, їх розміщенні і так далі.

Візуалізація «Правила» відображає асоціативні правила у вигляді списку правил. Цей список представлений таблицею із стовпцями: «номер правила», «умова», «слідство», «підтримка %», «підтримка, кількість», «достовірність» (рис. 8.28).

N	Условие	Следствие	Поддержка		Достоверность, %
			%	Кол-во	
1	ВАФЛИ	СУХАРИ	22,73	10	71,43
2	СУХАРИ	ВАФЛИ	22,73	10	71,43
3	КЕТЧУПЫ, СОУСЫ, АДЖИКА	МАКАРОННЫЕ ИЗДЕЛИЯ	45,45	20	86,96
4	МАКАРОННЫЕ ИЗДЕЛИЯ	КЕТЧУПЫ, СОУСЫ, АДЖИКА	45,45	20	83,33
5	МЕД	ЧАЙ	40,91	18	81,82
6	СЫРЫ	ЧАЙ	29,55	13	68,42
7	ВАФЛИ И СУХАРИ	ЧАЙ	20,45	9	90,00
8	ВАФЛИ И ЧАЙ	СУХАРИ	20,45	9	69,23

Рис. 8.28. Візуалізатор «Правила».

Таким чином, експертові надається набір правил, які описують поведінку покупців. Наприклад, якщо покупець купив вафлі, то він з вірогідністю 71% також купить і сухарі.

Візуалізація «Дерево правил» - це завжди дворівневе дерево. Воно може бути побудоване або по умові, або по слідству. При побудові дерева правил по умові, на першому (верхньому) рівні знаходяться вузли з умовами, а на другому рівні - вузли із слідством. Другий варіант дерева правил - дерево, побудоване по слідству. Тут на першому рівні розташовуються вузли із слідством.

Праворуч від дерева знаходиться список правил, побудований по вибраному вузлу дерева. Для кожного правила відображуються підтримка і достовірність. Якщо дерево побудоване по умові, то зверху списку відображується умова правила, а список складається з його слідств. Тоді

правила відповідають на питання, що буде за такої умови. Якщо ж дерево побудоване по слідству, то зверху списку відображується слідство правила, а список складається з його умов. Ці правила відповідають на питання, що потрібне, аби було задане слідство. Дана візуалізація відображає ті ж самі правила, що і попередня, але в зручнішій для аналізу формі (рис. 8.29).

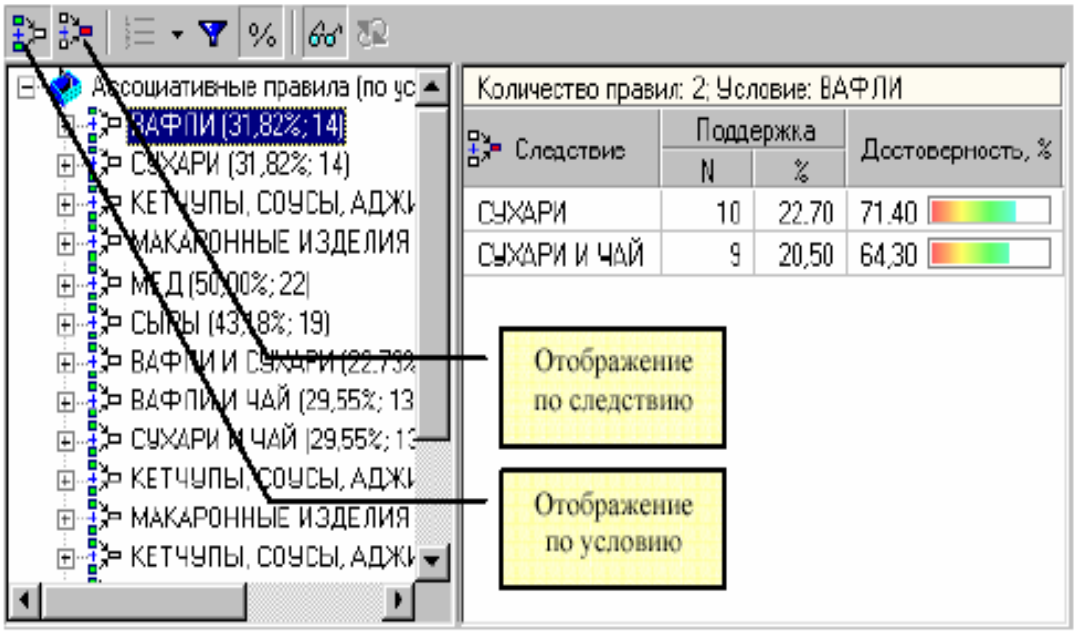


Рис. 8.29. Візуалізатор «Дерево правил».

В даному випадку правила відображують по умові. Тоді результат, що відображається в даний момент, можна інтерпретувати як 2 правила:

- Якщо покупець придбав вафлі, то він з вірогідністю 71% також придбає сухарі.
- Якщо покупець придбав вафлі, то він з вірогідністю 64% також придбає, сухарі і чай.

Аналогічно інтерпретуються і інші правила.

Аналіз "Що-якщо" в асоціативних правилах дозволяє відповісти на питання: що отримаємо як слідство, якщо виберемо дані умови? Наприклад, які товари купуються спільно з вибраними товарами. У вікні зліва розташований список всіх елементів транзакцій. Праворуч від кожного елементу вказана підтримка – скільки разів даний елемент зустрічається в транзакціях. У правому

верхньому кутку розташований список елементів, що входять в умову. Це, наприклад, список товарів, які придбав покупець. Для них потрібно знайти слідство. Наприклад, товари, що вживаються спільно з ними. Аби запропонувати людині те, що він можливо забув купити. У правому нижньому кутку розташований список слідств. Праворуч від елементів списку відображається підтримка і достовірність.

Хай необхідно проаналізувати, що, можливо, забув покупець придбати, якщо він вже купив вафлі і мед? Для цього необхідно додати в список умов ці товари (наприклад, за допомогою подвійного клацання миші) і потім натискувати на кнопку «*Вычислить правила*». При цьому в списку слідств з'являться товари, що спільно купуються з даними. В нашому випадку з'являться «СУХАРІ», «ЧАЙ», «СУХАРІ І ЧАЙ». Тобто можливо, покупець забув придбати сухарі або чай або і те і інше (рис. 8.30).

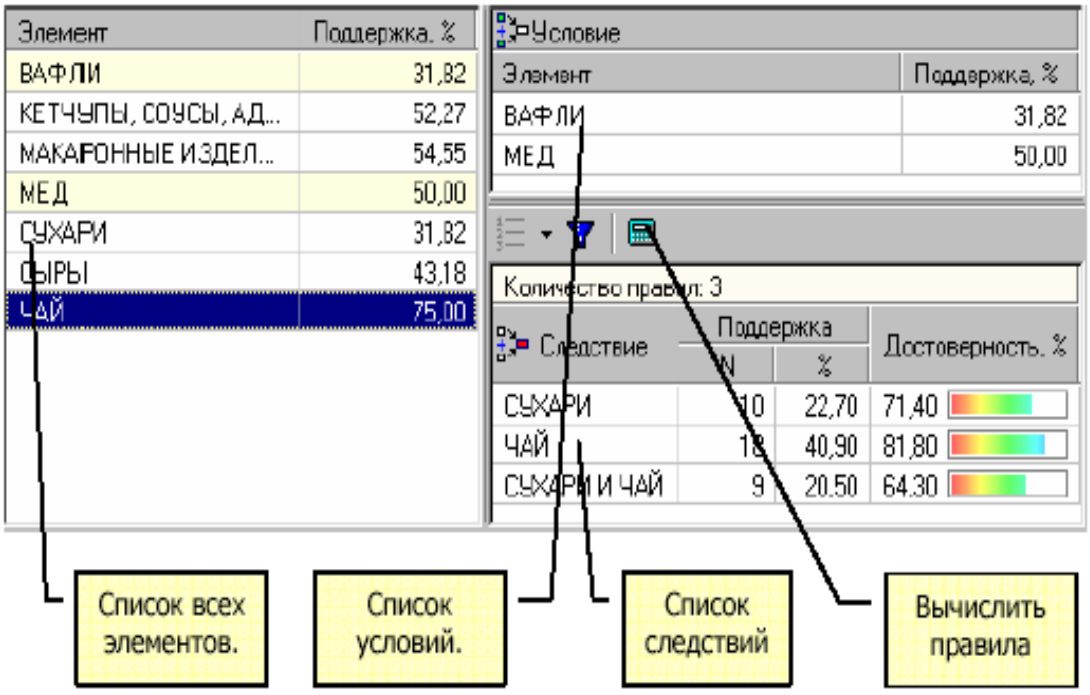


Рис. 8.30. Візуалізатор «Що-якщо».

8.7. Лабораторна робота №7 «Аналітичні рішення на основі трансформації даних і прогнозування за допомогою лінійної регресії».

Завдання до лабораторної роботи.

1. Відповідно до заданого варіанту підготувати необхідні дані у вигляді таблиць MS Excel і зберегти їх як персональні файли. Для підготовки даних використовувати тематичні сайти Інтернет, результати проходження практик, довідники і каталоги.

2. Здійснити трансформацію даних шляхом:

- розбиття даних на групи відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 1»;
- розбиття дати (по тижнях) відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 2»;
- квантування по інтервалах відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 3»;
- отримання необхідних розрахунків відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 4»;
- групування даних відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 5»;
- перетворення даних до ковзаючого вікна відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 6».

3. Отримати прогноз за допомогою лінійного регресійного аналіза відповідно до методики, представленої в розділі «Порядок виконання роботи. Частина 7».

4. Виконати економічний аналіз різних ситуацій, застосовуючи основні візуалізатора, відповідно до методики, представленої в розділі «Порядок виконання роботи».

Варіанти завдань:

N варіанту	Галузь
1	Продажі продовольчих товарів (шаблон покупок)
2	Продажі промислових товарів (шаблон покупок)
3	Продажі будівельних товарів (шаблон покупок)
4	Продажі комп'ютерної техніки (шаблон покупок)
5	Зручне розміщення товарів на прилавках супермаркетів
6	Зручне розміщення товарів на прилавках промислових магазинів
7	Зручне розміщення товарів на вітринах книжкових електронних магазинів
8	Зручне розміщення товарів на вітринах електронних магазинів загального призначення
9	Зручне розміщення товарів на вітринах комп'ютерних електронних магазинів
10	Стимулювання продажів промислових товарів
11	Стимулювання продажів продукції ЗМІ
12	Стимулювання продажів продовольчих товарів
13	Стимулювання продажів комп'ютерної техніки

Порядок виконання роботи.

ЧАСТИНА 1. Розбиття даних на групи.

Часто для проведення аналізу або побудови моделі прогнозу доводиться розбивати дані на групи, виходячи з певних критеріїв. У першому випадку така необхідність виникає, якщо аналітик бажає проглянути, наприклад інформацію не по всій сукупності даних, а по певних групах (наприклад, яку суму кредиту беруть на ті або інші цілі, або кредитори того або іншого віку). У другому випадку (прогнозування) аналітикові необхідно враховувати той факт, що певні групи (в даному випадку групи кредиторів) поводяться по різному, і що модель прогнозу, побудована на всіх даних не враховуватиме нюансів, що виникають в цих групах. Тобто краще побудувати декілька моделей прогнозу, наприклад,

залежно від сумової групи кредиту, і розробляти прогноз на них, ніж побудувати одну модель прогнозу. Виходячи з цього, в Deductor Studio надається широкий набір інструментів, які дозволяють тим або іншим способом розбивати вихідні дані на групи, групувати будь-яким способом всілякі показники і т. п.

1. Розглянемо розбиття даних на групи на прикладі даних по ризиках кредитування фізичних осіб, підготовлених в MS Excel. Стовпці, що цікавлять нас: «СУМА КРЕДИТУ», «ДАТА КРЕДИТУВАННЯ», «МЕТА КРЕДИТУВАННЯ» і «ВІК» (рис. 8.31).

Сумма кредита	Стоимость кре	Срок кредита	Дата кредитов	Цель кредитования
7000	1400	6	01.01.03	Иное
7500	1500	6	01.01.03	Иное
14500	2900	12	01.01.03	Покупка товара
15000	3000	6	01.01.03	Покупка товара
32000	6400	12	01.01.03	Иное
11500	2300	6	01.01.03	Турпоездки, развлечения и т.п.
5000	1000	6	01.01.03	Покупка и ремонт недвижимости
61500	12300	30	01.01.03	Покупка товара
13500	2700	12	01.01.03	Оплата услуг (мед., юрид. и т.п.)
25000	5000	18	01.01.03	Покупка товара
25500	5100	24	01.01.03	Покупка товара
9500	1900	6	01.01.03	Покупка товара
53000	10600	24	01.01.03	Иное
27500	5500	18	02.01.03	Покупка товара
4000	800	6	02.01.03	Оплата услуг (мед., юрид. и т.п.)

Рис. 8.31. Вхідні дані проекту.

2. Після імпорту даних з табличного файлу найбільш інформативно проглянути дані можна за допомогою візуалізації «*Куб*», вибравши в якості вимірів стовпці «ВІК» і «МЕТА КРЕДИТУВАННЯ», а в якості факту – стовпець «СУМА КРЕДИТУ». Останні стовпці встановити як непридатні (рис. 8.32).

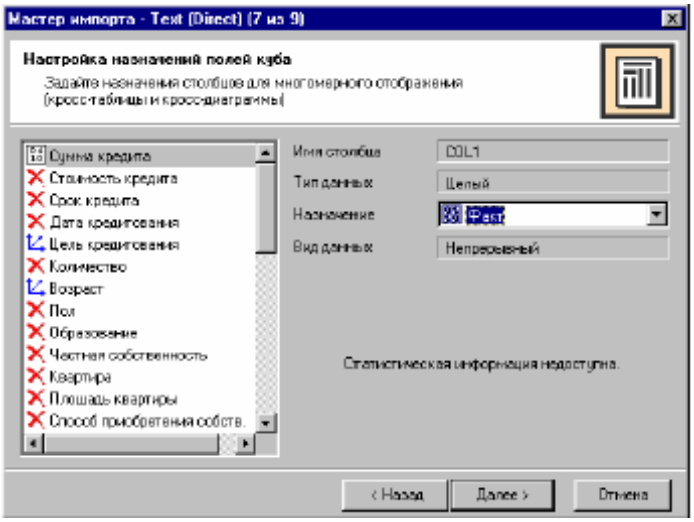


Рис. 8.32. Налаштування візуалізатора «Куб».

3. На наступному кроці налаштування куба слід вказати вимір «МЕТА КРЕДИТУВАННЯ» як вимір в термінах, а вимір «ВІК» як вимір в стовпцях, перетягнувши їх за допомогою миші у відповідні вікна з області доступних вимірів (рис. 8.33).

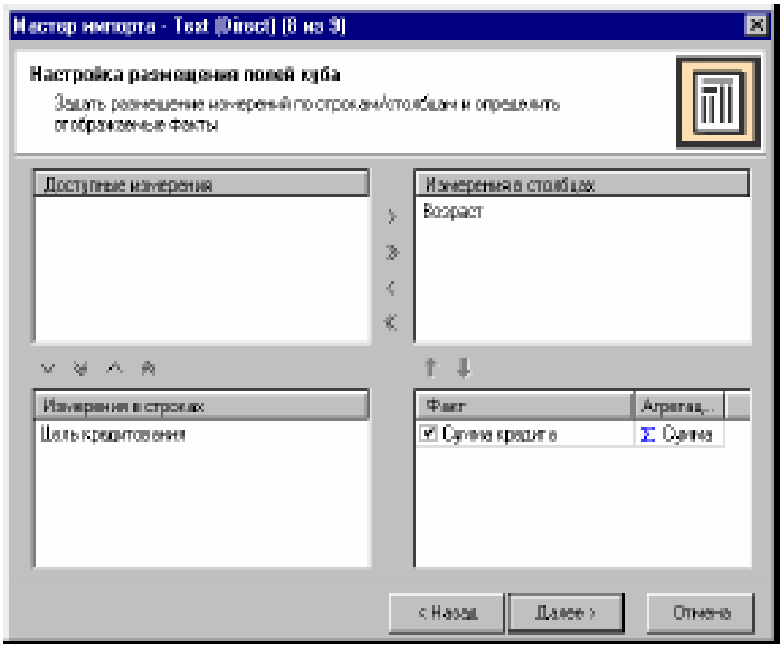


Рис. 8.33. Налаштування полів візуалізатора «Куб».

4. В результаті, на крос-діаграмі (одна із закладок візуалізації «Куб») можна проглянути вихідні дані (рис. 8.34).

	Возраст		
Цель кредитования	19	20	21
Иное	50 000,00	17 000,00	8 500,00
Оплата за образование		17 500,00	29 500,00
Оплата услуг (мед., юрид. и т.п.)			
Покупка и ремонт недвижимости	78 000,00		13 000,00
Покупка товара	46 500,00	73 500,00	76 500,00
Турпоездки, развлечения и т.п.		30 500,00	
Итого	174 500,00	138 500,00	127 500,00

Рис. 8.34. Результати розбиття даних на групи.

ЧАСТИНА 2. Розбиття дати (по тижнях).

Розбиття дати служить для аналізу всіляких показників за певний період (день, тиждень, місяць, квартал, рік). Суть розбиття полягає в тому, що на основі стовпця з інформацією про дату формується інший стовпець, в якому вказується, до якого заданого інтервалу часу належить рядок даних. Тип інтервалу задається аналітиком, виходячи з того, що він хоче отримати – дані за рік, квартал, місяць, тиждень, день або відразу по всіх інтервалах.

1. Хай нам необхідно отримати дані по сумах взятих кредитів по тижнях (у раніше створеному файлі міститься інформація за перші два тижні 2010 року). Для цього в майстрові обробки «*Дата и Время*» на другому кроці виберемо як використовуване поле «ДАТА КРЕДИТУВАННЯ», а в таблиці налаштувань, що з'явилася після цього, виберемо значення «*Используемое*» в стовпці «*Строка*» напроти рядка «*Год + Неделя*» (рис. 8.35).

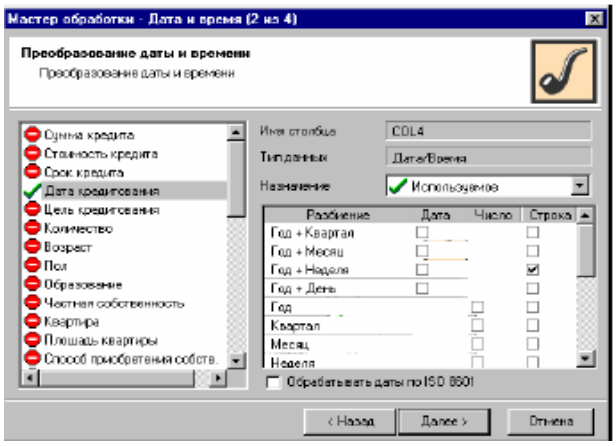


Рис. 8.35. Майстер обробки «Дата и Время».

2. Більше жодні налаштування не знадобляться, тому перейдемо далі до вибору типа візуалізації. Виберемо в якості візуалізації «*Таблицу*» і «*Куб*», поставивши галочок у відповідних позиціях. У майстрові налаштування полів куба виберемо в якості виміру стовпець, що з'явився після обробки, «ДАТА КРЕДИТУВАННЯ_YWStr (Рік + Тиждень)» і стовпець «МЕТА КРЕДИТУВАННЯ», а в якості факту – «СУМА КРЕДИТУ». Останні поля зробимо невживаними.

3. На наступному кроці перенесемо один вимір з області «доступних» в область «*Измерения в строках*», а інше – в область «*Измерения в столбцах*». Таким чином, на крос-діаграмі маємо суми взятих кредитів по тижнях (за перші два тижні року) в розрізі цілей кредитування (рис. 8.36).

	Дата кредитования (Год + Неделя)		
Цель кредитования	2003-W01	2003-W02	Итого
Иное	358 000,00	137 000,00	495 000,00
Оплата за образование	62 000,00	312 000,00	374 000,00
Оплата услуг (мед., юрид. и т.п.)	110 500,00	191 000,00	301 500,00
Покупка и ремонт недвижимости	404 000,00	538 000,00	942 000,00
Покупка товара	642 000,00	643 500,00	1 285 500,00
Турпоездки, развлечения и т.п.	35 500,00	113 500,00	149 000,00
Итого	1 612 000,00	1 935 000,00	3 547 000,00

Рис. 8.36. Крос-діаграма результатів.

У таблиці з даними видно, що нове поле - «ДАТА КРЕДИТУВАННЯ_YWStr (Рік + Тиждень)» містить однакові значення (дата початку тижня) для рядків, які потрапляють в один і той же тиждень (дата початку тижня або номер тижня з початку року) (рис. 8.37).

Срок кредита	Дата кредитов	Дата кред	Цель кредитования
24	05.01.03	2003-W01	Покупка товара
12	05.01.03	2003-W01	Покупка товара
30	05.01.03	2003-W01	Иное
36	06.01.03	2003-W02	Покупка и ремонт недвижимости
12	06.01.03	2003-W02	Оплата за образование
18	06.01.03	2003-W02	Иное
6	06.01.03	2003-W02	Покупка товара

Рис. 8.37. Аналіз результатів розрахунків.

ЧАСТИНА 3. Квантування віку кредиторів на 5 інтервалів.

Часто аналітикові необхідно віднести безперервні дані (наприклад, кількість продажів) до якого-небудь кінцевого набору (наприклад, всю сукупність даних про кількість продажів необхідно розбити на 5 інтервалів – від 0 до 100, від 100 до 200 і так далі, і віднести кожен запис вхідного набору до деякого конкретного інтервалу) для аналізу або фільтрації виходячи саме з цих інтервалів. Для цього в Deductor Studio застосовується інструмент *квантування (або дискретизація).*

Квантування призначене для перетворення безперервних даних в дискретні. Перетворення може проходи як *по інтервалах* (дані розбиваються на задану кількість інтервалів однакової довжини), так і *по квантилях* (дані розбиваються на інтервали різної довжини так, щоб в кожному інтервалі знаходилася однакова кількість даних). В якості значення результуючого набору даних можуть виступати номер інтервалу, нижній або верхній кордон інтервалу, середина інтервалу, або мітка інтервалу (значення визначені аналітиком).

Прикладом використання даного інструменту може служити розбиття даних про вік кредиторів на 5 інтервалів (до 30 років, від 30 до 40, від 40 до 50, від 50 до 60, старше 60 років). Вхідні дані розподіляться по п'яти інтервалах саме так, оскільки, згідно статистиці, мінімальне значення віку кредитора 19, а максимальне 69 років. Це необхідно аналітикові для оцінки кредиторської активності різних вікових груп, з метою ухвалення рішення про стимулювання кредиторів в групах з низькою активністю (наприклад, зменшення вартості кредиту для цих груп) і, мабуть, збільшення прибутку у вікових групах кредиторів з високим ризиком (шляхом збільшення для них вартості кредиту). Причому аналітик бажає бачити дані в розрізі по тижнях (тому продовжимо роботу на останніх отриманих даних попереднього прикладу).

1. Скористаємося майстром квантування (рис. 8.38).

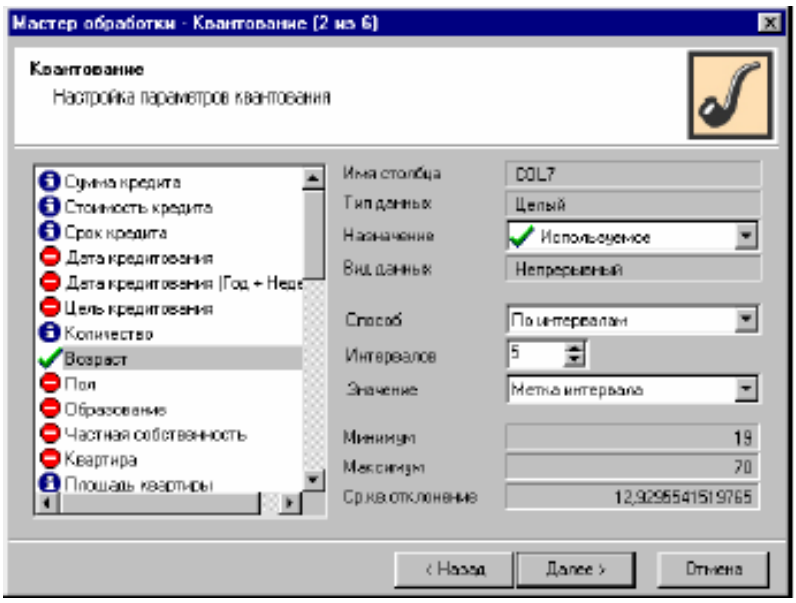


Рис. 8.38. Візуалізатор «Майстер квантування».

В ньому виберемо в якості використовуваного - значення поля «Вік» , вкажемо спосіб розбиття «*По интервалам*», задамо кількість інтервалів рівним 5, в якості значення виберемо «*Метку интервала*».

2. На наступному кроці майстра визначимо мітки віку кредиторів: «до 30 років», «від 30 до 40 років» і так далі (рис. 8.39).

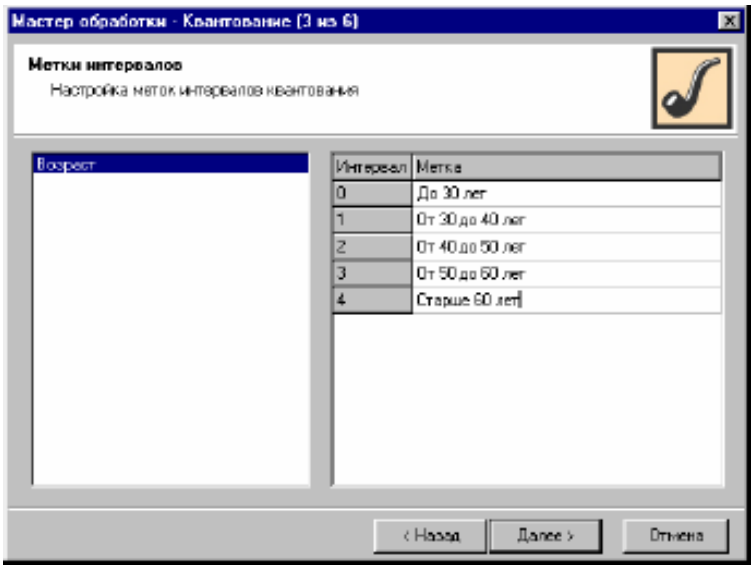


Рис. 8.39. Налаштування міток в «Майстері квантування».

3. Після обробки виберемо в якості способу відображення «*Куб*». У майстрові вкажемо «СУМА КРЕДИТУ» в якості факту, «ВІК» і поле «ДАТА КРЕДИТУВАННЯ (Рік + Тиждень)» в якості виміру, останні поля вкажемо

невживаними. Далі перенесемо «ВІК» з доступних вимірів в «Виміри в рядках», а «ДАТА КРЕДИТУВАННЯ (Рік + Тиждень)» у «Виміри в стовпцях». На крос-діаграмі тепер видно інформацію про те, які суми кредитів беруть кредитори певних вікових груп в розрізі по тижнях (рис. 8.40).

	Дата кредитования (Год + Неделя)		
Возраст	2003-W01	2003-W02	Итого
До 30 лет	798 500,00	808 500,00	1 607 000,00
От 30 до 40 лет	298 000,00	561 000,00	859 000,00
От 40 до 50 лет	195 000,00	295 500,00	490 500,00
От 50 до 60 лет	111 000,00	209 500,00	320 500,00
Старше 60 лет	209 500,00	60 500,00	270 000,00
Итого	1 612 000,00	1 935 000,00	3 547 000,00

Рис. 8.40. Результати квантування.

Тепер аналітик, отримавши такі дані, може дати рекомендації про зниження вартості кредиту для осіб, старше 50 років, або про вживання інших заходів, здатних залучити більшу кількість кредиторів цих груп, або заходів, направлених на те, аби кредитори брали кредит на великі суми.

ЧАСТИНА 4. Обчислювані дані.

Інколи виникає необхідність на деякому етапі обробки даних отримати нові (похідні) дані. Можливо, аналітикові потрібно обчислити процентне відхилення значення одного поля відносно іншого, або підрахувати суму, різницю полів, отримати на основі даних показник і вже його використовувати для подальшої обробки, залежно від значення полів обчислити ті або інші вирази. У Deductor Studio таку можливість надає інструмент «*Вычисляемые данные*». Він дозволяє створювати нові поля, що обчислюють задані аналітиком вирази. Тобто обчислювані дані служать для здобуття похідних даних на основі наявних у вхідному наборі. Майстер надає широкий набір функцій різного напряму. У майстрові представлений список нових виразів, де надаються необхідні аналітикові вираження, список доступних функцій з

коротким описом кожної, список доступних операцій і також список доступних стовпців, які можна задіювати при створенні вираження.

1. Розглянемо застосування цього методу на прикладі даних з файлу, підготовленого засобами MS Excel. У ньому міститься таблиця з полями «АРГУМЕНТ1», «АРГУМЕНТ2», «АРГУМЕНТ3» – набір аргументів. По-перше необхідно імпортувати даний файл в програму. Для перегляду вхідних даних в даному випадку зручніше використовувати візуалізатор «*Таблица*» (рис. 8.41).

	Аргумент1	Аргумент2	Аргумент3
	0	4	4
	0	5	5
	0	6	6
	0	7	7
	0	8	8
	0	9	9

Рис. 8.41. Перегляд даних за допомогою візуалізатора «Таблица».

Допустимо необхідно на основі аргументів розрахувати деякі математичні функції. Хай це будуть дві функції одного аргументу (АРГУМЕНТ3), одна функція від двох аргументів, одна кусково-задана функція і функція, що показує відносне відхилення (АРГУМЕНТ1 + 1 від АРГУМЕНТ2 + 1). Передбачається, що всі ці функції використовуватимуться для подальшої обробки.

2. Функції *F1(АРГУМЕНТ3), F2(АРГУМЕНТ3)*. Розрахуємо значення функцій SIN(АРГУМЕНТ3 * АРГУМЕНТ3) * LN(АРГУМЕНТ3 + 1) * EXP(-АРГУМЕНТ3/10) та 10*SIN(АРГУМЕНТ3 * АРГУМЕНТ3/100) / (АРГУМЕНТ3 + 1) * EXP(- АРГУМЕНТ3/10). Для цього, знаходячись на вузлі імпорту, запустимо майстер обробки. Виберемо в якості обробки - «*Вычисляемые данные*». На другому кроці майстра в списку виразів в першому рядку в графі «*Название выражения*» замість напису «*Выражение*» напишемо

F1(АРГУМЕНТ3). У полі редактора виразів (у верхній частині майстра) напишемо «SIN(COL3*COL3)*LN(COL3+1)*EXP(-COL3/10)» (рис. 8.42).

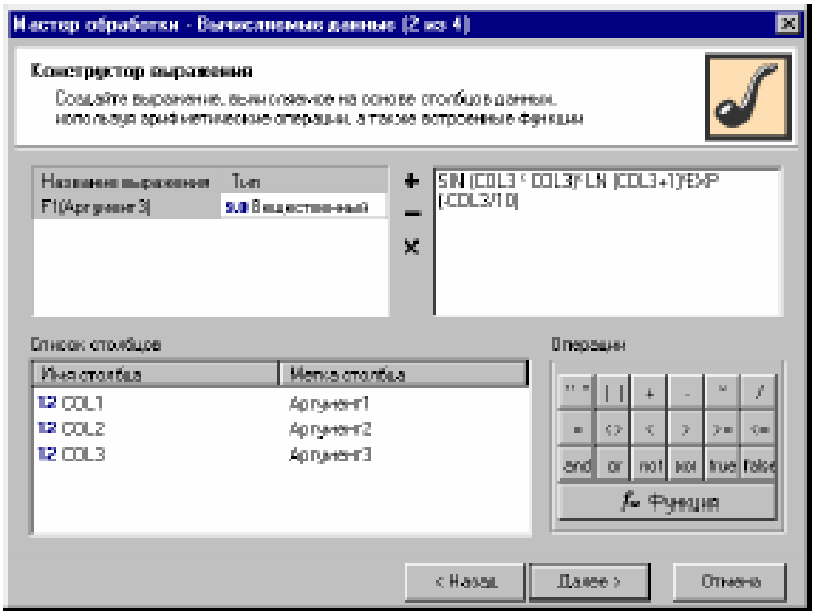


Рис. 8.42. Меню конструктора виразів.

Таким чином, ми створили новий стовпець, задали йому назву «F1(АРГУМЕНТ3)» і також визначили, які значення прийматимуть записи цього поля. На цьому створення обчислюваного значення закінчене, тому переходимо на наступний крок майстра, де пропонується вибрати спосіб відображення даних. Самим інформативним в даному випадку є діаграма, яку і слід вибрати. Якщо вибрати в майстрові налаштувань діаграми в якості відображаємого поля - «F1(АРГУМЕНТ3)» і в якості типа графіка - «Лінії», то можна побачити графік обчисленої функції (рис. 8.43).

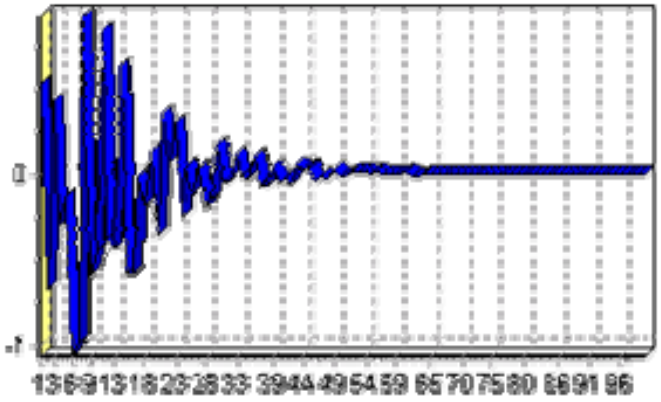


Рис. 8.43. Результати обчислювання даних.

Складна функція F2(АРГУМЕНТ3) відрізняється лише виглядом функції («10*SIN(COL3*COL3/100)/(COL3+1)*EXP(-COL3/10)») (рис. 8.44).

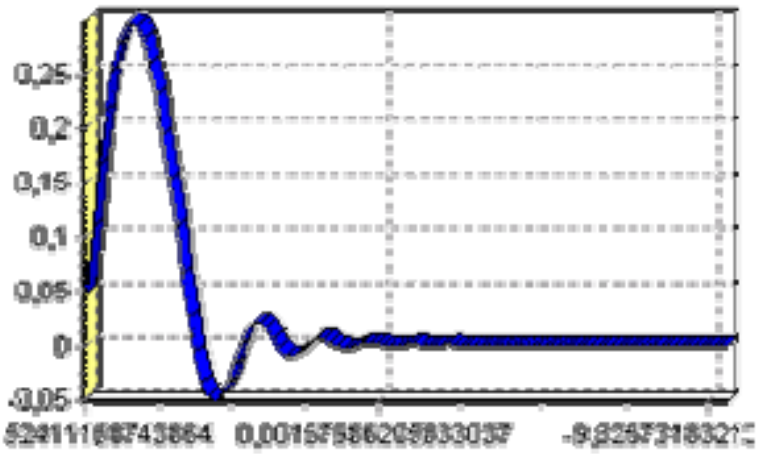


Рис. 8.44. Результати розрахунку функції F2(АРГУМЕНТ3).

3. Функція від двох аргументів *F3(АРГУМЕНТ1; АРГУМЕНТ2)*. Така функція цікава тим, що для її перегляду в трьох вимірах можна використовувати візуалізацію «*Куб*». Задамо назву виразу «F3(АРГУМЕНТ1; АРГУМЕНТ2)», у полі обчислюваного виразу напишемо «COL1*COL1/100 – COL2*COL2/100». Виберемо візуалізацію «*Куб*» і налаштуємо його так, що «АРГУМЕНТ1» і «АРГУМЕНТ2» були б вимірами, «F3(АРГУМЕНТ1; АРГУМЕНТ2)» – фактом, а «АРГУМЕНТ3» – невживаним. Вибравши «АРГУМЕНТ1» виміром в стовпцях, а «АРГУМЕНТ2» – вимірам в рядках перейдемо до перегляду крос-діаграми. Для наочнішого перегляду встановимо тип діаграми «*Области*». Тепер можна проглянути обчислену функцію в об'ємному вигляді (рис. 8.45).

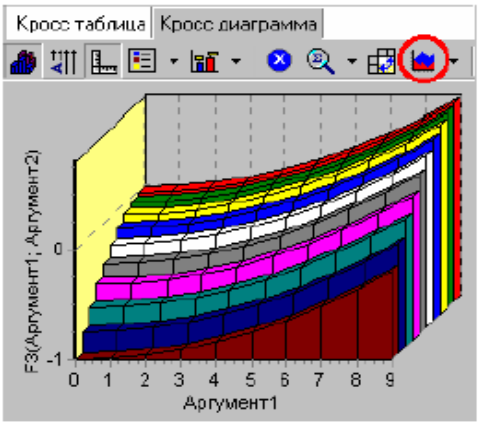


Рис. 8.45. Результати обчислення функцій.

4. Обчислення відхилення «*АРГУМЕНТ1 + 1 від АРГУМЕНТ2 + 1*». Покажемо приклад вживання однієї із вбудованих функцій – обчислення пайового відхилення одного аргументу від іншого (RELDEV). Список всіх вбудованих функцій разом з описом можна поглянути в майстрові в лівому нижньому кутку. Задавши в якості обчислюваного виразу RELDEV(COL1 + 1; COL2 + 1) можна на діаграмі побачити дане відхилення (рис. 8.46).

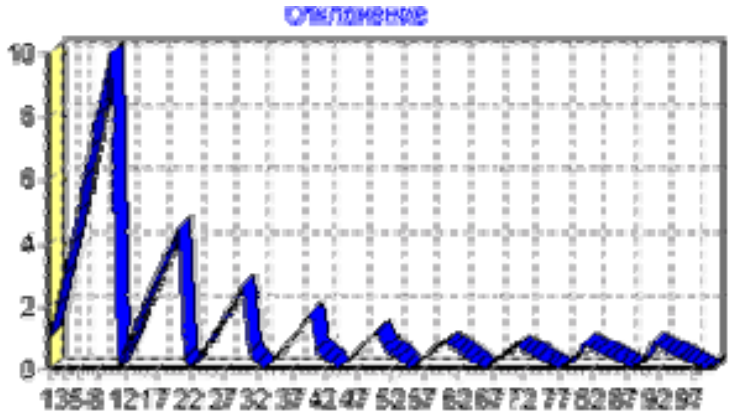


Рис. 8.46. Обчислювання відхилення.

5. Кусково-задана функція. Хай функція приймає значення SQRT(АРГУМЕНТ3/50) при значеннях АРГУМЕНТ3 від 0 до 50 і значеннях АРГУМЕНТ3*АРГУМЕНТ3/2500 - останніх. Для обчислення подібної функції необхідно скористатися функцією *IFF(арумент1; аргумент2; аргумент3),* яка дозволяє залежно від логічного значення першого аргументу отримати другий або третій аргумент. Згідно прикладу, якщо значення аргументу більше нуля і менше 50 необхідно отримати вираз SQRT(АРГУМЕНТ3/50), інакше – вираз АРГУМЕНТ3*АРГУМЕНТ3/250. Таким чином, в полі побудови виразу необхідно написати «IFF((COL3>0)AND(COL3<50); SQRT(COL3/50); COL3*COL3/2500)». Зробивши це в майстрові обробки «*Вычисляемые данные*», і вибравши далі візуалізацію «*Диаграмма*», і також вибравши в майстрові налаштування діаграми поле із значеннями кусково-заданої функції, можна поглянути на необхідний результат (рис. 8. 47).

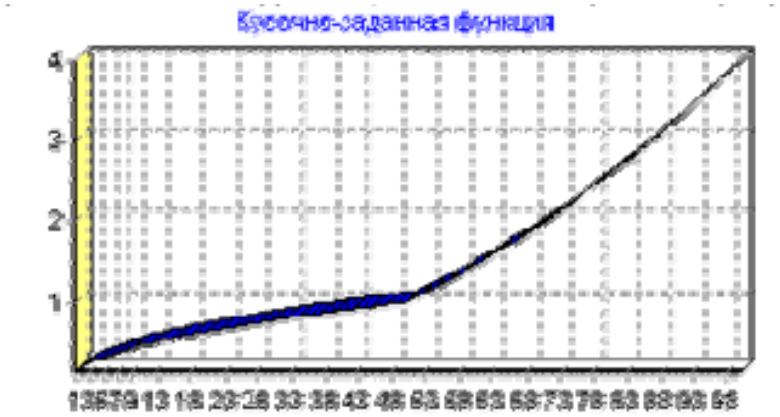


Рис. 8.47. Результати розрахунку кусково-заданої функції.

ЧАСТИНА 5. Групування даних.

Аналітикові для ухвалення рішення завжди необхідна звідна інформація. Сукупні дані набагато більше інформативні, тим більше, якщо їх можна отримати в різних розрізах. У Deductor Studio передбачений інструмент, що реалізовує збір звідної інформації, – «*Группировка*». Групування дозволяє об'єднувати записи по полях - вимірах і агрегувати дані в полях-фактах для подальшого аналізу.

1. Допустимо, що у аналітика є статистика по банках України за певний період. Перед ним стоїть задача виявлення ряду міст, в яких прибуток банків найбільший. Для цього аналітик повинен звернути увагу на наступні поля таблиці з файлу: «БАНК», «ФІЛІЇ», «МІСТО», «ПРИБУТОК». Ясно, що для вирішення поставленої задачі насамперед необхідно знайти сумарний прибуток всіх банків в кожному місті. Для цього і необхідне групування.

По-перше, слід імпортувати дані по банках з табличного файлу. Проглянути вхідну інформацію можна у вигляді куба, де по рядках будуть назви банків, а по стовпцях – міста. За допомогою візуалізації «*Куб*» також можна отримати необхідну інформацію, вибравши в якості виміру поле «МІСТО», а в якості факту - «ПРИБУТОК». Але нам необхідно отримати ці дані для подальшої обробки, отже, необхідно зробити аналогічне групування.

2. Групування по містах. Знаходячись у вузлі імпорту, запустимо майстер обробки. Виберемо в якості обробки – групування даних. На другому кроці майстра встановимо призначення поля «МІСТО» як вимір, а призначення поля «ПРИБУТОК» як факт. В якості функції агрегації поля «ПРИБУТОК» слід вказати «Суму» (рис. 8.48).

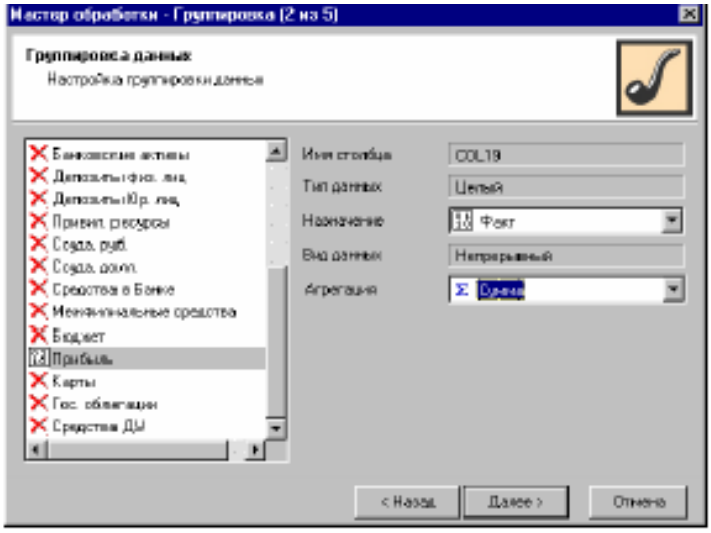


Рис. 8.48. Меню обробки групування даних.

Таким чином, після обробки отримаємо сумарні дані по прибутку всіх банків по кожному місту. Їх можна проглянути, використовуючи таблицю. Тепер аналітикові можна виконувати наступний етап обробки даних (рис. 8.49).

Город	Прибыль
Москва	6076922
Санкт-Петербург	233620
Уфа	370468
Санкт-Петербург	128038
Ханты-Мансийск	30679
Казань	68576
Челябинск	63956

Рис. 8.49. Результат групування даних.

ЧАСТИНА 6. Перетворення даних до ковзаючого вікна.

Коли потрібно спрогнозувати часовий ряд, тим більше, якщо в наявності його періодичність (сезонність), то кращого результату можна

добитися, враховуючи значення чинників не лише в даний момент часу, але і за неділю тому, дві, три, сезон назад, або два. Таку можливість можна дістати після трансформації даних до ковзаючого вікна. Так, наприклад, при сезонності продажів з періодом 12 місяців, для прогнозування кількості продажів на місяць вперед можна як вхідний чинник вказати не лише значення кількості продажів за попередній місяць, але і за 12 місяців назад. Обробка створює нові стовпці шляхом здвигу даних вхідного стовпця вниз і вгору (глибина занурення, горизонт прогнозу).

1. Продемонструємо принцип трансформації даних, використовуючи дані з файлу. У ньому всього 2 поля –«АРГУМЕНТ» - аргумент (час), «ФУНКЦІЯ» – часовий ряд. Імпортуємо дані з файлу (необхідно вказати тип полів – вещественний) і побудуємо діаграму (рис. 8.50).

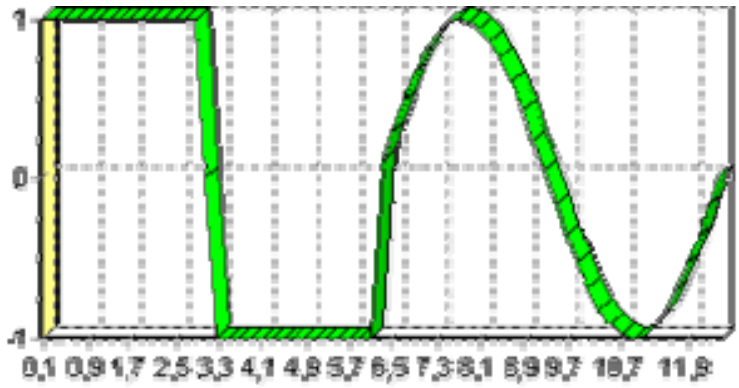


Рис. 8.50. Діаграма вхідних даних.

2. Перетворення ковзаючим вікном. У майстрові перетворення вкажемо призначення стовпця «ФУНКЦІЯ» використовуваним, встановимо для нього глибину занурення 12 і горизонт прогнозу 1.

Після трансформації були отримані нові стовпці – «ФУНКЦІЯ - 11», ... «ФУНКЦИЯ - 1», «ФУНКЦІЯ + 1» на основі стовпця «ФУКЦИЯ». Якщо на діаграмі проглянути декілька таких стовпців, то видно, що дані в них зрушені відносно один одного (рис. 8.51).

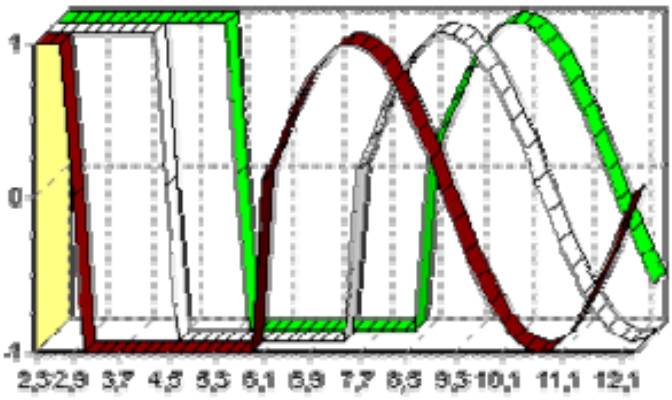


Рис. 8.51. Результати трансформації даних.

ЧАСТИНА 7. Прогнозування за допомогою лінійного регресійного аналізу.

1. Для проведення лінійного регресійного аналізу необхідно запустити майстер обробки і вибрати в якості обробки даних лінійну регресію. На другому кроці майстра налаштуємо поля вхідних даних. Вочевидь, що чинниками будуть спостереження, температура, легковажність і вологість, а результатом – рішення про те, грати в гольф чи ні. Тому необхідно вказати призначення поля «КОД» як інформаційне, поля «РІШЕННЯ» як вихідне, а призначення останніх полів – як вхідні (рис. 8.52).

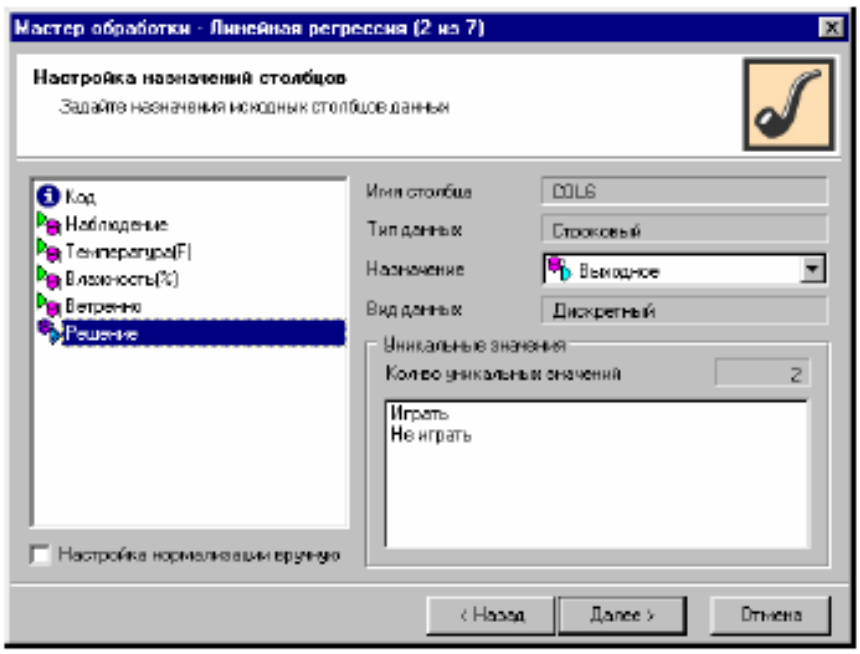


Рис. 8.52. Налаштування поля вхідних даних.

2. На наступному кроці необхідно налаштувати спосіб розділення вхідної множини даних на тестове і навчальне, а також кількість прикладів в тій і іншій множині. Вкажемо, що дані обох множин беруться випадковим чином. Задамо розмір тестової множини рівним двом прикладам шляхом зміни значення стовпця «Розмір в рядках» рядка «Тестова множина».

3. Наступний крок майстра дозволяє виконати обробку, натискуючи на кнопку «Пуск». Під час навчання відображається поточна величина помилки і відсоток розпізнаних прикладів. Після побудови моделі, можна, скориставшись візуалізацією «Що - якщо», зробити прогноз на основі введених метеоумов, а також оцінити якість побудованої моделі, використовуючи таблицю зв'язаності (рис. 8.53).

	Классифицировано		
Фактически	Играть	Не играть	Итого
Играть	8	1	9
Не играть	3	2	5
Итого	11	3	14

	Наблюдение	Темп	Влажн	Ветренно	Решение	Решение_OUT
▶	Солнечно	75	70	Да	Играть	Играть
	Солнечно	69	70	Нет	Играть	Играть
	Пасмурно	83	78	Нет	Играть	Играть

Рис. 8.53. Результат регресійного аналізу.

На таблиці зв'язаності видно, що не всі приклади були класифіковані правильно. Наприклад, ситуація, коли йде дощ, вітер, вологість 80% і температура 65 за Фаренгейтом (18 за Цельсієм) була розцінена як сповна прийнятна для гри в гольф, хоча за таких умов в гольф не грали.

Діаграма «Що – якщо» дозволить провести аналіз впливу одного чинника на результат ухвалення рішення при незмінних останніх (рис. 8.54). З неї, зокрема, видно, що за інших рівних умов (сонячно, температура 75, вітер) на рішення про гру сильно впливає вологість. Якщо вона менше 73%, то варто грати в гольф, якщо більше – то ні.

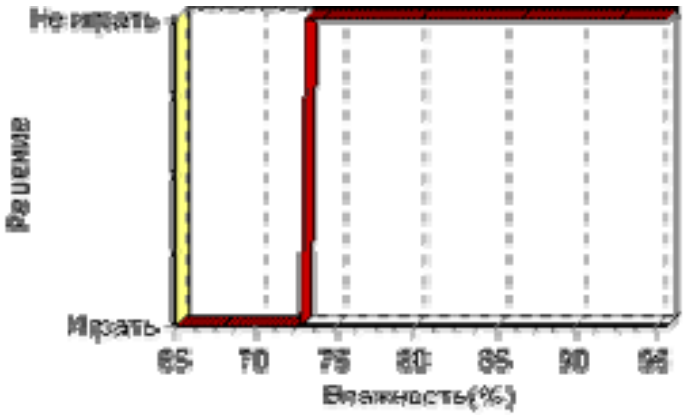


Рис. 8.54. Аналіз за допомогою діаграми «Що – якщо».

ЛІТЕРАТУРА

1. Айвазян С.А., Бухштабер В.М., Енюков И.С., Мешалкин Л.Д. Прикладная статистика: классификация и снижение размерности. – М.: Финансы и статистика, 1989.

2. Ананий В. Левитин Е. Алгоритмы: введение в разработку и анализ. – М.: Вильямс, 2006.

3. Барсегян А.А. Куприянов М.С. Степаненко В.В. Технологии анализа данных: Data Mining, Visual Mining, Text Mining, OLAP. – СПб: БХВ–Петербург, 2008.

4. Барсегян А.А., Куприянов М.С., Степаненко В.В., Холод И.И. Методы и модели анализа данных: OLAP и Data Mining. – СПб.: БХВ–Петербург, 2004.

5. Батыршин И.З. и др. Теория и практика нечетких гибридных систем. – М.: Физматлит, 2007.

6. Бююль А., Цефель П. SPSS: искусство обработки информации**.** – СПб.: ДиаСофт, 2001.

7. Гладков Л.А., Курейчик В.В., Курейчик В.М. Генетические алгоритмы – М.: Физматлит, 2006.

8. Горбань А.Н. и др. Нейроинформатика**.** – Новосибирск: Наука, 1998.

9. Дюк В.А., Самойленко А.П. Data Mining: учебный курс. – СПб.: Питер, 2001**.**

10. Дюран Б., Оделл П. Кластерный анализ. – М.: Статистика, 1977.

11. Елманова Н., Федоров А. Введение в OLAP-технологии Microsoft. – М.: Диалог-МИФИ, 2002.

12. Емельянов В.В. Теория и практика эволюционного моделирования. – М.: Физматлит, 2003.

13. Жамбю М. Иерархический кластер–анализ и соответствия. – М.: Финансы и статистика, 1988.

14. Заде Л. Понятие лингвистической переменной и ее применение к принятию приближенных решений. – М.: Мир, 1976.

15. Захарченко П.В. Моделі курортно-рекреаційної економіки. // Формування ринкової економіки в Україні – 2007. - № 16. - С. 76 – 83.

16. Инмон Б. Производительность систем хранилищ данных. // Data Warehouse Environment – 2000. – №4. – С. 41 - 48.

17. Кацко И.А., Паклин Н.Б. Практикум по анализу данных на компьютере. – М.: Колосс, 2009.

18. Каширина И.Л. Введение в эволюционное моделирование. – Воронеж: ВГТУ, 2007.

19. Киселев М., Соломатин Е. Средства добычи знаний в бизнесе и финансах. // Открытые системы – 1997. – № 4. – С. 41 - 44.

20. Кофман А. Введение в теорию нечетких множеств. – М.: Радио и связь, 1982.

21. Кохонен Т. Ассоциативная память. – М.: Мир, 1980.

22. Круглов В. В., Борисов В. В. Искусственные нейронные сети. Теория и практика. – М.: Горячая линия–Телеком, 2001.

23. Круглов В.В., Дли М.И. Интеллектуальные информационные системы: компьютерная поддержка систем нечеткой логики и нечеткого вывода. – М.: Физматлит, 2002.

24. Кулаичев А.П. Методы и средства комплексного анализа данных. – М: ИНФРА-М, 2006.

25. МакКоннелл Дж. Основы современных алгоритмов. – М.: Техносфера, 2004.

26. Мандель И.Д. Кластерный анализ. – М.: Финансы и Статистика, 1988.

27. Недосекин А.О. Нечетко-множественный анализ фондовых инвестиций. – СПб: Сезам, 2002.

28. Недосекин А.О. Фондовый менеджмент в расплывчатых условиях. – СПб.: Сезам, 2003.

29. Орлов А. И. Нечисловая статистика. – М.: МЗ-Пресс, 2004.

30. Паклин Н.Б., Орешков В.И. Бизнес-аналитика: от данных к знаниям.– СПб.: Питер, 2009.

31. Панченко Т.В. Генетические алгоритмы. – Астрахань: Астраханский университет, 2007.

32. Роберт Калан. Основные концепции нейронных сетей. – М.: Вильямс, 2001.

33. Розенблатт Ф. Принципы нейродинамики. – М.: Мир, 1964.

34. Рутковская Д., Пилиньский М., Рутковский Л. Нейронные сети, генетические алгоритмы и нечеткие системы. – М.: Горячая Линия–Телеком, 2007.

35. Саймон Хайкин. Нейронные сети. Полный курс. – М.: Вильямс 2006.

36. Ситник В.Ф. та ін. Системи підтримки прийняття рішень. – К.: КНЕУ, 2001.

37. Ситник В.Ф., Краснюк М.Т. Інтелектуальний аналіз даних (дейтамайнінг). – К.: КНЕУ, 2007.

38. Спирли Э. Корпоративные хранилища данных. Планирование, разработка, реализация. – М.: Вильямс, 2001.

39. Субботін С.О., Олійник А.О., Олійник О.О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей. – Запоріжжя: ЗНТУ, 2009.

40. Тарков М.С. Нейрокомпьютерные системы. – М.: БИНОМ, 2006.

41. Тахтенгерц Э.А. Компьютерная поддержка принятия решений. – М.: СИНТЕГ, 1998.

42. Тейво Кохонен, Гвидо Дебок .Анализ финансовых данных с помощью самоорганизующихся карт. – М.: Альпина, 2001.

43. Уоссермен Ф. Нейрокомпьютерная техника. - М.: Мир, 1992.

44. Хоббс Л., Хилсон С., Лоуенд Ш. Oracle9iR2: Разработка и эксплуатация хранилищ баз данных.– М.: Кудин–Образ, 2004.

45. Черняк О.І., Кучерук Л.В. Застосування байєсівських мереж в економіці // Вісник Харківського національного університету імені В.Н. Каразіна. Економічна серія – 2009. - № 869. - С.199 - 209.

46. Черняк О.І., Ставицький А.В., Чорноус Г.О. Системи обробки економічної інформації: Підручник. – К.: Знання, 2006.

47. Чубукова И.А. Data Mining. – М.: БИНОМ, 2008.

48. Шпирко О.В. Особливості застосування методів кластерного аналізу для сегментації і моделювання поведінки споживачів // Теоретичні та прикладні питання економіки – 2007. - Вип. 13. - С.202 - 205.

49. Штовба С.Д. Введение в теорию нечетких множеств и нечеткую логику. http://matlab.exponenta.ru/fuzzylogic/index.php.

50. Штовба С.Д. Муравьиные алгоритмы // Exponenta Pro. Математика в приложениях – 2003. – №4. – С.70 - 75.

51. Эделстейн Г. Интеллектуальные средства анализа, интерпретации и представления данных в информационных хранилищах. // ComputerWeek – 1996. – № 16. – С. 32 - 33.

52. Яхъяева Г.Э. Нечеткие множества и нейронные сети. – М.: БИНОМ, 2008.

53. Adamson C., Venerable M. Data Warehouse Design Solutions. – John Wiley & Sons, 1998.

54. Artificial Neural Networks: Concepts and Theory. – IEEE Computer Society Press, 1992.

55. Beltratti A., Margarita S., Terna P. Neural Networks for Economic and Financial Modeling. – ITCP, 1995.

56. Bojadziev G. Fuzzy Logic for Business, Finance and Management // Advances in Fuzzy Systems – 1997. – Vol. 12. – P. 244 - 249.

57. Brin S. et al. Dynamic Itemset Counting and Implication Rules for Market Basket Data. – New York: ACM Press, 1997.

58. Buntine W. A theory of classification rules. – John Wiley & Sons, 1992.

59. Fayyad, Piatetsky-Shapiro, Smyth, Uthurusamy. Advances in Knowledge Discovery and Data Mining. – AAAI/MIT Press, 1996.

60. Han J., Kamber M. Data Mining: Concepts and Techniques. – Morgan Kaufmann Publishers, 2000.

61. Heckerman D. Bayesian Networks for Data Mining. // Data Mining and Knowledge Discovery – 1997. – № 1. – P. 79 - 119.

62. Holland J. H. Adaptation in natural and artificial systems. – University of Michigan Press, 1975.

63. Introduction to Data Mining and Knowledge Discovery: Third Edition. – Two Crows Corp, 1999.

64. Kimball R. The Data Warehouse Toolkit: Practical Techniques for Building Dimensional Data Warehouses. – John Wiley, 1996.

65. Kosko B. Neural Networks and Fuzzy Systems. – NJ: Prentice-Hall, 1991.

66. Koza J. R. Genetic Programming. – Cambridge: The MIT Press, 1998.

67. Mitchell M. An Introduction to Genetic Algorithms. – Cambridge: MIT Press, 1999.

68. Parsaye K. A Characterization of Data Mining Technologies and Processes. // The Journal of Data Warehousing – 1998. – № 1. – P. 31 – 38.

69. Shafer J., Agrawal R., Mehta M. SPRINT: A Scalable Parallel Classifier for Data Mining. – Morgan Kaufmann, 1996.

70. Siu Nin Lam. Discovering Association Rules in Data Mining. – Department of Computer Science, University of Illinois at Urbana-Champaign, 2009.

71. Zimmerman H. Fuzzy Sets Theory - and Its Applications. – Kluwer Academic Publishers, 2001.

72. http://www.basegroup.ru – сайт компанії Basegroup.

73. http://www.olap.ru – сайт з OLAP – технологій та сховищ даних.

74. www.kdnuggets.com – сайт одного із засновників Data Mining Г. Піатецького-Шапіро.

75. http://www.sas.com/ - сайт компанії SAS.

76. http://www.statsoft.ru/ – сайт компанії StatSoft (STATISTICA).

77. http://gandalf.fcee.urv.es/sigef/english/frame.html – SIGEF Association official website (нечіткі обчислення).

78. http://www.nsu.ru/matlab/MatLab_RU/matlab/default.asp.htm – консультаційний центр MATLAB компанії SoftLine.

ПРЕДМЕТНИЙ ПОКАЖЧИК

Б

В

Г

Д

Е

З

І

К

Л

М

Н

О

П

Р

С

Т

Ф

Х

Ш

ВІДПОВІДІ НА ЗАПИТАННЯ ТЕСТІВ.

Розділ 1. 1. а. 2. б, д. 3. а, б. 4. в. 5. б, в, д. 6. б. 7. б, а, в, д, г. 8. в. 9. а. 10. б. 11. в. 12. а. 13. в. 14. б. 15. а. 16. б. 17. в. 18. б. 19. а. 20. в.

Розділ 2. 1. б, в, д. 2. а. 3. в. 4. б. 5. а. 6. в. 7. б. 8. а, в. 9. в. 10. а. 11. в. 12. б. 13. в. 14. а. 15. б. 16. в. 17. в. 18. б. 19. а. 20. б.

Розділ 3. 1. а, г. 2. а, в. 3. б. 4. в. 5. в. 6. а, в, г. 7. б. 8. а. 9. а. 10. б. 11. в. 12. б. 13. а. 14. в. 15. б. 16. а. 17. в. 18. б. 19. в. 20. б.

Розділ 4. 1. г. 2. б. 3. а. 4. а. 5. б. 6. б., в. 7. б., в., д., з. 8. в. 9. б. 10. а. 11. а. 12. б. 13. в. 14. а., в. 15. б. 16.б., в. 17. в. 18. б. 19. а. 20. б., д.

Розділ 5. 1. г. 2. в. 3. б. 4. а. 5. а. 6. б. 7. в. 8. б. 9. а. 10. в. 11. а. 12. б. 13. г. 14. в. 15. б. 16. а. 17. в. 18. а., б. 19. в. 20. б.

Розділ 6. 1. в. 2. а. 3. а., б. 4. в., г. 5. б. 6. а., в. 7. б. 8. в. 9. а. 10. г. 11. б. 12. в. 13. а., г. 14. а. 15. б. 16. в. 17. в. 18. б. 19. а. 20. б.

Розділ 7. 1. в. 2. б. 3. а., в. 4. б., в. 5. б. 6. в. 7. в. 8. а. 9. в., д. 10. б. 11. а., в. 12. б. 13. в. 14. в. 15. б., в. 16. а. 17. б. 18. в. 19. б. 20. а.