Тема 9. Методи дерев рішень, класифікації та прогнозування

План

1. Метод дерев рішень.
2. Переваги дерев рішень.
3. Алгоритми.
4. Метод опорних векторів.
5. Лінійний SVM.
6. Метод «найближчого сусіда».
7. Байєсовська класифікація.

Мета вивчення теми: вивчити методи прогнозування та класифікації; засвоїти поняття дерева рішень; вивчити метод опорних векторів; засвоїти метод найближчого сусіда; засвоїти поняття байєсовської класифікації.

Перелік ключових слів та понять із теми

Прогнозування, класифікація, метод дерев рішень, метод опорних векторів, метод найближчого сусіда, байєсовська класифікація, алгоритм CART, крос-перевірка

Теоретичні відомості з теми

1. Метод дерев рішень

Метод дерев рішень (decision trees) є одним із найбільш популярних методів розв'язання задач класифікації й прогнозування. Іноді цей метод Data Mining також називають деревами вирішальних правил, деревами класифікації і регресії.

Як видно з останньої назви, за допомогою даного методу розв'язуються задачі класифікації й прогнозування.

Якщо залежна, тобто цільова змінна приймає дискретні значення, за допомогою методу дерева рішень розв'язується задача класифікації.

Якщо ж залежна змінна приймає безперервні значення, то дерево рішень установлює залежність цієї змінної від незалежних змінних, тобто розв'язує задачу чисельного прогнозування.

У найбільш простому вигляді дерево рішень – це спосіб показу правил в ієрархічній, послідовній структурі. Основа такої структури – відповіді «Так» або «Ні» на низку питань.

Наведемо приклад дерева рішень, задача якого – відповістити на запитання: «Чи грати в гольф?» Щоб розв'язати задачу, тобто прийняти рішення, чи грати в гольф, слід віднести поточну ситуацію до одного з відомих класів (у цьому випадку – «грати» або «не грати»). Для цього потрібно відповісти на низку питань, які є у вузлах цього дерева, починаючи з його кореня.

Перший вузол нашого дерева «Сонячно?» є вузлом перевірки, тобто умовою. При позитивній відповіді на запитання здійснюється перехід до

лівої частини дерева, що називається лівою гілкою, при негативному – до правої частини дерева. Отже, внутрішній вузол дерева є вузлом перевірки певної умови. Далі йде наступне питання і т.д., поки не буде досягнутий кінцевий вузол дерева, що є вузлом розв'язку. Для нашого дерева існує два типи кінцевого вузла: «грати» і «не грати» у гольф.

У результаті проходження від кореня дерева (іноді називається кореневою вершиною) до його вершини розв'язується задача класифікації, тобто вибирається один із класів – «грати» чи «не грати» у гольф**.**

Метою побудови дерева рішень 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?»
- Листок, кінцевий вузол дерева, вузол розв'язку або вершина: «Грати», «Не грати». Гілки дерева (варіанти відповіді): «Так», «Ні».

У розглянутому прикладі розв'язується задача бінарної класифікації, тобто створюється дихотомічна класифікаційна модель. Приклад демонструє роботу так званих бінарних дерев.

У вузлах бінарних дерев розгалуження може відбуватися тільки у двох напрямках, тобто існує можливість тільки двох відповідей на поставлене питання («так» і «ні»).

Бінарні дерева є найпростішим, частковим випадком дерев рішень. В інших випадках, відповідей і, відповідно, гілок дерева, що виходять із його внутрішнього вузла, може бути більше двох.

Розглянемо більш складний приклад.

База даних, на основі якої повинне здійснюватися прогнозування, містить такі ретроспективні дані про клієнтів банку, що є її атрибутами:

- вік,
- наявність нерухомості,
- освіта,
- середньомісячний дохід,
- чи повернув клієнт вчасно кредит.

Задача полягає в тому, щоб на підставі перерахованих вище даних (крім останнього атрибута) визначити, чи варто видавати кредит новому клієнтові.

Як ми вже розглядали в лекції, присвяченій задачі класифікації, така задача розв'язується у два етапи:

- побудова класифікаційної моделі
- її використання.

На етапі побудови моделі, власне, і будується дерево класифікації або створюється набір якихось правил.

На етапі використання моделі побудоване дерево, або шлях від його кореня до однієї з вершин, 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 :».

Наведемо приклад дерева класифікації, за допомогою якого розв'язується задача «Чи видавати кредит клієнтові?». Вона є типовою задачею класифікації, і за допомогою дерев рішень одержують досить хороші варіанти її розв'язку.

Як ми бачимо, внутрішні вузли дерева (вік, наявність нерухомості, дохід і освіта) є атрибутами описаної вище бази даних.

Ці атрибути називають прогнозуючими, або атрибутами розщеплення (splitting attribute). Кінцеві вузли дерева, або листки, іменуються мітками класу, що є значеннями залежної категоріальної змінної «видавати» або «не видавати» кредит.

Кожна гілка дерева, що йде від внутрішнього вузла, відзначена предикатом розщеплення. Останній може відноситися лише до одного атрибуту розщеплення даного вузла.

Характерна риса предикатів розщеплення: кожний запис використовує унікальний шлях від кореня дерева тільки до одного вузла-розв'язку. Об'єднана інформація про атрибути розщеплення й предикати розщеплення у вузлі називається критерієм розщеплення (splitting criterion).

Наприклад, критерій розщеплення «Яка освіта?», міг би мати два предикати розщеплення й виглядати інакше: освіта «вища» і «не вища». Тоді дерево рішень мало б інший вигляд.

Отже, для цієї задачі (як і для будь-якої іншої) може бути побудовано множина дерев рішень різної якості, з різною прогнозуючою точністю.

Якість побудованого дерева рішень досить сильно залежить від правильного вибору критерію розщеплення. Над розробкою й удосконаленням критеріїв працюють багато дослідників.

Метод дерев рішень часто називають «наївним» підходом. Але завдяки певній низці переваг, цей метод є одним із найбільш популярних для розв'язання задач класифікації.

2. Переваги дерев рішень

Розглянемо влесне ці переваги.

Інтуїтивність дерев рішень. Класифікаційна модель, представлена у вигляді дерева рішень, є інтуїтивною і спрощує розуміння розв'язуваної задачі.

Результат роботи алгоритмів конструювання дерев рішень, *на відміну, наприклад, від нейронних мереж, що представляють собою «чорні ящики»,* легко інтерпретується користувачем. Ця властивість дерев рішень не тільки важлива при віднесенні до певного класу нового об'єкта, але й корисна при інтерпретації моделі класифікації в цілому. Дерево рішень дозволяє зрозуміти й пояснити, чому конкретний об'єкт відноситься до того або іншого класу.

Дерева рішень дають можливість витягати правила з бази даних звичайною мовою. Приклад правила: Якщо Вік >35 і Дохід >200, то видати кредит.

Дерева рішень дозволяють створювати класифікаційні моделі в тих сферах, де аналітикові 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 тільки 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 суттєво впливає на час навчання.

Точність моделей, створених за допомогою дерев рішень, порівняно з іншими методами побудови класифікаційних моделей (статистичні методи, нейронні мережі).

Розроблений ряд масштабованих алгоритмів, які можуть бути використані для побудови дерев рішень на надвеликих базах даних. Масштабованість тут означає, що із зростанням кількості прикладів або записів бази даних час, затрачуваний на навчання, тобто побудову дерев рішень, зростає лінійно. Приклади таких алгоритмів: SLIQ, SPRINT.

Швидкий процес навчання. На побудову класифікаційних моделей за допомогою алгоритмів конструювання дерев рішень потрібно значно менше часу, ніж, наприклад, на навчання нейронних мереж.

Більшість алгоритмів конструювання дерев рішень мають можливість спеціальної обробки пропущених значень.

Багато класичних статистичних методів, за допомогою яких розв’язуються задачі класифікації, можуть працювати тільки із числовими даними, у той час як дерева рішень працюють і з числовими, і з категоріальними типами даних.

Багато статистичних методів є параметричними, і користувач повинен заздалегідь володіти певною інформацією, наприклад, знати вид моделі, мати гіпотезу про вид залежності між змінними, припускати, який вид розподілу мають дані. Дерева рішень, на відміну від таких методів, будують непараметричні моделі. Отже, дерева рішень здатні розв’язувати такі задачі Data Mining, у яких відсутня апріорна інформація про вид залежності між досліджуваними даними.

Процес конструювання дерева рішень. Нагадаємо, що розглянута нами задача класифікації відноситься до стратегії навчання з учителем, яке іноді називається індуктивним навчанням. У цих випадках усі об’єкти тренувального набору даних заздалегідь віднесені до одного з визначених класів.

Алгоритми конструювання дерев рішень складається з етапів «побудова» або «створення» дерева (tree building) і «скорочення» дерева (tree pruning). У ході створення дерева вирішуються питання вибору критерію

розщеплення й зупинки навчання (якщо це передбачено алгоритмом). У ході етапу скорочення дерева вирішується питання відсікання деяких його гілок.

Розглянемо ці питання детальніше.

Критерій розщеплення. Процес створення дерева відбувається зверху вниз, тобто є спадним. У ході процесу алгоритм повинен знайти такий критерій розщеплення, іноді також називається критерієм розбивки, щоб розбити множину на підмножини, які б асоціювалися з даним вузлом перевірки. Кожний вузол перевірки повинен бути позначений певним атрибутом. Існує правило вибору атрибута: він повинен розбивати вихідну множину даних таким чином, щоб об'єкти підмножин, що одержуються в результаті цієї розбивки, були представниками одного класу або ж були максимально наближені до такої розбивки. Остання фраза означає, що кількість об'єктів з інших класів, так званих «домішок», у кожному класі прагнула до мінімуму.

Існують різні критерії розщеплення. Найбільш відомі – міра ентропії й індекс Gini.

У деяких методах для вибору атрибута розщеплення використовується так звана міра інформативності підпросторів атрибутів, яка ґрунтується на ентропійному підході й відома за назвою «захід інформаційного виграшу» (information gain measure) або захід ентропії.

Інший критерій розщеплення, запропонований Брейманом (Breiman) та ін., реалізований в алгоритмі CART і називається індексом 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 за формулою:

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2, \qquad (9.1)$$

де *T* – поточний вузол, *pj* – імовірність класу *j* у вузлі *T*, *n* – кількість класів.

Велике дерево не означає, що воно «вдале». Чим більше окремих випадків описано в дереві рішень, тим менша кількість об'єктів потрапляє в кожний окремий випадок. Такі дерева називають «гіллястими» або «кущистими», вони складаються з невиправдано великої кількості вузлів і гілок, вихідна множина розбивається на велику кількість підмножин, що складаються із дуже малого числа об'єктів. У результаті «переповнення» таких дерев їх здатність до узагальнення зменшується, і побудовані моделі не можуть давати вірні відповіді.

У процесі побудови дерева, щоб його розміри не стали надмірно великими, використовують спеціальні процедури, які дозволяють створювати оптимальні дерева, так звані дерева «вдалих розмірів» (Breiman, 1984).

Який розмір дерева може вважатися оптимальним? Дерево повинно бути досить складним, щоб ураховувати інформацію з досліджуваного

набору даних, але одночасно воно повинно бути досить простим. Інакше кажучи, дерево повинно використовувати інформацію, що поліпшує якість моделі, і ігнорувати ту інформацію, яка її не поліпшує.

Отут існує дві можливі стратегії. Перша полягає в нарощуванні дерева до певного розміру відповідно до параметрів, заданих користувачем.

Визначення цих параметрів може ґрунтуватися на досвіді й інтуїції аналітика, а також на деяких «діагностичних повідомленнях» системи, що конструюють дерево рішень.

Друга стратегія полягає у використанні набору процедур, що визначають «вдалий розмір» дерева, вони розроблені Бріманом, Куїлендом та ін. в 1984 році. Однак, як відзначають автори, не можна сказати, що ці процедури доступні починаючому 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 нарощування внутрішніх вузлів.

Зупинка побудови дерева. Розглянемо правило зупинки. Воно повинне визначити, чи є розглянутий вузол внутрішнім вузлом (при цьому він буде розбиватися далі) або ж він є кінцевим вузлом, тобто вузлом розв’язком.

Зупинка – такий момент у процесі побудови дерева, коли слід припинити подальші розгалуження.

Один із варіантів правил зупинки – «рання зупинка» (prepruning), вона визначає доцільність розбивки вузла. Перевага використання такого варіанта – зменшення часу на навчання моделі. Однак тут виникає ризик зниження точності класифікації. Тому рекомендується «замість зупинки використовувати відсікання» (Breiman, 1984).

Другий варіант зупинки навчання – обмеження глибини дерева. У цьому випадку побудова закінчується, якщо досягнута задана глибина.

Ще один варіант зупинки – задання мінімальної кількості прикладів, які будуть утримуватися в кінцевих вузлах дерева. При цьому варіанті розгалуження тривають до того моменту, поки всі кінцеві вузли дерева не будуть чистими або будуть містити не більш ніж задане число об’єктів.

Існує ще ряд правил, але слід зазначити, що жодне з них не має великої практичної цінності, а деякі можуть бути застосовні лише в окремих випадках.

Скорочення дерева або відсікання гілок. Вирішенням проблеми занадто гіллястого дерева є його скорочення шляхом відсікання (pruning) деяких гілок.

Якість класифікаційної моделі, побудованої за допомогою дерева рішень, характеризується двома основними ознаками: точністю розпізнавання й помилкою.

Точність розпізнавання розраховується як відношення об’єктів, правильно класифікованих у процесі навчання, до загальної кількості об’єктів набору даних, які брали участь у навчанні.

Помилка розраховується як відношення об’єктів, неправильно класифікованих у процесі навчання, до загальної кількості об’єктів набору даних, які брали участь у навчанні.

Відсікання гілок або заміну деяких гілок піддеревом слід проводити там, де ця процедура не призводить до зростання помилки. Процес проходить знизу вгору, тобто є висхідним. Це більш популярна процедура, ніж використання правил зупинки. Дерева, одержувані після відсікання деяких гілок, називають усіченими.

Якщо таке усічене дерево усе ще не є інтуїтивним і складне для розуміння, використовують витяг правил, які поєднують у набори для опису класів.

Кожний шлях від кореня дерева до його вершини або листка дає одне правило. Умовами правила є перевірки на внутрішніх вузлах дерева.

3. Алгоритми

Сьогодні існує велика кількість алгоритмів, що реалізують дерева рішень: CART, C4.5, CHAID, CN2, Newid, Itrule і інші. Атрибути набору даних можуть мати як дискретне, так і числове значення.

Алгоритм CART (Classification and Regression Tree), як видно з назви, розв’язує задачу класифікації й регресії. Він розроблений в 1974-1984 роках чотирма професорами статистики – Leo Breiman (Berkeley), Jerry Friedman (Stanford), Charles Stone (Berkeley) і Richard Olshen (Stanford).

Алгоритм CART призначений для побудови бінарного дерева рішень. Бінарні дерева також називають двійковими. Приклад такого дерева розглядався на початку лекції.

Інші особливості алгоритму CART:

- функція оцінки якості розбивки;
- механізм відсікання дерева;
- алгоритм обробки пропущених значень;
- побудова дерев регресії.

Кожний вузол бінарного дерева при розбивці має тільки двох нащадків, що називаються дочірніми галузями. Подальший поділ гілок залежить від того, чи багато вихідних даних описує дана гілка. На кожному кроці побудови дерева правило, формоване у вузлі, ділить задану множину прикладів на дві частини. Права його частина (гілка right) – це та частина множини, у якій правило виконується; ліва (гілка left) – та, для якої правило не виконується.

Функція оцінки якості розбивки, яка використовується для вибору оптимального правила, – індекс Gini – був описаний вище. Відзначимо, що дана оціночна функція заснована на ідеї зменшення невизначеності у вузлі. Допустимо, є вузол, і він розбитий на два класи. Максимальна невизначеність

у вузлі буде досягнута при розбивці його на дві підмножини по 50 прикладів, а максимальна визначеність – при розбивці на 100 і 0 прикладів.

Правила розбивки. Нагадаємо, що алгоритм CART працює із числовими й категоріальними атрибутами. У кожному вузлі розбивка може йти тільки по одному атрибуту. Якщо атрибут є числовим, то у внутрішньому вузлі формується правило виду $x_i \leq c$, Значення c у більшості випадків вибирається як середнє арифметичне двох сусідніх впорядкованих значень змінної x_i навчального набору даних. Якщо ж атрибут відноситься до категоріального типу, то у внутрішньому вузлі формується правило x_i $V(x_i)$, де $V(x_i)$ – деяка непорожня підмножина множин значень змінної x_i у навчальному наборі даних.

Механізм відсікання. Цим механізмом, що має назву minimal cost-complexity tree pruning, алгоритм CART принципово відрізняється від інших алгоритмів конструювання дерев рішень. У розглянутому алгоритмі відсікання – це деякий компроміс між одержанням дерева «підходящого розміру» і одержанням найбільш точної оцінки класифікації. Метод полягає в одержанні послідовності зменшуваних дерев, але дерева розглядаються не всі, а тільки «кращі представники».

Перехресна перевірка (V-fold cross-validation) є найбільш складною й одночасно оригінальною частиною алгоритму CART. Вона являє собою шлях вибору остаточного дерева, за умови, що набір даних має невеликий обсяг або ж записи набору даних настільки специфічні, що розділити набір на навчальну й тестову вибірку не представляється можливим.

Отже, основні характеристики алгоритму CART: бінарне розщеплення, критерій розщеплення – індекс Gini, алгоритми minimal cost-complexity tree pruning і V-fold cross-validation, принцип «виростити дерево, а потім скоротити», висока швидкість побудови, обробка пропущених значень.

Алгоритм C4.5 будує дерево рішень з необмеженою кількістю гілок у вузлах. Даний алгоритм може працювати тільки з дискретним залежним атрибутом і тому може розв’язувати тільки задачу класифікації. C4.5 вважається одним із найвідоміших і широко використовуваних алгоритмів побудови дерев класифікації.

Для роботи алгоритму C4.5 необхідне дотримання таких вимог:

- Кожний запис набору даних повинен бути асоційованим з одним із визначених класів, тобто один з атрибутів набору даних повинен бути міткою класу.
- Класи повинні бути дискретними. Кожний приклад повинен однозначно відноситися до одного із класів.
- Кількість класів повинна бути значно менше кількості записів у досліджуваному наборі даних.

Остання версія алгоритму – алгоритм C4.8 – реалізована в інструменті Weka як J4.8 (Java). Комерційна реалізація методу: C5.0, розроблювач Rulequest, Австралія.

Алгоритм C4.5 повільно працює на надвеликих й зашумлених наборах даних.

Ми розглянули два відомі алгоритми побудови дерев рішень CART і C4.5. Обидва алгоритми є робастними, тобто стійкими до шумів і викидів даних.

Алгоритми побудови дерев рішень відрізняються такими характеристиками:

- вид розщеплення – бінарне (binary), множинне (multi-way);
- критерії розщеплення – ентропія, gini, інші;
- можливість обробки пропущених значень;
- процедура скорочення гілок або відсікання.;
- можливості витягування правил з дерев.

Жоден алгоритм побудови дерева не можна апріорі вважати найкращим або досконалим, підтвердження доцільності використання конкретного алгоритму повинно бути перевірене й підтверджене експериментом.

Розробка нових масштабованих алгоритмів. Найбільш серйозна вимога, яка зараз пред'являється до алгоритмів конструювання дерев рішень – це масштабованість, тобто алгоритм повинен мати масштабований метод доступу до даних.

Розроблений ряд нових масштабованих алгоритмів, серед них – алгоритм Sprint, запропонований Джоном Боярином і його колегами. Sprint, що є масштабованим варіантом розглянутого в лекції алгоритму CART, висуває мінімальні вимоги до об'єму оперативної пам'яті.

4. Метод опорних векторів

Метод опорних векторів (Support Vector Machine – SVM) відноситься до групи граничних методів. Він визначає класи за допомогою границь областей.

За допомогою даного методу розв'язуються задачі бінарної класифікації. В основі методу лежить поняття площин розв'язків. Площина (plane) розв'язку розділяє об'єкти з різною класовою приналежністю.

На рис. 9.1 наведений приклад, у якому беруть участь об'єкти двох типів. Поділяюча лінія задає границю, праворуч від якої – усі об'єкти типу brown (коричневий), а ліворуч – типу yellow (жовтий). Новий об'єкт, що потрапляє праворуч, класифікується як об'єкт класу brown або – як об'єкт класу yellow, якщо він розташувався ліворуч від поділяючої прямої. У цьому випадку кожний об'єкт характеризується двома вимірами.

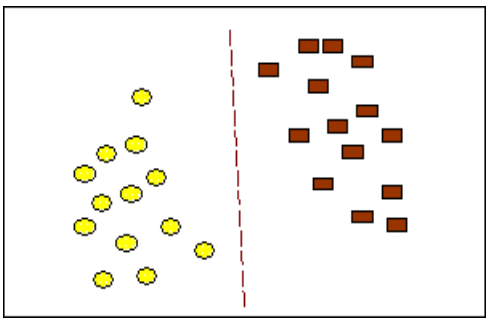

Рисунок 9.1 – Поділ класів прямою лінією

Ціль методу опорних векторів – знайти площину, що розділяє дві множини об’єктів; така площина показана на рис. 9.2. На цьому рисунку множина зразків поділена на два класи: жовті об’єкти належать класу A, коричневі – класу B.

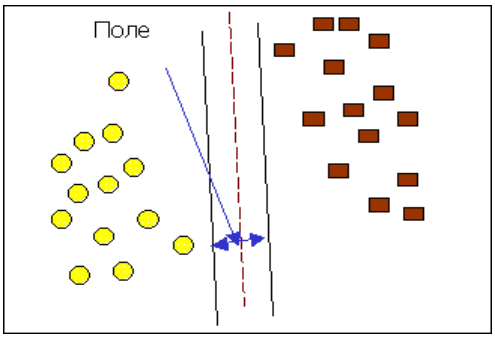

Рисунок 9.2 – До визначення опорних векторів

Метод відшукує зразки, що перебувають на границях між двома класами, тобто опорні вектори. П’ять векторів, які є опорними для даної множини, зображені на 9.3.

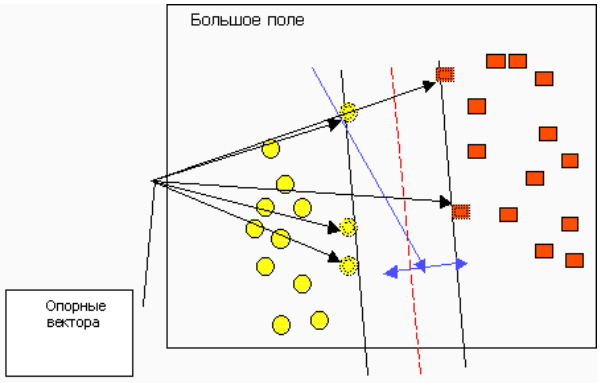

Рисунок 9.3 – Опорні вектори

Опорними векторами називаються об’єкти множини, що знаходяться на границях областей. Класифікація вважається гарною, якщо область між границями порожня.

5. Лінійний SVM

Розв’язання задачі бінарної класифікації за допомогою методу опорних векторів полягає в пошуку деякої лінійної функції, яка правильно розділяє набір даних на два класи. Розглянемо задачу класифікації, де число класів рівне двом.

Задачу можна сформулювати як пошук функції *f(x)*, що приймає значення менше нуля для векторів одного класу й більше нуля – для векторів іншого класу. Як вихідні дані для розв’язання поставленої задачі, тобто пошуку функції *f(x)*, що класифікує, надано тренувальний набір векторів простору, для яких відома їхня приналежність до одного із класів. Сімейство

функцій, що класифікують, можна описати через функцію *f(x)*. Гіперплощина визначена вектором *a* і значенням *b*, тобто *f(x)=ax+b*. Розв'язок даної задачі проілюстрований на рис. 9.4.

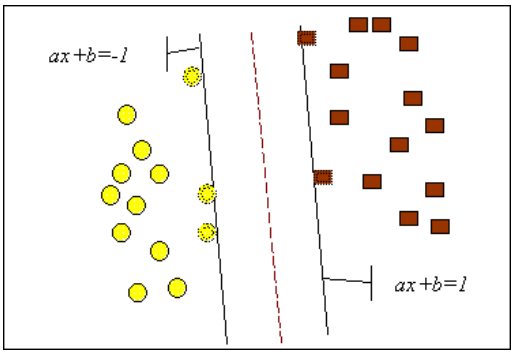

Рисунок 9.4 – Лінійний SVM

У результаті розв'язання задачі, тобто побудови SVM-моделі, знайдена функція, що приймає значення менше нуля для векторів одного класу й більше нуля – для векторів іншого класу. Для кожного нового об'єкта негативне або позитивне значення визначає приналежність об'єкта до одного із класів.

Найкращою функцією класифікації є функція, для якої очікуваний ризик мінімальний. Поняття очікуваного ризику в цьому випадку означає очікуваний рівень помилки класифікації.

Прямо оцінити очікуваний рівень помилки побудованої моделі неможливо, це можна зробити за допомогою поняття емпіричного ризику. Однак слід прийняти, що мінімізація останнього не завжди приводить до мінімізації очікуваного ризику. Це слід пам'ятати при роботі з відносно невеликими наборами тренувальних даних.

Емпіричний ризик – рівень помилки класифікації на тренувальному наборі.

Отже, у результаті розв'язання задачі методом опорних векторів для лінійно поділюваних даних ми одержуємо функцію класифікації, яка мінімізує верхню оцінку очікуваного ризику.

Однією з проблем, пов'язаних із розв'язком задач класифікації розглянутим методом, є та обставина, що не завжди можна легко знайти лінійну границю між двома класами.

У таких випадках один із варіантів – збільшення розмірності, тобто перенесення даних із площини в тривимірний простір, де можливо побудувати таку площину, яка ідеально розділить множину зразків на два класи. Опорними векторами в цьому випадку будуть служити об'єкти з обох класів, що є екстремальними.

Отже, за допомогою додавання так званого оператора ядра й додаткових розмірностей, знаходяться границі між класами у вигляді гіперплощин.

Однак слід пам’ятати: складність побудови SVM-моделі полягає в тому, що чим вища розмірність простору, тим складніше з ним працювати. Один із варіантів роботи з даними високої розмірності – це попереднє застосування якого-небудь методу зниження розмірності даних для виявлення найбільш істотних компонентів, а потім використання методу опорних векторів.

Як і будь-який інший метод, метод SVM має свої сильні й слабкі сторони, які слід враховувати при виборі цього методу.

Недолік методу полягає в тому, що для класифікації використовується не вся множина зразків, а лише їхня невелика частина, яка перебуває на границях.

Перевага методу полягає в тому, що для класифікації методом опорних векторів, на відміну від більшості інших методів, достатньо невеликого набору даних. При правильній роботі моделі, побудованої на тестовій множині, цілком можливе застосування даного методу на реальних даних.

Метод опорних векторів дозволяє:

- одержати функцію класифікації з мінімальною верхньою оцінкою очікуваного ризику (рівня помилки класифікації);
- використовувати лінійний класифікатор для роботи з нелінійно поділюваними даними, поєднуючи простоту з ефективністю.

6. Метод «найближчого сусіда»

Метод «найближчого сусіда» або системи міркувань на основі аналогічних випадків.

Слід відразу зазначити, що метод «найближчого сусіда» («nearest neighbour») відноситься до класу методів, робота яких ґрунтується на зберіганні даних у пам’яті для порівняння з новими елементами. Із появою нового запису для прогнозування мають місце відхилення між цим записом і подібними наборами даних, та ідентифікується найбільш подібний (або близький сусід).

Наприклад, при розгляді нового клієнта банку, його атрибути порівнюються з усіма існуючими клієнтами даного банку (дохід, вік і т.д.). Множина «найближчих сусідів» потенційного клієнта банку вибирається на підставі найближчого значення доходу, віку і т.д.

При такому підході використовується термін «*k-найближчий сусід*» («*k-nearest neighbour*»). Термін означає, що вибирається *k* «верхніх» (найближчих) сусідів для їхнього розгляду як множини «найближчих сусідів». Оскільки не завжди зручно зберігати всі дані, іноді зберігається тільки множина «типових» випадків. У такому випадку використовуваний метод називають міркуванням за аналогією (Case Based Reasoning, CBR), міркуванням на основі аналогічних випадків, міркуванням по прецедентах.

Прецедент – це опис ситуації в комбінації з докладною вказівкою дій, що застосовують у даній ситуації.

Підхід, заснований на прецедентах, умовно можна поділити на такі етапи:

- збір докладної інформації про поставлене завдання;
- 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 обраного розв'язку до поточної проблеми, якщо це необхідно;
- перевірка коректності кожного нового отриманого розв'язку;
- занесення детальної інформації про новий прецедент у базу прецедентів.

Отже, висновок, заснований на прецедентах, являє собою такий метод аналізу даних, який робить висновок щодо даної ситуації за результатами пошуку аналогій, що зберігаються в базі прецедентів.

Даний метод за своєю суттю належить до категорії «навчання без вчителя», тобто являється технологією «що навчається самостійно», завдяки чому робочі характеристики кожної бази прецедентів із плином часу і накопиченням прикладів покращуються. Розробка баз прецедентів за конкретною предметною областю відбувається звичайною для людини мовою, отже, може бути виконана найбільш досвідченими співробітниками компанії – експертами або аналітиками, що працюють у цій предметній області.

Однак це не означає, що *CBR*-системи самостійно можуть ухвалювати рішення. Останнє завжди залишається за людиною, цей метод лише пропонує можливі варіанти розв'язку й указує на «найрозумніший» з її точки зору.

Переваги методу:

- Простота використання отриманих результатів.
- Розв'язки не унікальні для конкретної ситуації, можливе їх використання для інших випадків.
- Метою пошуку є не гарантовано вірний розв'язок, а кращий з можливих.

Недоліки методу «найближчого сусіда»:

- Цей метод не створює яких-небудь моделей або правил, що узагальнюють попередній досвід, – у виборі розв'язку вони ґрунтуються на всьому масиві доступних історичних даних, тому неможливо сказати, на якій підставі будуються відповіді.
- Існує складність вибору заходу «близькості» (метрики). Від цього заходу головним чином залежить обсяг множини записів, які потрібно зберігати в пам'яті для досягнення задовільної класифікації або прогнозу. Також існує висока залежність результатів класифікації від обраної метрики.
- При використанні методу виникає необхідність повного перебору навчальної вибірки при розпізнаванні, як наслідок цього – обчислювальна трудомісткість.
- Типові завдання цього методу – це завдання невеликої розмірності за

кількістю класів і змінних.

За допомогою даного методу розв'язуються задачі класифікації й регресії.

Розглянемо докладно принципи роботи методу *k*-найближчих сусідів для розв'язання задач класифікації й регресії (прогнозування).

Розв'язання задачі класифікації нових об'єктів. Ця задача схематично зображена на рис. 9.5. Приклади (відомі екземпляри) відзначені «+» або «-», що визначають приналежність до відповідного класу, а новий об'єкт, який потрібно класифікувати, позначений кружечком. Нові об'єкти також називають точками запиту.

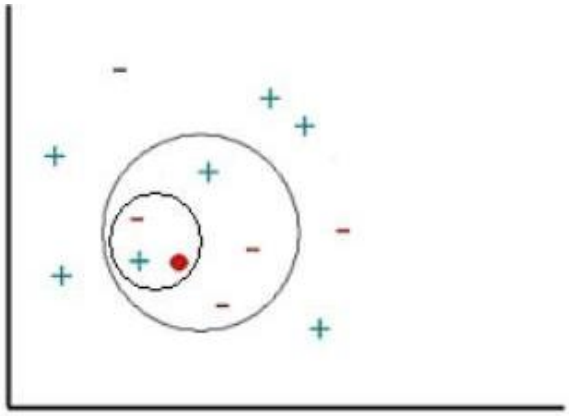

Рисунок 9.5 – Класифікація об'єктів множини при різному значенні параметра *k*

Наша мета полягає в оцінці (класифікації) відгуку точок запиту з використанням спеціально обраного числа їх найближчих сусідів. Інакше кажучи, ми прагнемо довідатися, до якого класу слід віднести точку запиту: знак «+» або знак «-».

Для початку розглянемо результат роботи методу *k*-найближчих сусідів із використанням одного найближчого сусіда. У цьому випадку відгук точки запиту буде класифікований як знак «плюс», тому що найближча сусідня точка має знак «плюс».

Тепер збільшимо число використовуваних найближчих сусідів до двох. Цього разу метод *k*-найближчих сусідів не зможе класифікувати відгук точки запиту, оскільки друга найближча точка має знак «мінус» і обидва 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 графіку її границя відзначена червоним (сірим) колом). Оскільки в околиці утримується 2 точки зі знаком «+» і 3 точки зі знаком «-», алгоритм *k*-найближчих сусідів привласнить знак «-» відгуку точки запиту.

Розв'язання задачі прогнозування. Далі розглянемо принцип роботи методу *k*-найближчих сусідів для розв'язання задачі регресії. Регресійні задачі пов'язані з прогнозуванням значення залежної змінної за значеннями незалежних змінних набору даних.

Розглянемо графік, показаний на рис. 9.6. Зображений на ній набір точок (*зелені* прямокутники) отриманий за зв'язком між незалежною змінною *x* і залежною змінною *y* (*крива червоного кольору*). Заданий набір зелених об'єктів (тобто набір прикладів); застосуємо метод *k*-найближчих сусідів для прогнозування виходу точки запиту *X* за цим набором прикладів (зелені прямокутники).

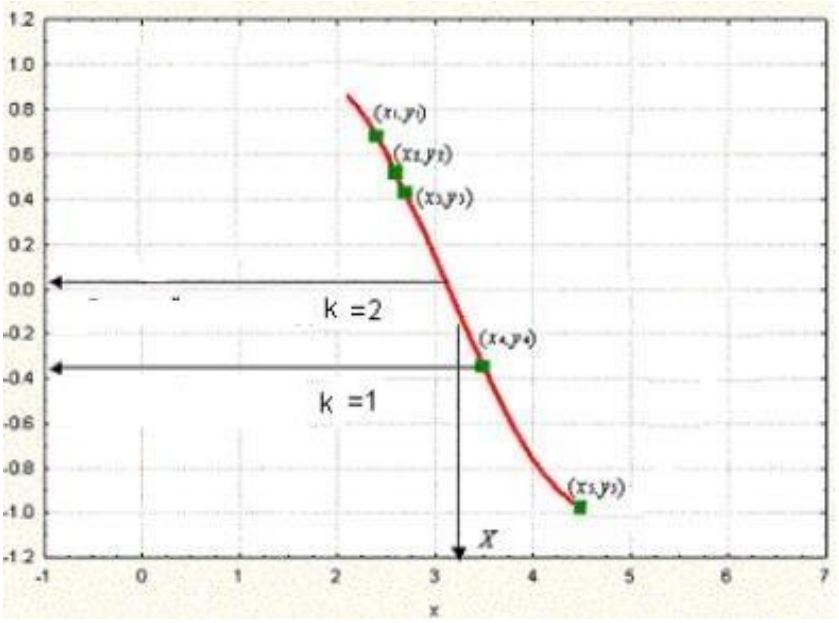

Рисунок 9.6 – Розв'язок задачі прогнозування при різних значеннях параметра *k*

Спочатку розглянемо як приклад метод *k*-найближчих сусідів з використанням одного найближчого сусіда, тобто при *k*, рівному одиниці. Шукаємо набір прикладів (зелені прямокутники) і виділяємо з їхнього числа найближчий до точки запиту *X*. Для цього випадку найближчий приклад – точка (x_4; y_4). Вихід x_4, таким чином, приймається як результат прогнозування виходу *X*. Отже, для одного найближчого сусіда можемо записати: вихід *Y* рівний y_4 ($Y = y_4$).

Далі розглянемо ситуацію, коли *k* рівне двом, тобто розглянемо два найближчі сусіди. У цьому випадку виділяємо вже дві найближчі до *X* точки. На цьому графіку це точки y_3 і y_4 відповідно. Обчисливши середнє їхніх виходів, записуємо розв'язок для *Y* у вигляді $Y = (y_3 + y_4)/2$.

Розв'язання задачі прогнозування здійснюється шляхом перенесення описаних вище дій на використання довільного числа найближчих сусідів таким чином, що вихід *Y* точки запиту *X* обчислюється як середньоарифметичне значення виходів *k*-найближчих сусідів точки запиту.

Незалежні й залежні змінні набору даних можуть бути як безперервними, так і категоріальними. Для безперервних залежних змінних задача розглядається як задача прогнозування, для дискретних змінних – як задача класифікації.

Прогнозування в задачі прогнозування виходить усередненням виходів *k*-найближчих сусідів, а розв'язок задачі класифікації заснований на принципі «за більшістю голосів».

Критичним моментом у використанні методу *k*-найближчих сусідів є вибір параметра *k*. Він один із найбільш важливих факторів, що визначають якість прогнозної або класифікаційної моделі.

Якщо обрано занадто мале значення параметра *k*, виникає ймовірність великого розкиду значень прогнозу. Якщо обране значення занадто велике, це може призвести до суттєвого зміщення моделі. Отже, повинно бути обране оптимальне значення параметра *k*. Тобто це значення повинно бути настільки великим, щоб звести до мінімуму ймовірність неправильної класифікації, і одночасно, досить малим, щоб *k* сусідів були розташовані досить близько до точки запиту.

Отже, розглядаємо *k* параметр, як згладжуючий, для якого потрібно знайти компроміс між силою розмаху (розкиду) моделі та її зміщеностю.

Один із варіантів оцінки параметра *k* – проведення крос-перевірки (Bishop, 1995). Така процедура реалізована, наприклад, у пакеті Statistica (Statsoft).

Крос-перевірка – відомий метод одержання оцінок невідомих параметрів моделі. Основна ідея методу – поділ вибірки даних на *v* «складок». *V* «складки» тут є випадковим чином виділені ізольовані підвибірки.

За фіксованим значенням *k* будується модель *k*-найближчих сусідів для одержання прогнозів на *v*-му сегменті (інші сегменти при цьому використовуються як приклади) і оцінюється помилка класифікації. Для регресійних задач найбільш часто за оцінку помилки виступає сума квадратів, а для класифікаційних задач зручніше розглядати точність (відсоток коректно класифікованих спостережень).

Далі процес послідовно повторюється для всіх можливих варіантів вибору *v*. По вичерпанню *v* «складок» (циклів), обчислені помилки усереднюються й використовуються як міра стабільності моделі (тобто міра якості прогнозування в точках запиту). Вищеописані дії повторюються для різних *k*, і значення, що відповідає найменшій помилці (або найбільшій класифікаційній точності), приймається як оптимальне (оптимальне в сенсі методу крос-перевірки).

Слід враховувати, що крос-перевірка – ємнісна з точки зору обчислень процедура, і необхідно надати час для роботи алгоритму, особливо якщо обсяг вибірки досить великий.

Другий варіант вибору значення параметра *k* – самостійно задати його значення. Однак цей спосіб слід використовувати, якщо є обґрунтовані припущення щодо можливого значення параметра, наприклад, про попередні дослідження подібних наборів даних.

Метод *k*-найближчих сусідів показує досить непогані результати в найрізноманітніших задачах.

Прикладом реального використання описаного вище методу є програмне забезпечення центру технічної підтримки компанії Dell, розроблене компанією Inference. Ця система допомагає співробітникам центру відповідати на велике число запитів, відразу пропонуючи відповіді на розповсюджені питання й 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 реалізують метод *k*-найближчих сусідів і *CBR*-метод, не дуже 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.

7. Байєсовська класифікація

Теоре́ма Ба́йєса – одна з основних теорем елементарної теорії ймовірностей, яка дозволяє визначити ймовірність якої-небудь події за умови, що сталася інша статистично взаємозалежна з нею подія. Іншими словами, за формулою Байєса можна більш точно перерахувати ймовірність, взявши до уваги раніше відому інформацію і дані нових спостережень. Формула Байєса може бути виведена з основних аксіом теорії ймовірностей, зокрема з умовної ймовірності. Особливість теореми Байєса полягає в тому, що для її практичного застосування потрібна велика кількість розрахунків, обчислень, тому байєсовські оцінки стали активно використовувати тільки після революції в комп'ютерних та мережевих технологіях. Теорема використовується не тільки в аналізі ймовірностей, а й активно застосовується для безлічі інших розрахунків. Психологічні експерименти показали, що люди часто невірно оцінюють ймовірність події, на основі отриманого досвіду (апостеріорна ймовірність), оскільки ігнорують саму ймовірність припущення (апріорна ймовірність). Тому правильний результат за формулою Байєса може суттєво відрізнятися від інтуїтивно очікуваного.

Формула Байєса:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \qquad (9.2)$$

де $P(A)$ – апріорна ймовірність гіпотези *A*;

$P(A|B)$ – ймовірність гіпотези *A* при настанні події *B* (апостеріорна ймовірність);

$P(B|A)$ – ймовірність настання події *B* при істинності гіпотези *A*;

$P(B)$ – ймовірність настання події *B*.

Формула виводиться із визначення умовної ймовірності:

$$P(A|B) = \frac{P(A \cup B)}{P(B)} \rightarrow P(A \cup B) = P(A|B)P(B) = P(B|A)P(A).$$

Байєсовський класифікатор – широкий клас алгоритмів класифікації, заснований на принципі максимуму апостеріорної ймовірності. Для класифікованого об'єкта обчислюються функції правдоподібності кожного з класів, за ними обчислюються апостеріорні ймовірності класів. Об'єкт відноситься до того класу, для якого апостеріорна ймовірність максимальна.

Апостеріорна ймовірність – умовна ймовірність випадкової події за умови того, що відомі апостеріорні дані, тобто отримані після досліду.

Альтернативні назви: байєсовське моделювання, байєсовська статистика, метод байєсовських мереж.

Споконвічно байєсовська класифікація використовувалася для формалізації знань експертів в експертних системах, зараз байєсовська класифікація також застосовується як один із методів Data Mining.

Так звана наївна класифікація або наївно-байєсовський підхід (naive-bayes approach) є найбільш простим варіантом методу, що використовує байєсовські мережі. При цьому підході вирішуються задачі класифікації, результатом роботи методу є так звані «прозорі» моделі.

«Наївна» класифікація – досить прозорий і зрозумілий метод класифікації. «Наївною» вона називається тому, що виходить із припущення про взаємну незалежність ознак.

Властивості наївної класифікації:

1. Використання всіх змінних і визначення всіх залежностей між ними.
2. Наявність двох припущень щодо змінних:

– усі змінні є однаково важливими;

– усі змінні є статистично незалежними, тобто значення однієї змінної нічого не говорить про значення іншої.

Більшість інших методів класифікації припускають, що перед початком класифікації ймовірність того, що об'єкт належить тому або іншому класу, однакова; але це не завжди правильно.

Допустимо, відомо, що певний відсоток даних належить конкретному класу. Виникає питання, чи можна використати цю інформацію при побудові моделі класифікації? Існує множина реальних прикладів використання цих апріорних знань, що допомагають класифікувати об'єкти. Типовий приклад із медичної практики. Якщо лікар відправляє результати аналізів пацієнта на додаткове дослідження, він відносить пацієнта до якогось певного класу. Яким чином можна застосувати цю інформацію? Можна використати її як додаткові дані при побудові класифікаційної моделі.

Відзначають такі переваги байєсовських мереж як методу Data Mining:

- у моделі визначаються залежності між усіма змінними, це дозволяє легко обробляти ситуації, в яких значення деяких змінних невідомі;
- байєсовські мережі досить просто інтерпретуються й дозволяють на

етапі прогностичного моделювання легко проводити аналіз за сценарієм «що, якщо»;

- байєсовський метод дозволяє природно поєднувати закономірності, виведені з даних, і, наприклад, експертні знання, отримані в явному вигляді;
- використання байєсовських мереж дозволяє уникнути проблеми переучування (overfitting), тобто надлишкового ускладнення моделі, що є слабкою стороною багатьох методів (наприклад, дерев рішень і нейронних мереж).

Наївно-байєсовський підхід має такі недоліки:

- перемножувати умовні ймовірності коректно тільки тоді, коли всі вхідні змінні дійсно статистично незалежні; хоча часто даний метод показує досить гарні результати при недотриманні умови статистичної незалежності, але теоретично така ситуація повинна оброблятися більш складними методами, заснованими на навчанні байєсовських мереж;
- неможлива безпосередня обробка безперервних змінних – потрібно їхнє перетворення до інтервальної шкали, щоб атрибути були дискретними; однак такі перетворення іноді можуть приводити до втрати значимих закономірностей.

Питання для самоконтролю

1. Дайте визначення методу дерев рішень (decision trees)? Для чого він використовується?
2. Як використовуються бінарні дерева? Наведіть приклад.
3. Які існують переваги методу дерева рішень?
4. Які існують критерії розщеплення дерева рішень?
5. Які існують варіанти зупинки навчання дерева рішень?
6. Назвіть відомі алгоритми, що реалізують дерева рішень.