ПРИКЛАДНІ ІНТЕЛЕКТУАЛЬНІ СИСТЕМИ ОБРОБКИ ЕКОНОМІЧНИХ ДАНИХ

Самостійна робота за Змістовним модулем 4: *Характеристика програмних засобів створення експертних систем*

1. Категорії сучасних програмних засобів розробки експертних систем

Як уже зазначалось у попередніх розділах, сучасні мови програмування використовуються в поєднанні з набором допоміжних програм, формуючи таким чином інструментальний засіб розробки програмних систем. Нагадаємо, що експертна система - це, по суті, різновид програмної системи, яка оперує із знаннями в певній предметній області з метою вироблення рекомендацій для вирішення задач.

Практично всі інструментальні засоби, що використовуються в процесі розробки експертних систем, застосовують методологію автоматизації проектування на основі прототипів. По відношенню до програмного забезпечення термін "прототип" означає працюючу модель програми, яка функціонально еквівалентна підмножині кінцевого продукту.

Ідея використання прототипів полягає в розробці на ранній стадії роботи проекту спрощеної версії кінцевої програми, яка була б спроможна послужити доказом продуктивності основних ідей, покладених в основу проекту. Тобто прототип має бути здатний вирішувати одну з характерних задач для заданої області застосування. На основі аналізу досвіду роботи з прототипом розробники можуть уточнити вимоги до основних функціональних характеристик експертної системи. Працездатність прототипу може послужити наочним доказом можливості рішення проблем за допомогою створюваної системи ще до того, як на її розробку будуть витрачені значні засоби.

Процес розробки експертної системи, як правило, складається з послідовності окремих етапів, упродовж яких нарощуються можливості системи, причому кожен з етапів поділяється на фази: проектування, реалізації, компонування та тестування. В результаті, після завершення чергового етапу утворюється система, здатна впоратися з більшими за складністю варіантами проблеми.

На відміну від експертних систем, при створенні більшості програмних продуктів інших видів використовується інша модель процесу: спочатку розробляється специфікація продукту, потім виконується планування, проектування компонентів, їх реалізація, компонування комплексу та тестування кінцевого варіанта. Той факт, що при розробці експертних систем існує можливість спочатку побудувати та всебічно випробувати прототип,

дозволяє уникнути безлічі переробок у процесі створення робочої версії системи.

Однак слід зазначити, що технологія послідовного нарощування функціональних можливостей містить у собі проблему інтеграції нових функцій системи з функціями, що були реалізовані в попередніх варіантах. Тому інструментальні засоби розробки експертних систем від початку створювалися на основі модульного представлення знань з урахуванням необхідності подолання виникаючих при цьому ускладнень.

За своїм призначенням та функціональними можливостями інструментальні засоби, що використовуються в програмуванні експертних систем, можна розділити на чотири досить великі категорії:

1. Оболонки експертних систем.
2. Мови програмування високого рівня.
3. Середовище програмування, що підтримує декілька парадигм.
4. Додаткові модулі.

Розглянемо детально кожну з цих категорій.

Оболонки експертних систем. Системи типу оболонки експертних систем створюються, як правило, на основі експертних систем, які достатньо добре зарекомендували себе на практиці. В процесі створення оболонки, з системи-прототипу видаляються компоненти, які є специфічними для області її безпосереднього застосування, а залишаються ті, що не мають вузької спеціалізації.

Прикладом може слугувати система EMYCIN, створена на основі системи MYCIN. У структурі EMYCIN був збережений інтерпретатор, усі базові структури знань та пов'язаний з ними механізм індексації. Оболонка була доповнена засобами підтримки бібліотеки типових випадків та висновків, зроблених по ним експертною системою.

У загальному сенсі оболонки експертних систем створюються з метою дозволити непрограмістам скористатися результатами роботи програмістів, що вирішували аналогічні проблеми. Клас цих інструментальних засобів орієнтований на достатньо вузький клас задач, хоча й ширший, ніж та програма, на основі якої була створена оболонка.

Більшість комерційних продуктів цього типу підходить виключно для тих проблем, в яких простір пошуку невеликий. Як правило, в них застосовується метод пошуку рішення з побудовою ланцюжка зворотного логічного висновку та обмежені можливості управління процесом. Цей підхід погано підходить для вирішення проблем конструювання, тобто об'єднання окремих елементів в єдиний комплекс з урахуванням заданих обмежень.

Проте, незважаючи на обмеження, цей тип експертних систем прогресує. У порівнянні з першими розробками, сучасні оболонки більш гнучкі, принаймні в тому, що без особливих зусиль можуть бути інтегровані в більшість операційних середовищ та оздоблені достатньо розвиненими засобами користувальницького інтерфейсу.

Мови програмування високого рівня. Мови програмування високого рівня можуть бути ефективним засобом швидкого створення прототипів

експертних систем. Вони дозволяють забезпечити гнучкість процесу розробки, мінімізацію матеріальних витрат і термінів виконання проекту. Інструментальні засоби цієї категорії позбавляють розробника необхідності заглиблюватися в деталі реалізації системи, такі як способи ефективного розподілу пам'яті, низькорівневі процедури доступу до даних та маніпулювання ними. Як правило, середовище розробки таких мов дозволяє суміщати вставку, редагування та тестування фрагментів програмного коду.

Досвідчений програміст, застосовуючи для проектування експертних систем мови високого рівня, одержує значно більшу свободу дій, ніж при використанні оболонки. Особливо це стосується програмування процедур управління та обробки невизначеності. Ці процедури дуже корисні для створення експериментальних систем, в яких не видається можливим заздалегідь обрати оптимальний режим управління.

Інструментальні засоби експертних систем, зазвичай, використовують тип мов програмування високого рівня, відомий як мова опису продукційних правил. Одним із найвідоміших представників таких мов є OPS5. Для цієї мови характерний порівняно простий синтаксис та механізм активізації правил. У ньому використовуються різні версії rete-алгоритму для оптимізації процесів узгодження фактів із правилами. Нагадаємо, rete-алгоритм позбавляє машину логічного висновку необхідності погоджувати факти з кожним правилом.

Для мови OPS5 характерною ознакою є труднощі при реалізації деяких типів структур управління ходом виконання. Наприклад, до них можна віднести рекурсивні та ітераційні цикли, оскільки вони вимагають серйозного ускладнення опису процесу обробки правил. Розробники мов, подібних OPS, завжди вимушені шукати компроміс між наочністю засобів мови програмування та ефективністю виконання програмного коду.

На думку фахівців, найбільш раціональний шлях подолання недоліків програмування продукційних правил полягає в об'єднанні їх з іншими парадигмами програмування. Прикладом такого об'єднання може бути комбінування продукційних правил та фреймів, що дозволяє зіставляти умови, специфіковані в правилах, із вмістом слотів фреймів.

Іншим типом мов програмування високого рівня, що застосовується в інструментальних засобах експертних систем, є об'єктно-орієнтовані мови. В цьому контексті мовами об'єктно-орієнтованого програмування створюється програмне середовище для організації знань у термінах декларативного представлення об'єктів предметної області. Усі дії, пов'язані з процедурною стороною рішення проблем, розподіляються між цими об'єктами, які мають у своєму розпорядженні власні процедури та можуть спілкуватися один з одним за допомогою інтерфейсів передачі повідомлень.

До ще одного корисного аспекту об'єктно-орієнтованого програмування належить можливість інтеграції символічних обчислень в операційне середовище, яке базується на засобах графічного інтерфейсу. Оснащення експертної системи цими засобами дозволяє краще представити користувачу процеси, що відбуваються в системі.

Основна причина складності використання об'єктно-орієнтованого стилю в програмуванні експертних систем полягає в організації співвідношення програмних об'єктів з абстрактними поняттями та категоріями предметної області. Тобто в експертних системах об'єкти повинні представляти факти та цілі, набори правил або окремі гіпотези, а не моделі елементів реального світу, як у класичних задачах. Тому схеми відображення цих понять та категорій на програмні об'єкти, а також повідомлення, якими вони повинні обмінюватися, мають бути ретельно продумані.

Окрім розглянутих мов програмування високого рівня, в інструментальних засобах експертних систем також застосовуються мови логічного програмування. Типовою мовою логічної розробки експертних систем є PROLOG. Для цього PROLOG володіє достатньо корисними можливостями, а саме:

- вбудований у PROLOG режим управління приблизно відповідає стратегії зворотного логічного висновку;
- індексовану базу даних фраз мови PROLOG можна використовувати для представлення правил;
- рекурсивні структури даних (графи та дерева) можна організувати за допомогою фраз мови PROLOG;
- універсальний механізм зіставлення мови PROLOG дозволяє виконувати зіставлення даних та шаблонів, що включають змінні;
- мовні засоби PROLOG дозволять програмісту розробити власний механізм обробки невизначеності.

Проте практика застосування ідей логічного програмування в експертних системах не позбавлена недоліків. Зокрема, наявність синтаксичних та семантичних обмежень, що присутні в стандартних версіях PROLOG, не були подолані ані в системах MECHO та PLANNER, ані в інших системах, що базуються на аналогічній ідеології.

Середовище розробки експертних систем, що підтримує декілька парадигм. Засоби цієї категорії включають декілька програмних модулів, що дозволяє комбінувати в процесі розробки експертної системи різні стилі програмування, вибираючи відповідні поєднання різних методів. Причиною їх створення стали результати роботи експертних систем із різними схемами представлення знань та логічного висновку. Виявилося, що кожна з них має свої слабкі сторони.

Зокрема, продукційні правила дозволяють представити в програмі емпірично виявлені зв'язки між умовами та діями. Проте вони значно гірше підходять для представлення відносин між об'єктами предметної області, включаючи такі важливі відносини, як "множина-елемент" або "множина-підмножина". З іншого боку, структуровані об'єкти, наприклад фрейми, виявляються зручним засобом для зберігання та маніпулювання описами об'єктів предметної області. Але застосування таких знань вимагає включення в програму фрагментів програмного коду, які потім важко аналізувати.

У результаті аналізу наведених реалій логічним чином була сформована ідея об'єднання методик в єдине середовище, в якому переваги одних

компенсують недоліки інших. Одним із перших багатофункціональних середовищ штучного інтелекту став відповідний продукт, що має назву LOOPS (Bobrow and Steflk, 1983). У ньому в рамках єдиної архітектури обміну повідомленнями були об'єднані чотири парадигми програмування: 1. Процедурно-орієнтоване програмування. Ця парадигма була представлена мовою LISP, в якій активним компонентом є процедури, а пасивним - дані. В рамках єдиного середовища процедури можуть бути також використані для обробки зовнішніх даних.

2. Програмування, орієнтоване на правила. Ця парадигма аналогічна попередній, але роль процедур виконують правила "умова-дія". В середовищі LOOPS набори правил за своєю суттю є об'єктами, які можна рекурсивно вкладати один у другий. Таким чином, частина "дія" одного правила, у свою чергу, може активізувати підлеглий набір правил.

3. Об'єктно-орієнтоване програмування. Згідно з цією парадигмою структуровані об'єкти володіють властивостями як процедур, так і даних. Обробка вхідних повідомлень призводить до передачі даних або зміни їх значень. Усі маніпуляції даними виконуються під управлінням того компонента, який звернувся до об'єкта. При цьому зовнішні об'єкти не інформуються про те, яким чином зберігаються дані та як вони модифікуються усередині об'єкта.

4. Програмування, орієнтоване на дані. В ньому процеси доступу та оновлення даних запускають певні процедури. Зі змінними, в яких зберігаються значення даних, пов'язані певні процедури, подібно до того, як це робиться у слотах фреймів.

Основу системи складає об'єктно-орієнтована парадигма. В рамках її модулів можна комбінувати модулі середовища, що підтримують різні стилі програмування. Зазвичай, умови в продукційних правилах пов'язуються із значеннями слотів структурованих об'єктів, а правила модифікують значення цих слотів. Саме такий стиль об'єднання парадигм реалізований у мові CLIPS.

У системах KEE і LOOPS поведінка об'єктів описується в термінах множини продукційних правил. У середовищі Knowledge Craft до перерахованих вище парадигм додане логічне програмування в стилі мови PROLOG. Одна з наступних версій KEE, відома під назвою KA-PPA-PC, надає в розпорядження програміста ще більш розширений набір стилів для комбінування правил, об'єктів та процедур.

Додаткові модулі розробки експертних систем. Засоби цієї категорії є автономними програмними модулями, які призначені для виконання специфічних задач у рамках вибраної архітектури експертної системи. Під "додатковими модулями" розуміються корисні програмні розробки, які можна виконувати разом із основним додатком. Як правило, такі програми реалізують деякі спеціальні функції, підключаючи їх, начебто, з поза меж системи. Причому звернення до таких функцій не потребує додаткового програмування в основному додатку.

Прикладом додаткового модуля може бути модуль роботи з семантичною мережею, використаний в системі VT. Цей модуль дозволяє в

процесі роботи над проектом відстежувати зв'язки між значен -нями раніше встановлених та нових параметрів проектування. Подібні модулі управління семантичною мережею можна застосовувати для розповсюдження внесених змін на всі компоненти системи. Іншим прикладом додаткового модуля може слугувати програмний пакет Simkit із комплекту середовища КЕЕ. Цей пакет дозволяє доповнити експертну систему методами моделювання.

Тенденція використання додаткових модулів найімовірніше розвиватиметься, оскільки користувачі експертних систем часто мають потребу в різних додаткових функціональних можливостях, специфічних для конкретного додатка. Більше того, часто виникає необхідність інтегрувати експертну систему з програмними продуктами інших класів. На практиці експертна система часто використовується разом з базою даних або системою управління, яка одержує інформацію від пакетів статистичної обробки або систем обробки сигналів.

2. Характерні складнощі розробки експертних систем та способи їх уникнення

Процес розробки та впровадження експертної системи має характерні складнощі. Їх систематизація дозволяє виробити практичні рекомендації, що дозволяють їх уникати. Один із найбільш об'єктивних способів систематизації цих складнощів та способів їх подолання запропонував у своїх працях вчений Уотерман (Waterman). Скористаємося його результатами.

Перша складність з'являється тоді, коли знання, що стосуються предметної області, дуже тісно переплетені з іншими частинами системи. Зокрема, може бути важко відокремити ці знання від знань загального застосування, що стосуються способів пошуку в просторі рішень. Уотерман припускає, що цього можна досягти, поклавши в основу організації бази знань набір правил. Проте у спеціалізованій літературі існують думки, що така організація не завжди гарантує досягнення бажаного результату.

Друга складність полягає у тому, що база знань, яка сформувалася після отримання та представлення множини правил у процесі опитування експертів, виявляється неповною настільки, що не дозволяє вирішувати потрібні задачі. Причиною цього є відсутність у ній фундаментальних концепцій предметної області або ці концепції представлені з помилками. Для її вирішення рекомендується послідовно нарощувати обсяг бази знань, починаючи із фундаментальних понять. Це дозволить ще на ранніх стадіях розробки виявити вказану проблему. Крім того, корисно виконувати тестування на кожному етапі розробки, використовуючи для цього відповідні інструментальні засоби інженерії знань.

Третя складність виникає, коли середовище розробки не має у своєму розпорядженні вбудованих засобів формування функцій пояснення експертної системи. Додавання таких функцій у вже спроектовану систему - задача не з легких. Для її запобігання слід піклуватися про прозорість експертної системи

з перших кроків її розробки. Крім того, без засобів моніторингу навіть її творець не може бути до кінця впевнений, що вона працює належним чином.

Четверта складність полягає у тому, що система може містити надмірну кількість дуже специфічних правил. Це, по-перше, призводить до уповільнення роботи системи, а по-друге - ускладнює управління нею. Для її запобігання рекомендується об'єднувати, де тільки можливо, дрібні правила в більш загальні. Це виглядає як прагнення знайти компроміс між ефективністю правил та їх зрозумілістю.

Подальші питання, з необхідністю вирішення яких стикаються розробники експертних систем, полягають у виборі відповідного інструментарію, оцінці наочності вибраних засобів і визначенні "хорошого стилю програмування" у вибраному середовищі розробки. Подальший матеріал буде сконцентрований саме на цих питаннях.

3. Методика вибору оптимального інструментарію для розробки експертної системи

В основу рекомендацій щодо вибору інструментальних засобів для побудови експертної системи у відповідних наукових працях покладене зіставлення характеристик проблем, що має вирішувати експертна система, та необхідних функціональних можливостей інструментального комплексу. При цьому рекомендується дотримуватись наступних загальних правил:

• слід вибирати інструмент зі ступенем сервісної насиченості, який не перевищує необхідного рівня для вирішення даної задачі;

• вибір інструментарію повинен визначатися, в першу чергу, характеристиками задачі, яку вирішуватиме експертна система, а не сторонніми обставинами (наприклад тим, що якийсь інструмент вже є в наявності або знайомий краще за інших);

• якщо успіх проекту залежить від терміну розробки, то слід вибирати інструментальне середовище із вбудованими засобами формування пояснень та елементів користувальницького інтерфейсу, оскільки їх розробка найбільш трудомістка;

• необхідно якнайшвидше провести випробування нового інструментального середовища на реальних даних.

Найважливішим питанням у процесі вибору інструментального середовища є питання способу визначення характеристик проблеми, для вирішення якої призначається експертна система. Ці характеристики можна звести до 4-х основних категорій.

1. Малий простір рішень, надійні дані та знання. Передбачається, що кількість альтернатив, які слід брати до уваги при пошуку рішення, є невеликою, всі дані є достовірними, а істинність правил не викликає сумнівів. Для вирішення проблем цієї категорії можна скористатися готовими рішеннями, тобто раніше створеною оболонкою на базі експертної системи, що вирішувала аналогічну проблему в іншій предметній області.

2. Ненадійні дані або знання. Якщо дані та(або) знання ненадійні, то існує небезпека, що дані, які вводяться в систему, не є достовірними, а правила в базі знань не мають однозначності. У цьому випадку в експертній системі потрібно комбінувати інформацію від декількох джерел та використовувати логіку нечітких міркувань.

3. Великий факторизований простір рішень. Простір пошуку можна назвати факторизованим, якщо існує можливість розділити його на декілька незалежних підпросторів, які можна обробляти окремо. Причому для різних підпросторів можуть бути використані різні множини правил або окремі підмножини однієї й тієї ж множини правил. Зазвичай, таке розбиття виконується на рівні проблеми, тобто велика загальна проблема розбивається на декілька дрібніших. Успіх у досягненні головної мети оцінюється за сукупністю успіхів у досягненні незалежних цілей.

4. Великий нефакторизований простір рішень. Простір рішень може виявитися нефакторизованим, якщо задача допускає вироблення приватного рішення будь-якого компонента тільки в контексті всього проекту. Загальний підхід до роботи у великому просторі пошуку полягає в тому, щоб послідовно розглядати його на різних рівнях абстракції. Тобто потрібно використовувати варіанти описання простору з різним рівнем урахування деталей. Рішення проблеми таким методом часто називають спадним уточненням. З'ясувавши характеристики проблеми, для вирішення якої розробляється експертна система, можна визначитися з властивостями простору рішень. Потім вони розглядаються спільно з передбачуваними характеристиками системи: моделлю подання знань, напрямком логічного висновку, способом формування пояснень. У результаті виявляються бажані характеристики інструментального середовища, які дозволяють підібрати потрібну модель інструментального середовища. Як показує практика, більшість розробників явно або неявно використовують саме такий підхід при створенні експертних систем.

У процесі вибору інструментального середовища важливе значення відіграють також наступні аспекти:

• наскільки просте середовище у використанні;

• як швидко розробники експертної системи зможуть оволодіти методикою роботи в цьому середовищі;

• яку підтримку готова надавати фірма-розробник середовища;

• яка буде загальна вартість середовища з урахуванням прямих і непрямих витрат.

У матеріалах, що описують програмні засоби з розробки експертних систем, можна зустріти твердження, що даним інструментом "може успішно користуватися програміст, мало знайомий із технологіями штучного інтелекту", або навіть непрограміст. Проте практика показує, що це не так. Практичне оволодіння типовими інструментальними засобами проектування експертних систем не поступається за складністю оволодінню новою мовою програмування.

Як правило, типове середовище розробки експертних систем підтримує чотири режими роботи:

• підготовка та редагування бази знань;

• використання бази знань для виконання консультацій, тобто прогін програми;

• виявлення та усунення помилок на стадії компіляції;

• виявлення та усунення помилок на стадії виконання.

Як показав досвід, навіть досвідчені програмісти важко засвоюють методику сумісного використання цих режимів у процесі проектування експертної системи. Це пов'язано, перш за все, з тим, що стандартна стратегія розробки бази знань передбачає постійне нарощування її обсягу. Тому інженеру по знанням доводиться виконувати ітеративні процедури поповнення бази знань значно частіше, ніж звичайному програмісту виконувати розширення функцій програми.

В міру збільшення складності проектованої системи відбувається збільшення обсягу бази знань, додавання до розгляду різного роду не-визначеностей, включення в роботу системи додаткових режимів. Тому стратегія проектування вимагає від розробників дедалі більш ретельної попередньої підготовки. Крім того, можна виділити інші характерні причини складності вибору інструментального середовища розробки експертних систем:

• більшість розвинених середовищ розробки надто дорогі для того, щоб купувати їх для проведення порівняльного аналізу;

• час, необхідний для засвоєння навиків роботи з системою та виявлення її сильних та слабких сторін, дуже великий, тому складно проводити порівняння конкуруючих моделей на практиці;

• термінологія, яку застосовують у документації розробники різних систем, істотно відрізняється, тому проводити їх порівняння на основі технічної документації достатньо важко.

Останнє зауваження справедливе відносно більшості програмних продуктів, що пропонуються на ринку. Коли ж йдеться про програмні засоби, пов'язані з областю штучного інтелекту, то новизна та незвичність термінології ще більш посилює проблему. Вже давно в середовищі фахівців існує думка, що порівняння конкуруючих систем одного класу можна виконувати тільки після ретельного вивчення їх на практиці.

На завершення слід зазначити, що будь-які інструментальні засоби потребують адекватної методології користування ними. Тому доречно буде розглянути загальні рекомендації, які визначають стиль розробки експертних систем:

1. Задача, яку передбачається вирішувати за допомогою експертної системи, повинна бути повністю під силу експерту-людині.

2. Задача повинна бути чітко сформульована. Краще створити систему, яка зможе надійно вирішувати обмежену задачу, ніж систему, що претендує на рішення широкого класу задач, проте дає правильне рішення лише час від часу.

3. Починаючи з першої стадії роботи над системою, необхідно визначити, як вона буде вдосконалюватися, та окреслити межі, яких вона повинна досягти в процесі еволюції.

4. Слід ретельно відпрацювати поведінку системи на наборі окремих випадків та організувати бібліотеку таких випадків. Тобто приклади, які застосовувались на етапі проектування, повинні бути репрезентовані.

5. Потрібно відділити ті знання, які є специфічними для певної предметної області, від знань, які стосуються загальної методики рішення проблем. Бажано, наскільки це можливо, спростити машину логічного висновку в системі.

6. Необхідно на перших стадіях проектування системи розробити однозначні угоди про оформлення програм. Це надасть їй одноманітний вигляд.

7. Бажано поступатися продуктивністю програми, якщо це зробить її більш зрозумілою та спростить її супровід. Це необхідно, оскільки в роботі інтерактивних експертних систем багато часу йде на діалог з користувачем та звернення до баз знань.

8. Як тільки постане питання про розробку нового прототипу системи, від попереднього необхідно відмовитися. Багато проектів зазнали невдачі лише тому, що їх автори не змогли позбавитися прихильності до першого варіанта реалізації власних ідей. Звичайно, у процесі розробки нового прототипу потрібно враховувати досвід створення попереднього, але тільки досвід, а не програмний код.

9. Розробка експертної системи, яка буде здатна успішно працювати, вимагає наполегливості та терпіння професійного програміста, залучення до роботи досвідченого експерта у відповідній області та певного рівня примусу з боку керівництва.

4. Фундаментальне поняття нейронних мереж

У 80-х роках XX століття до числа відомих принципів програмування увійшов новий напрям розробок інформаційних систем, який одержав назву штучних нейронних систем. Цей напрям був заснований на відкритті одного із способів обробки інформації мозком людини, згідно з яким процеси рішення задач моделюються шляхом навчання моделей нейронів, що утворюють мережу. Зазначений підхід іноді згадується під назвою коннекціонізму (від англ. connection - з'єднання).

Фундаментальне поняття нейронних мереж обумовлює структура системи обробки інформації, що складається з великої кількості елементів обробки даних - нейронів, які сполучаються зв'язками - синапсами. Конфігурація нейронної мережі налаштовується на реалізацію конкретного додатка за допомогою процесу засвоєння знань, тобто навчання. При цьому аналогічно біологічним системам навчання передбачає внесення змін у характеристики синаптичних з'єднань між нейронами.

Передбачено багато способів класифікації штучних нейронних систем. Найбільш корисна, з практичної точки зору, класифікаційна ознака полягає в наявності або відсутності потреби застосовувати для такої системи навчальну сукупність даних. Якщо навчальна сукупність передбачена, то штучна нейронна система називається заснованою на контрольованій моделі. В іншому випадку модель вважається неконтрольованою.

Прикладом контрольованої штучної нейронної системи є система, яка використовується для розпізнавання образів. На відміну від неї, у випадках, коли точно не відомо якими повинні бути вихідні дані, штучна нейронна система може бути застосована як класифікатор для групування вхідних даних. Наприклад, розпізнавання хворих та здорових людей у разі виявлення спалахів захворювання.

Нейронні мережі також можуть класифікуватися згідно з наступними характеристиками:

- спосіб з'єднання нейронів;
- застосування обчислень, що виконують нейрони;
- спосіб передачі шаблонів активності по мережі;
- спосіб та швидкість навчання.

Нейронні мережі застосовувалися для вирішення практичних задач усіх типів. Основною перевагою нейронних мереж є те, що вони дозволяють вирішувати задачі, які виявляються надто складними для звичайних технологій. Маються на увазі задачі, які не мають алгоритмічного рішення, або для яких алгоритмічне рішення є дуже складним, щоб його можна було визначити аналітично. Власне кажучи, нейронні мережі добре підходять для вирішення задач, які успішно вирішують люди, але не можуть пояснити, як вони це роблять.

До числа задач нейронних систем належать задачі розпізнавання образів та прогнозування подій, зокрема, знаходження тенденції зміни даних. У теперішні часи штучний інтелект на основі нейронних мереж широко використовується під час аналізу прихованих закономірностей для виявлення в історичних даних таких шаблонів, якими можна керуватися в майбутньому. Наприклад, аналіз прихованих закономірностей у даних може застосовуватися в банківській установі для визначення характеристик потенційного зловмисника серед позичальників.

Крім того, нейронні мережі використовуються як інтерфейсна частина в експертних системах, які обробляють великі обсяги вхідних даних від датчиків, і від яких вимагається реакція в реальному часі. Зокрема, у 1980-х роках нейронна мережа, яка експлуатувалася на звичайному мікрокомп'ютері, дозволила одержати дуже якісне рішення задачі комівояжера за 0,1 секунди. Цей результат набагато перевищував час отримання оптимального рішення із застосуванням традиційної алгоритмічної системи на аналогічному устаткуванні, який складав на той час одну годину.

Слід зазначити, що задача комівояжера має важливе практичне значення, оскільки є класичною задачею, яку доводиться вирішувати під час маршрутизації пакетів в системі передачі даних. Задача пошуку оптимальних

маршрутів є важливим засобом мінімізації часу, оскільки від цього залежать ефективність та швидкодія, як при маршрутизації пакетів даних через Інтернет, так і при доставці посилок поштою численним адресатам.

5. Характеристики штучної нейронної системи

Штучна нейронна система може розглядатися як аналоговий обчислювальний комплекс, в якому використовуються прості елементи обробки даних, які, здебільшого, паралельно сполучені один з одним. Елементи обробки даних виконують дуже прості логічні або арифметичні операції над своїми вхідними даними. Основою функціонування штучної нейронної системи є те, що з кожним елементом такої системи пов'язані вагові коефіцієнти. Ці вагові коефіцієнти представляють

інформацію, що зберігається в системі. Схема типового штучного нейрону зображена на рис. 1.

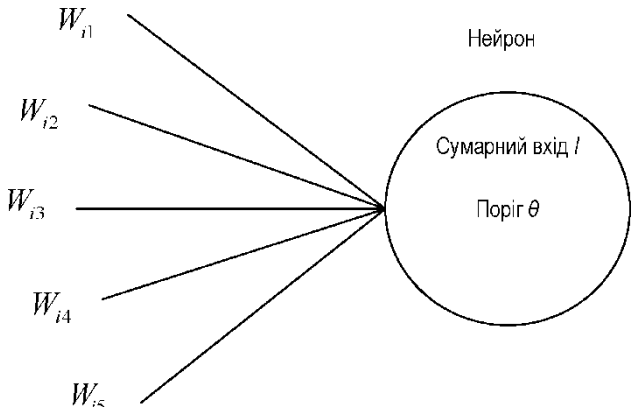

Рис. 1. Типовий штучний нейрон

Нейрон може мати багато входів, але тільки один вихід. Людський мозок містить приблизно 10^{11} нейронів, і кожен нейрон може мати тисячі з'єднань з іншими. Вхідні сигнали I і нейрона помножуються на вагові коефіцієнти та складаються для отримання сумарного входу нейрона - I:

$$I = \sum_j W_{ij} I_j$$

Функція, яка зв'язує вихід нейрона з його входами, називається функцією активізації. Вона має вигляд сигмоідальної функції $(1+e^{-x})^{-1}$. Формалізація реакції нейрона полягає у тому, що вихідний сигнал прямує до однієї з меж при отриманні дуже малих та дуже великих вхідних сигналів. Крім того, з кожним нейроном пов'язане по-рогове значення - в, яке у формулі обчислення вихідного сигналу віднімається від загального вхідного сигналу. В результаті, вихідний сигнал нейрона - О часто описується наступним чином:

$$O = \frac{1}{1+e^{-(I-\theta)}}.$$

Як приклад можна навести штучну нейронну мережу, яка здатна обчислювати значення логічної операції "виключне АБО" (xor) від її входів із

використанням способу, названого зворотним розповсюдженням. Структура мережі із зворотним розповсюдженням (відома також як мережа, заснована на узагальненому дельта-правилі) представлена на рис. 6.

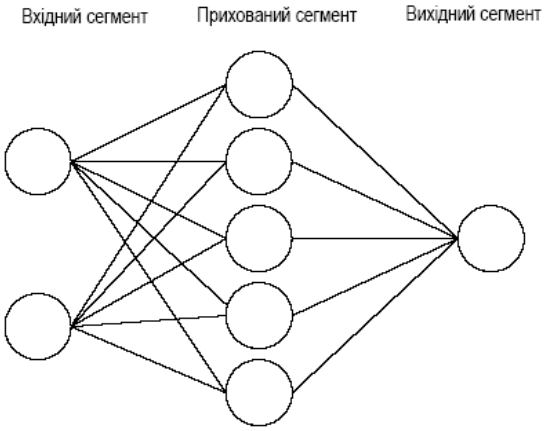

Рис. 6. Мережа із зворотним розповсюдженням

Мережа із зворотним розповсюдженням, зазвичай, поділяється на три сегменти, хоча можуть бути сформовані також додаткові сегменти. Сегменти (сегмент), що знаходяться між вхідним та вихідним сегментами, називаються прихованими сегментами, оскільки зовнішній світ сприймає наочно тільки вхідний і вихідний сегменти. Мережа, що обчислює значення логічної операції "виключне АБО", видає на вихід істинне значення, тільки у випадках, коли не на всіх її входах є істинні значення або не на всіх входах є помилкові значення. Кількість вузлів у прихованому секторі може змінюватись, залежно від мети проекту.

Слід зазначити, що нейронні мережі не вимагають програмування в звичному значенні цього слова. Для навчання нейронних мереж застосовуються спеціальні алгоритми навчання нейронних мереж, такі як зустрічне розповсюдження та зворотне розповсюдження. Програміст "програмує" мережу, задаючи вхідні дані та відповідні вихідні дані. Мережа навчається, автоматично корегуючи вагові коефіцієнти для синаптичних з'єднань між нейронами.

Вагові коефіцієнти, разом із пороговими значеннями нейронів, визначають характер розповсюдження даних по мережі і, тим самим, задають правильний відгук на дані, що використовуються в процесі навчання. Навчання мережі з метою отримання правильних відповідей може потребувати багато часу. Наскільки багато - залежить від того, яка кількість образів повинна бути засвоєна в ході навчання мережі, а також від можливостей застосовуваних апаратних та допоміжних програмних засобів. Проте після завершення такого навчання мережа здатна надавати відповіді з високою швидкістю.

За своєю архітектурою штучна нейронна система відрізняється від інших обчислювальних систем. У класичній інформаційній системі реалізується можливість з'єднання дискретної інформації з елементами пам'яті. Наприклад, зазвичай, інформаційна система зберігає дані про

конкретний об'єкт у групі суміжних елементів пам'яті. Отже, можливість доступу та маніпулювання даними досягається за рахунок створення взаємно однозначного зв'язку між атрибутами об'єкта та адресами елементів пам'яті, в яких вони записані.

На відміну від таких систем, моделі штучних нейронних систем розробляються на основі сучасних теорій функціонування мозку, згідно з якими інформація представлена в мозку за допомогою вагових коефіцієнтів. При цьому безпосередньої кореляції між конкретним значенням вагового коефіцієнта і конкретним елементом збереженої інформації не існує.

Таке розподілене представлення інформації аналогічне технології збереження та представлення зображень, яка використовується в голограмах. Згідно з цією технологією лінії голограми діють як дифракційні решітки. За їх допомогою під час проходження лазерного променя відтворюється збережене зображення, проте самі дані не піддаються безпосередній інтерпретації.

Нейронна мережа виступає в ролі прийнятного засобу рішення задачі, коли присутня велика кількість емпіричних даних, але немає алгоритму, який був би спроможний забезпечити отримання достатньо точного рішення з необхідною швидкістю. В даному контексті технологія представлення даних штучної нейронної системи має суттєві переваги перед іншими інформаційними технологіями. Ці переваги можна сформулювати таким чином:

1. Пам'ять нейронної мережі є відмовостійкою. При видаленні окремих частин нейронної мережі відбувається лише зниження якості інформації, що в ній зберігається, але не повне її зникнення. Це відбувається тому, що інформація зберігається в розподіленій формі.

2. Якість інформації в нейронній мережі, яка підлягає скороченню, знижується поступово, пропорційно тій частині мережі, що була видалена. Катастрофічної втрати інформації не відбувається.

3. Дані в нейронній мережі зберігаються природним чином за допомогою асоціативної пам'яті. Асоціативною пам'яттю називають таку пам'ять, в якій достатньо виконати пошук частково представлених даних, щоб повністю відновити всю інформацію. У цьому полягає відмінність асоціативної пам'яті від звичайної пам'яті, в якій отримання даних здійснюється шляхом вказівки точної адреси відповідних елементів пам'яті.

4. Нейронні мережі дозволяють виконувати екстраполяцію та інтерполяцію на основі інформації, що зберігається в них. Тобто навчання дозволяє додати мережі здатності здійснювати пошук важливих особливостей або зв'язків в даних. Після цього мережа в змозі екстраполювати та виявляти зв'язки в нових даних, що до неї надходять. Наприклад, в одному експерименті було проведене навчання нейронної мережі на гіпотетичному прикладі. Після закінчення навчання мережа набула здатності правильно відповідати на питання, відносно яких навчання не проводилося.

5. Нейронні мережі - пластичні. Навіть після видалення певної кількості нейронів може бути проведене повторне навчання мережі до її первинного рівня (звісно, якщо в ній залишилася достатня кількість нейронів). Така

особливість є також характерною для мозку людини, в якому можуть бути пошкоджені деякі частини, але з часом, за допомогою навчання, досягнуто первинного рівня навичок та знань.

Завдяки таким особливостям штучні нейронні системи стають дуже привабливими для застосування в роботизованих космічних апаратах, устаткуванні нафтопромисловості, підводних апаратах, засобах управління технологічними процесами та в інших технічних пристроях, які повинні функціонувати тривалий час без ремонту в несприятливому середовищі. Штучні нейронні системи не тільки дозволяють розв'язати проблему надійності, але й надають можливість зменшити експлуатаційні витрати завдяки своїй пластичності.

Проте в цілому штучні нейронні системи не зовсім підходять для створення додатків, в яких потрібні складні математичні розрахунки або пошук оптимального рішення. Крім того, застосування штучної нейронної системи не буде найкращим варіантом у випадку, якщо існує алгоритмічне рішення, яке вже надало позитивний результат внаслідок практичного застосування для рішення подібних задач.

7. Практичне спрямування нейронних мереж

Розробка штучних нейронних мереж почалася з досліджень математичного моделювання нейронів, проведених вченими Маккалло-хом та Піттсом у 1943 році. Пояснення процесу навчання за допомогою нейронів, запропоноване вченим Хеббом (Hebb) у 1949 році. У хеббівському навчанні ефективність активізації одних нейронів іншими зростає відповідно до збільшення кількості запусків. Термін "запуск" означає, що нейрон випускає електрохімічний імпульс, здатний стимулювати інші, пов'язані з ним нейрони.

Підвищення ефективності активізації свідчить про те, що провідність з'єднань між нейронами на ділянках їх сполучення - синапсів -зростає із збільшенням кількості запусків. У штучній нейронній системі для моделювання змін провідності синапсів природних нейронів застосовується спосіб коректування вагових коефіцієнтів з'єднань між нейронами.

У 1961 році Розенблатт оприлюднив свою книгу, що значно вплинула на подальший хід досліджень нейронних систем. У ній розглядався різновид штучної нейронної системи, що отримав назву "пер-септрон". Персептрон складається з двох сегментів нейронів і дає можливість використовувати простий алгоритм навчання. Вагові коефіцієнти в ньому повинні встановлюватися вручну, на відміну від сучасних штучних нейронних систем, які здатні самостійно встановлювати власні вагові коефіцієнти на основі навчання.

Рання епоха дослідження персептронів завершилася в 1969 році, коли вчені Мінський (Minsky) і Пейперт (Papert) оприлюднили книгу "Perceptrons", в якій показали теоретичні обмеження персептрона, що досі розглядався як обчислювальна машина загального призначення. Автори роботи підкреслили той факт, що персептрони здатні обчислювати тільки 14 з 16 основних

логічних функцій, тому мають критичні недоліки. Це означало, що персептрон не можна вважати обчислювальним пристроєм загального призначення.

Через ці відкриття бюджетне фінансування досліджень у сфері штучних нейронних систем було скорочене на користь символічного підходу до розробки штучного інтелекту з використанням таких мов, як LISP. У 1970-х роках набувають широкого поширення нові способи представлення символічної інформації в розробках штучного інтелекту на основі фреймів, запропоновані Мінським. Надалі, на основі фреймів, був створений сучасний підхід, в якому застосовуються сценарії.

На початку 1980-х років вчені, які складали групу з дослідження можливостей проведення паралельних обчислень, виявили зацікавленість до теорій пізнання, заснованих на нейронних мережах. У результаті, Хопфілд (Hopfield) розробив надійну теоретичну підставу застосування штучних нейронних систем на прикладі так званої мережі Хопфілда. З її допомогою вчений довів, що штучні нейронні системи можуть успішно вирішувати найрізноманітніші задачі. Зокрема, Хоп-філд показав, що за допомогою штучної нейронної системи задачу комівояжера можна вирішувати за одну годину, тоді як у звичних ал-

горитмічних рішеннях відбувається комбінаторний вибух. Загальна структура мережі Хопфілда наведена на рис. 3.

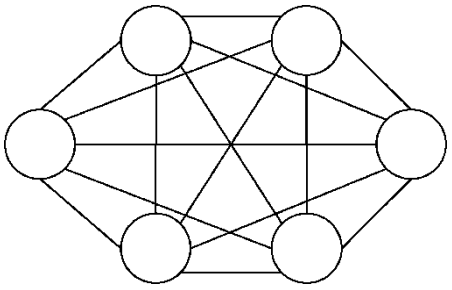

Рис. 3. Штучна нейронна мережа Хопфілда

Ще одним широко відомим типом штучної нейронної системи є мережа із зустрічним розповсюдженням, запропонована Хехтом-Нілсеном (Hecht-Nielsen) у 1986 році. З її допомогою було встановлено, що один з важливих теоретичних результатів у математиці - теорему Колмогорова - можна інтерпретувати як доказ того, що трисегментна мережа з n входами та 2 нейронами в прихованому сегменті дозволяє представити будь-яку безперервну функцію.

Важливий приклад ефективного навчання за допомогою зворотного розповсюдження був продемонстрований за допомогою нейрон-ної мережі, яка навчалася правильній вимові слів, представлених у вигляді тексту. Ця штучна нейронна система навчалася шляхом коректування свого виходу за допомогою пристрою перетворення тексту в мову, що має назву DECTalk.

Для розробки правил грамотної вимови, які були застосовані в пристрої DECTalk, знадобилося близько двадцяти років лінгвістичних досліджень. На відміну від нього, штучна нейронна система засвоїла еквівалентні навички

вимови за одну добу, після прослуховування правильної вимови словосполучень під час читання тексту. При цьому в цій штучній нейронній системі не було закладено шляхом програмування жодних первинних лінгвістичних навичок.

Штучні нейронні системи були також застосовані для розпізнавання радарних цілей за допомогою комп'ютерів. Існує перспектива створення на основі нових реалізацій нейронних мереж із застосуванням оптичних компонентів, нового покоління обчислювальної техніки - оптичних комп'ютерів. За прогнозами, їх швидкодія буде здатна перевищити швидкодію сучасних електронних комп'ютерів у мільйони разів.

Привабливість реалізацій штучних нейронних систем на основі оптичних компонентів обумовлена тим, що світло характеризується властивістю паралельного розповсюдження. Це означає, що не відбувається взаємного впливу між променями світла, що спільно розповсюджується. Крім того, такі оптичні компоненти, як дзеркала, лінзи, високошвидкісні програмовані модулятори світла, масиви оптичних пристроїв та дифракційні решітки дозволяють достатньо легко створювати та здійснювати маніпуляції з великими кількостями фотонів.

Штучні нейронні системи можуть бути також покладені в основу експертних систем. Зокрема, в одній із відомих експертних систем штучна нейронна система є базою знань у сфері діагностики захворювань, що була сформована шляхом навчання на основі прикладів, взятих із практичної медицини. Ця експертна система робить спроби розпізнати захворювання, використовуючи як вхідні дані його симптоми.

Для того, щоб використовувати базу знань на основі штучної нейронної системи, була спроектована машина логічного висновку, яка має назву MACIE (Matrix Controlled Inference Engine). У цій системі використовується прямий логічний висновок - для формування логічних висновків, та зворотний логічний висновок - для передачі користувачу запитів на надання додаткових даних, потрібних для знаходження рішень. Машина MACIE здатна інтерпретувати відповіді штучної нейронної системи і виробляти правила IF-THEN, що застосовуються для пояснення її знань.

У такій експертній системі на основі штучної нейронної системи використовується індуктивне навчання. Це означає, що система логічно формує, за допомогою прикладів, інформацію, що міститься в її базі знань. Нагадаємо, що індукція - це процес логічного виведення загального висновку із проміжних.

Існує значна кількість комерційних програмних засобів знаходження рішень, що дозволяють явно виробляти правила на основі прикладів. Індуктивне навчання використовується для усунення вузьких місць, пов'язаних із набуттям знань. Все навантаження набуття знань покладається на експертну систему. Завдяки цьому з'являється можливість скорочувати час розробки та підвищувати надійність інформаційної системи, здатної логічним шляхом виводити правила, які до тих пір не були відомі.