МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки

Прикладні методи аналізу даних

навчальний посібник

підготовки _____аспірант_____
назва освітньо-кваліфікаційного рівня

для PHD студентів

Розробник: проф., д. т. н. Новотарський М. А.
(посада, вчений ступінь та звання П.І.Б.)

Затверджено на засіданні кафедри
обчислювальної техніки Протокол
№ 15 від 29.05.2024

Завідувач кафедри ОТ

_____ С. Г. Стіренко
(підпис) (ініціали прізвище)

Київ – 2024

ЗМІСТ

Розділ 1

Вступні положення прикладного аналізу даних

Термінологія.

Мова йде про області науки, які в західній літературі називаються Data Mining та Machine Learning.

Перекладаємо як «Аналіз даних».

Термін не зовсім вдалий, оскільки слово «аналіз» в математиці досить звично, має усталену значення і входить до складу назви багатьох класичних розділів: математичний аналіз, функціональний аналіз і т.д.

Аналіз даних – це, перш за все, прикладна наука, в якій не існує свого специфічного математичного апарату, в тому сенсі, що немає кінцевого набору базових фактів, з яких ми можемо дізнатися, як вирішувати задачі.

Багато задач «індивідуальні», і зараз з'являються все нові і нові класи задач, під які необхідно розробляти математичний апарат. Тут ще більшу роль відіграє той факт, що аналіз даних відносно новий напрямок в науці.

1.1. Загальне поняття про аналіз даних

Аналіз даних в природі

Уникнення приманки

У природі – щури аналізують дані своїх відчуттів, щоб уникати отруйних приманок.

Поїдання їжі малими порціями з наступним аналізом стану. Якщо їжа асоціюється з поганим станом, то виникає передбачення, що наступний прийом їжі викличе поганий стан.

У прикладному аналізі даних – використовують механізм, який називають індуктивне міркування або індуктивний висновок. Певні закономірності поширюють на великі обсяги даних

Марновірство

У природі – голуби можуть помилково пов'язувати годування з своїми діями, які вони виконували саме в цей час.

У прикладному аналізі даних – марновірство призводить до відсутності прогресу в аналізі

Єдина відмінність між уникненням приманки щурами та марновірством голубів є включення ***попередніх знань***, які базуються на механізмі навчання. Це називається індуктивним зміщенням (inductive bias).

1.2. Коли нам потрібен аналіз даних

Існують дві основних причини для створення програм для аналізу даних:

1. Задача, яку необхідно вирішувати має велику складність.
2. При розв'язуванні задачі необхідно використовувати властивість адаптивності.

Складні задачі включають:

1. Задачі, які розв'язують тварини/люди: водіння автомобіля, мова, розпізнавання та розуміння образу та ін.

2. Задачі які знаходяться поза людськими можливостями: астрономічні дані, задачі використання медичних архівів для одержання нових медичних знань, прогноз погоди, аналіз геномних даних, веб-пошукові машини і задачі електронної комерції.

Адаптивні задачі

Програми декодування рукописного тексту, де фіксована програма може адаптуватися до варіацій між почерком різних користувачів; Програми виявлення спаму, автоматично адаптуються до змін у характері спам-листів; програми розпізнавання мови

1.3. Сфери застосування задач прикладного аналізу

•Штучний інтелект
•Комп'ютерне зір
• Розпізнавання мови
• Комп'ютерна лінгвістика та обробка природних мов
• Медична діагностика
• Біоінформатика
• Технічна діагностика
• Фінансові додатки
• Пошук і рубрикація текстів
• Інтелектуальні ігри
• Експертні системи
• Реклама

1.4. Machine Learning vs Data Mining

Data Mining (видобуток даних, інтелектуальний аналіз даних, глибинний аналіз даних) – сукупність методів виявлення в даних раніше невідомих, нетривіальних, практично корисних і доступних для інтерпретації знань, необхідних для прийняття рішень в різних сферах людської діяльності.

[Г.П'ятецький-Шапіро, 1989]

Отже, і ML, і DM витягають закономірності («знання») з даних, але (дещо) з різними цілями:

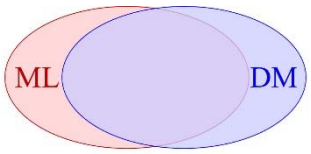
• ML – щоб навчити машину;
• DM – щоб навчити людину.

Тому:

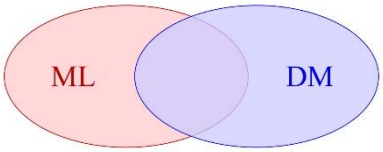
• в ML мінімізують помилку;
• в DM важлива можливість інтерпретації результату.

Machine Learning vs Data Mining

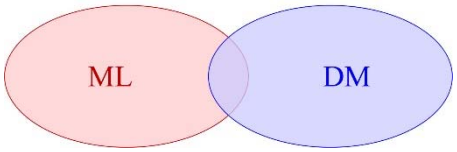
Методи ML DM



Математичні
постановки завдань



Змістовні
постановки завдань



1.5. Класифікація задач машинного навчання

• Дедуктивное навчання (експертні системи).
• Індуктивне навчання
- Навчання з учителем:
* класифікація (classification)
* Відновлення регресії (regression estimation)
* Логічне навчання (knowledge discovery).
*. . .
- Навчання без учителя:
* кластеризація (clustering)
* Візуалізація даних (visualization)
* Зниження розмірності (dimensionality reduction)
*. . .
- Навчання з підкріпленням (reinforcement learning)
- Активне навчання

1.6. Дедуктивний та індуктивний способи навчання

Навчання людей:
1. Ознайомлення з правилами, теоріями, інструкціями і т. п.
2. Отримання досвіду (власного або чужого).
Навчання технічних систем:
1. Дедуктивное, або аналітичне, навчання (експертні системи).
Знання, сформульовані експертом.
Програма виводить конкретні факти і нові правила.
2. Індуктивне навчання.
На основі емпіричних даних програма будує загальне правило.

Емпіричні дані можуть бути отримані самою програмою в попередні сеанси її роботи або просто пред'явлені їй.

1.7. Навчання з учителем та навчання без учителя

Supervised learning

Навчання з учителем є процес «використання досвіду, щоб отримати досвід»

Особливість: «досвід», тобто навчальний приклад, містить метрику важливості інформації (скажімо, мітки спам/не-спам).

«Тести», тобто робочі дані, до яких повинен застосовуватися досвід, метрики не містять.

«Досвід» спрямовують на виявлення невідомої інформації у тестових даних.

Unsupervised learning

Особливість: Початковий «досвід», тобто навчальні приклади, відсутні.

*Під час навчання без учителя не існує різниці між навчальними та тестовими даними.

*Система обробляє вхідні дані з метою складання певної узагальненої або стисненої версії цих даних.

Типовий приклад: Кластеризація набору даних у підмножини подібних об’єктів.

1.8. Навчання з підкріпленням

Reinforcement learning

Це проміжна парадигма навчання

Тренувальний набір містить більше інформації, ніж тестовий набір. Система повинна визначити ще більше інформації для тестових наборів.

Наприклад.

Необхідно створити функцію значень, яка задає для кожної конфігурації фігур на шахівниці, відсоток, на який позиція білих краще за позицію чорних.

Проте, єдина інформація, доступна учням на час навчання – це конфігурації, які відбулися в ході фактичних ігор в шахи, позначені міткою, яка вказує на те, хто згодом виграв цю гру.

1.9. Активне та пасивне навчання

Парадигми навчання можуть відрізнятися залежно від ролі, яку відіграє система, яка навчається.

Ми відрізняємо «активне» та «пасивне» навчання.

При активному навчанні система взаємодіє із навколишнім середовищем під час навчання, скажімо, шляхом створення запитів або проведення експериментів.

При пасивному навчанні система лише спостерігає за інформацією, наданою навколишнім середовищем (або вчителем), не впливаючи на неї та не направляючи її.

Приклад. Пасивне навчання. Система спам-фільтрації зазвичай пасивна (очікуючи, що користувачі позначать електронну пошту, яка надходить до них).

Активне навчання. При активній спам-фільтрації можна було б уявити запити до користувача з вимогою позначити певні електронні повідомлення, що викликають сумнів, щоб покращити розуміння того, що таке спам.

1.10. Зв'язок з іншими областями науки

Прикладний аналіз даних використовує положення з математичної статистики, теорії інформації, теорії ігор та оптимізації

Прикладний аналіз даних можна розглядати як галузь AI (Artificial Intelligence).Існує два підходи:

-імітація інтелектуальної діяльності,

-виконання завдань, що не під силу людському розуму.

Основні відмінності від статистики

В статистиці прийнято працювати з попередньо заданими моделями даних (нормальний розподіл даних або лінійність функціональних залежностей).

В прикладному аналізі даних працюють при «free-distribution», де система найчастіше бере на себе припущення щодо характеру розподілу даних і дозволяє алгоритму навчання визначити, які моделі максимально наближають процес створення даних.

Запитання до розділу 1

1. Коли необхідно застосовувати аналіз даних?

2. У чому полягає різниця між аналізом даних та машинним навчанням?

3. Чим відрізняється навчання з учителем від навчання без учителя?

4. Які особливості навчання з підкріпленням?

5. Які особливості активного навчання і чим воно відрізняється від пасивного навчання?

Розділ 2
Загальні поняття про прикладні задачі аналізу

2.1. Прикладні задачі аналізу даних

Задачі класифікації (classification) *Медична діагностика*

Об'єкт: пацієнт на певній стадії захворювання

Ознаки: результати обстежень, симптоми захворювання і методи лікування.

Приклади бінарних ознак: стать, наявність головного болю, слабкості, нудоти, і т. д.

Приклади порядкових ознак: тяжкість стану (задовільний, середньої тяжкості, важкий, вкрай важке).

Приклади кількісних ознаки: вік, пульс, артеріальний тиск, вміст гемоглобіну в крові, доза препарату, і т. д.

Можна вирішувати різні завдання:

класифікувати вид захворювання (диференціальна діагностика);

визначати найбільш доцільний спосіб лікування;

прогнозувати тривалість і результат захворювання;

оцінювати ризик ускладнень;

знаходити синдроми – найбільш характерні для даного захворювання сукупності симптомів.

Цінність такого роду систем в тому, що вони здатні миттєво аналізувати і узагальнювати величезну кількість випадків, що недоступно людині.

Задача оцінювання позичальників

Об'єкт: позичальник – фізична або юридична особи, яка претендують на отримання кредиту.

Класи: хороший або поганий

Бінарні ознаки: стать, наявність телефону.

Номінальні ознаки: місце проживання, професія.

Порядкові ознаки: освіта, посада на роботі.

Кількісні ознаки: вік, стаж роботи, дохід сім'ї, розмір заборгованостей в інших банках, сума кредиту.

Навчальна вибірка: позичальники з відомою кредитною історією. Формування вектора ознак.На стадії навчання проводиться синтез і відбір інформативних ознак і визначається, скільки балів призначати за кожну ознаку, щоб ризик прийнятих рішень був мінімальний.

Наступне завдання: вирішити, на яких умовах видавати кредит:

визначити процентну ставку, термін погашення, інші параметри кредитного договору.

Це завдання також зводиться до статистичного навчання.

Задачі регресійної оцінки (regression estimation)

Походження терміну «регресія»

Термін «регресія» був введений в 1886 році антропологом Френсісом Гальтоном при вивченні статистичних закономірностей спадковості зросту. Повсякденний досвід підказує, що в середньому зріст дорослих дітей тим більше, чим вище їх батьки. Але Гальтон виявив, що сини дуже високих батьків часто мають не такий високий зріст. Він зібрав вибірку даних по 928 парах батько-син. Кількісно залежність непогано описувалась лінійною функцією $y = \frac{2}{3}x$, де $x-$ відхилення зросту батька від середнього, а $y-$ відхилення зросту сина від середнього. Гальтон назвав це явище «регресією до пересічності», тобто, до середнього значення в популяції. Термін «регресія», або рух назад натякав також на нестандартний для того часу хід дослідження: спочатку були зібрані дані, потім по них вгадана модель залежності, тоді як традиційно поступали навпаки: дані використовувалися лише для перевірки теоретичних моделей. Це був один з перших випадків моделювання, заснованого виключно на даних. Пізніше термін, що виник в конкретній прикладній задачі, закріпився за широким класом методів відновлення залежностей. Величезна кількість регресійних задач виникає в фізичних експериментах, в промисловому виробництві, в економіці.

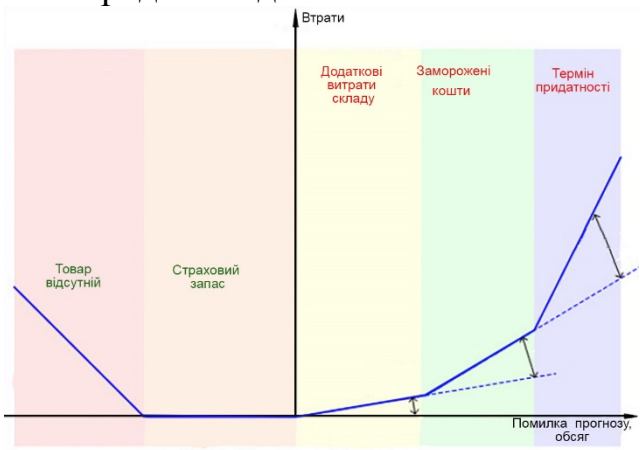
Задача прогнозування споживчого попиту

Об’єкт: кортеж (товар,магазин, день)

Ознаки:

-бінарні: вихідний свято, промоакція;

-кількісні: обсяги продаж по днях.



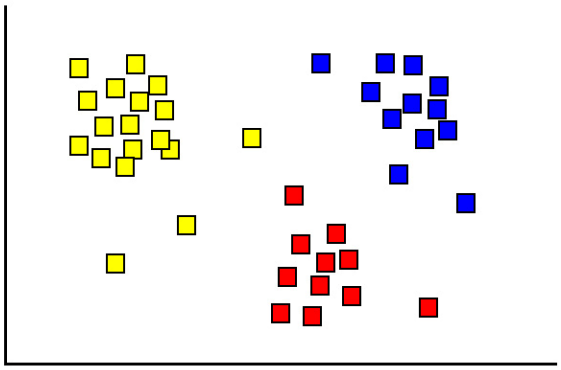
Особливості: складна функція втрат; надвеликі вибірки, проблема вибору ознак

На основі цих прогнозів здійснюється: планування закупівель, управління асортиментом, формування цінової політики, планування промоакцій).

Задачі кластеризації

Задачі кластеризації (clustering) відрізняються від класифікації (classification) тим, що в них не задаються відповіді $y_i = y^*\left(x_i\right)$.

Відомі тільки самі об’єкти x_i, і необхідно розбити вибірку на підмножини (кластери) так, щоб кожен кластер складався з схожих об’єктів, а об’єкти різних кластерів істотно відрізнялися.



Для цього необхідно задати функцію відстані на множині об’єктів. Число кластерів також може задаватися, але частіше потрібно визначити і його.

Задача рубрикації текстів

Об’єкт: текстовий документ

Навчальна вибірка: множина документів, класифікованих за рубриками вручну.

Задача рубрикації текстів

Нехай дано ієрархічний рубрикатор, розроблений експертами для даної предметної області. Потрібно класифікувати за рубриками множину документів, яке може бути істотно більше навчальної.

Вирішення даної задачі

Використовують функцію відстані, що порівнює тексти за складом термінів. Термінами, як правило, є спеціальні поняття предметної області, власні імена, географічні назви, і т. д.

Документи вважаються схожими, якщо множини їх термінів суттєво перетинаються.

Задача соціологічного опитування

З одного боку, щоб результати опитування були об’єктивні, необхідно забезпечити представництво вибірки респондентів.

З іншого боку, потрібно мінімізувати вартість проведення опитування.

Тому при плануванні опитувань виникає допоміжна задача: відібрати якомога менше респондентів, щоб вони утворювали репрезентативну вибірку, тобто представляли весь спектр громадської думки.

Один із способів це зробити полягає в наступному.

Спочатку складаються ознакові описи досить великого числа точок опитування (це можуть бути міста, райони, магазини, і т. д.).

Для цього використовуються недорогі способи збору інформації – пробні опитування або фіксація деяких характеристик самих точок.

Потім вирішується завдання кластеризації, і з кожного кластера відбирається по одній представницькій точці.

Тільки на відібраній множині точок виробляється основне, найбільш ресурсномістке, опитування. Задачі кластеризації, в яких частина об'єктів (як правило, незначна) розмічена по класах, називаються задачами з частковим навчанням (semi-supervised learning). Вважається, що вони не зводяться безпосередньо до класифікації або кластеризації, і для їх вирішення потрібні особливі методи.

Задачі пошуку асоціацій

Задачі пошуку асоціативних правил (association rule induction) винесені в окремий клас і відносяться до задач навчання без учителя, хоча мають багато спільного з задачами класифікації.

Задача аналізу ринкових кошиків

Задача аналізу ринкових кошиків (market basket analysis) полягає у тому, щоб за даними про покупки товарів в супермаркеті (буквально, за чеками) визначити, які товари часто спільно купуються. Ця інформація може бути корисною для:

- оптимізації розміщення товарів на полицях,
- планування рекламних кампаній (промо-акцій),
- управління асортиментом і цінами.

Завдання полягає в тому, щоб виявити всі набори товарів, які часто купують разом. Наприклад, «якщо куплено хліб, то з ймовірністю 60% буде куплено і молоко». До багатьох підручників з бізнес-аналітики увійшов приклад, коли система пошуку асоціативних правил виявила неочевидну закономірність: ввечері перед вихідними днями зростають спільні продажі памперсів і пива.

Розмістивши дорогі сорти пива поруч з памперсами, менеджери змогли збільшити продажі в масштабах всієї роздрібної мережі, що окупило впровадження системи аналізу даних. Пізніше маркетологи та соціологи запропонували розумне пояснення цьому явищу, однак виявлено воно було саме шляхом аналізу даних.

Задача виділення термінів

Задача виділення термінів (term extraction) з текстів, яке вирішується перед задачею рубрикації, може бути зведена до пошуку асоціацій. Термінами вважаються окремі слова або стійкі словосполучення, які часто зустрічаються в невеликій підмножині документів, і рідко – у всіх інших. Множина термінів, які часто зустрічаються спільно, утворює тему, найімовірніше, відповідну до деякої рубрики.

2.2. Сучасна реалізація символьного навчання

Theano – це бібліотека Python, яка дозволяє ефективно визначати, оптимізувати та оцінювати математичні вирази з використанням багатовимірних масивів.

Theano включає:

- мову, яка представляє математичні вирази та керує ними,
- компілятр для створення функцій, здатних обчислювати значення для цих виразів,
- бібліотеку, яка виконуватиме ці функції чисельними методами.

Особливості Theano:

1. Тісна інтеграція з NumPy

(Використовує numpy.ndarray у скомпільованих Theano функціях)

2. Прозоре використання графічного процесора GPU

(Виконує обчислення, що вимагають даних, набагато швидше, ніж на ЦП)

3. Ефективне символьне диференціювання

(Theano робить похідні для функцій з одним або багатьма входами)

4. Оптимізація швидкості та стабільності

(Отримує правильну відповідь для log(1+x), навіть при малому x)

5. Динамічне створення C коду

6. Unit-тестування та самоперевірка.

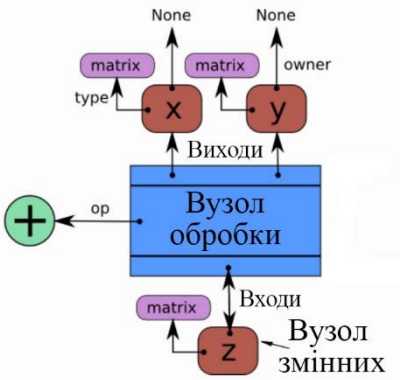
2.3. Мова опису математичних виразів

Структура у вигляді графа

Theano представляє символьні математичні вирази у вигляді орієнтованих ациклічних графів. Ці графи є дводольними, оскільки містять два види вузлів:

• Вузли **змінних** (або змінні), які представляють *дані*, як правило, тензори;

• Вузли **обробки**, які представляють виконання математичних *операцій*.



Призначення вузлів змінних та обробки

Вузли змінних призначені для:

- задавання вхідних даних;
- збереження вихідних даних графа;
- запам’ятовування проміжних даних.

Вузол обробки має входи і виходи, які є вузлами змінних.

Вузол реалізує математичну операцію («Op») над його вхідними змінними.

Кожний вузол змінних може бути, в свою чергу, вхідним для декількох вузлів обробки, але може бути вихідним тільки для одного вузла обробки.

2.4. Типи змінних

Основними категоріями типів є:

• *TensorType*, який представляє n-вимірні масиви в основній пам’яті, значення, пов’язані з змінами цього типу є об’єктами NumPy: *ndarray*;

• *CudaNdarrayType*, який представляє n-вимірні масиви в пам’яті GPU, що пов’язані з об’єктами *CudaNdarray*, які використовуваними застарілому GPU;

• *GpuArrayType*, пов’язаний з об’єктами *GpuArray*, його еквівалент у новому інтерфейсі GPU;

• Sparse, для матриць, що розміщуються в основній пам’яті, представлені матрицями SciPy, CSC або CSR.

Кількісні розміри і тип даних (float32, int64 і т. д.) та форма є частиною типу.

2.5. Побудова графа

1. Створюємо типізовані вільні символьні змінні, які відповідають входам графа.
2. Створюємо операції які відповідають заданим типам змінних.

Модуль *tensor:* Містить функції для тензорних операцій (на базі NumPy)
Результати обробки

- передаються на вихідний вузол змінних
- передаються на групу вихідних вузлів

3. Клонування(Дозволяє програмувати цикли)

- клонування всього графа.
- клонування частини графа

2.6. Символьне диференціювання

Обчислення базується в застосуванні правила ланцюга. Ця процедура називається градієнтним зворотним поширенням, або зворотним режимом диференціювання.

Приклад.

Розглянемо три функції $f : R^M \to R$, $g : R^N \to R^M$ і $C : R^N \to R$ такі, що $C\left(x\right) = f\left(g\left(x\right)\right)$, то:

$$\frac{\partial C}{\partial x}\Big|_x = \frac{\partial f}{\partial g}\Big|_{g(x)} \cdot \frac{\partial g}{\partial x}\Big|_x$$

Замість того, щоб обчислювати (і зберігати в пам'яті) повністю цілу $M \times N$ матрицю якобіана, $\frac{\partial g}{\partial x}\Big|_x$, все, що нам потрібно – це функція $\nabla g_x : R^N \to R^M$, $v \mapsto v \cdot \frac{\partial g}{\partial x}\Big|_x$, яка обчислює вектор-якобіан точковий добуток для будь-якого вектора v.

2.7. Фаза компіляції

Створюється *функція* Theano (об'єкт, що викликається Python), який може обчислити значення для заданих вихідних символьних змінних, шляхом задавання значення для вхідних змінних.

Сукупність вхідних та вихідних змінних повинна бути надана при компіляції функції, але входи не повинні бути входами повного графа обчислень, і виходи також не повинні бути кінцевими виходами.

Можна скомпілювати функцію, що йде від деяких проміжних змінних графа до інших проміжних змінних, до тих пір, поки набір входів буде містити всю інформацію, необхідну для обчислення множини виходів.

Кілька функцій Theano можуть бути скомпільовані, для того, щоб обчислювати різні частин одного і того ж обчислювального графа.

2.8. Оптимізація графів

Структура обчислювального графа дозволяє замінити частини графа. Наприклад, вузол змінних, який є виходом одного конкретного вузла обробки може бути замінено виходом іншого вузла обробки, якщо вони мають однакові типи Тип Оптимізація вказує, як виконувати заміщення змінних іншими змінами, що представляють еквівалентні обчислення. Деякі з них *локальні*, а це означає, що вони лише дивляться на один вузол обробки і можуть замінити його виходи, деякі з них глобальні, і можуть розглядати весь граф обчислень і виконувати довільні заміни. Оптимізація в основному організована в описаних нижче стадіях, в яких існують певні збіги.

• *Нормалізація:* Перетворити графік у канонічну форму, щоб полегшити завдання наступних оптимізацій (наприклад, $x * x \Rightarrow x^2$). Це дає деякі спрощення, наприклад видалення дублікатів обчислень, видалення деяких

непотрібних обчислень ($\frac{xy}{y} \Rightarrow x$) та обчислення значення виразів, якщо всі їхні вхідні дані відомі (згортання констант, $2+2 \Rightarrow 4$).

• *Стабілізація:* збільшення числової стійкості, наприклад $\log\left(1+x\right) \Rightarrow \log 1p\left(x\right)$, де $\log 1p$ є стабільною реалізацією для випадку малих x.

• *Спеціалізація:* Включити більш швидкі реалізації операцій. Наприклад, послідовні поелементні операції зливаються разом щоб уникнути необхідності створення циклу над тензором декілька разів.

• *GPU:* замінити версію за замовчуванням для Op та змінних на специфічну версію CPU, використовуючи старішу або більш нову за потребою. Якщо запитується графічний процесор.

• *Безпосереднє виконання:* Замінити версію Op за замовчуванням на версією, яка може працювати in-place, як перегляд або оцінка деструктивності на його входах. Типи масивів, що використовуються Theano, як, наприклад, ndarray, підтримують довільні масиви, тому всі операції переносу, а також базове нарізання (basic slicing), може відбутися на місці, в реальному часі. Деякі операції, як і більшість поелементних, можуть перезаписати і повернути його, щоб уникнути захоплення пам'яті. Оскільки деструктивні операції вводять додаткові залежності між вузлами обробки (значення може бути перезаписаним лише останньою операцією, щоб прочитати його), цикли залежності повинні бути виявлені та видалені.

• *Сканування:* оптимізувати продуктивність та використання пам'яті вузлів сканування. Наприклад, зберегти значення лише для останнього кроку виведення в пам'ять, якщо вся послідовність не потрібна, об'єднати різні вузли сканування для виконання обчислень лише один раз і перемістити інваріанти.

Запитання до розділу 2

1. В чому полягає суть задачі класифікації? Наведіть приклад такої задачі.
2. Які особливості задач прогнозування споживчого попиту? Наведіть приклад залежності втрат від помилка прогнозу.
3. Коли застосовують задачі кластеризації і яка мета досягається при вирішенні цих задач?
4. Опишіть порядок вирішення задачі соціологічного опитування і опишіть переваги, які дає вирішення даної задачі.
5. Що таке символьне навчання і яке програмне забезпечення дозволяє його реалізувати?

Розділ 3
Базові поняття прикладних методів аналізу даних

Нехай задано множину об'єктів X, множину допустимих відповідей Y, і існує цільова функція $y^* : X \to Y$, значення якої $y_i = y^*\left(x_i\right)$ відомі лише на скінченній підмножині об'єктів $\left\{x_1, x_2, ..., x_m\right\} \subset X$. Упорядковані двійки $\left(x_i, y_i\right)$ називатимемо випробуваннями. Всю сукупність пар

$$X_m = \left\{\left(x_1, y_1\right), \left(x_2, y_2\right), ...\left(x_m, y_m\right)\right\}$$

будемо називати навчальною вибіркою (training sample)

Навчання полягає в тому, щоб шляхом використання вибірки X_m відновити залежність y^*, тобто побудувати таку функцію прийняття рішень (decision function) $a : X \to Y$, яка б наближено відповідала цільовій функції y^* не тільки на об'єктах навчальної вибірки, але і на всій множині X.

Функція прийняття рішень a повинна допускати ефективну комп'ютерну реалізацію, тобто повинна бути представлена у вигляді певного алгоритму. Тому для спрощення можемо називати її просто алгоритмом.

3.1. Об'єкти та ознаки

Ознака (*feature*) об'єкта x – це результат вимірювання деяких характерстики об'єкта. Формально ознакою об'єкта будемо називати відображення:

$f : X \to D_f$,

де D_f – множина допустимих значень ознаки.

В залежності від природи множини D_f ознаки діляться на кілька типів.

Якщо $D_f = \left\{0,1\right\}$, то функція f є бінарною ознакою;

Якщо D_f – скінченна множина, то f – номінальна ознака;

Якщо D_f – скінченна упорядкована множина, то f – упорядкована ознака;

Якщо $D_f \subseteq R$, то f – кількісна ознака.

Якщо всі ознаки мають однаковий тип, тобто:

$type\left(D_{f_1}\right) = type\left(D_{f_2}\right) = ... = type\left(D_{f_n}\right)$,

то початкові дані називають однорідними.

Якщо існує хоча б один набір ознак іншого типу, то говорять про різнорідний набір даних.

Вектор ознак $\left(f_1\left(x\right), f_2\left(x\right),....f_n\left(x\right)\right)$ будемо називати описом об'єкта $x \in X$ за його ознаками.

В подальшому кожен об'єкт x розглядатиметься через сукупність його ознак, множина яких є підмножиною декартового добутку

$D_{f_1} \times D_{f_2} \times ... \times D_{f_2}$.

Тому сукупність всіх описів об'єктів вибірки X_m за їх ознаками може бути представлена у вигляді $m \times n$ матриці:

$$F = \left\| f_i\left(x_i\right) \right\|_{m \times n} = \begin{pmatrix} f_1\left(x_1\right) & ... & f_n\left(x_1\right) \\ ... & ... & ... \\ f_1\left(x_m\right) & ... & f_n\left(x_m\right) \end{pmatrix}$$

Цю матрицю називають *матрицею об'єктів* ознак і вона є найбільш поширеним способом представлення початкових даних у прикладних задачах.

3.2. Типи задач

В задачах статистичного навчання відповіді Y можуть характеризуватися різною природою

В залежності від цієї природи розглянемо такі типи задач навчання:

Задача класифікації (classification)

В цій задачі відповіді поділяються на M класів, які не перетинаються: $Y = \left\{1, 2, ..., M\right\}$

У цьому випадку всі об'єкти X розбиваються на класи:

$$K = \left\{ x \in X \middle| y^*\left(x\right) = y\left(x\right) \right\},$$

а основне завдання алгоритма a полягає у тому, щоб максимально точно визначити, до якого з класів відноситься даний об'єкт x

Якщо $Y \subseteq R$, то це задача відновлення регресії (оцінка регресії).

Задача регресійної оцінки ((regression estimation)

Для даної задачі множина виходів $Y \subseteq R$. Алгоритм a повинен максимально наближене значення y для кожного об'єкта x, який не належить навчальній виборці.

Задача прогнозування (forecasting)

У загальному випадку задача прогнозування є окремим випадком задачі класифікації. Або задачі регресійної оцінки. Тобто при виконанні задачі прогнозування необхідно за попередньою оцінкою поведінки об'єкта $x \in X$ необхідно визначити його подальшу поведінку.

3.3. Модель алгоритмів та метод навчання

Визначення.

Моделлю алгоритмів називається параметричне сімейство відображень:

$$A = \left\{ g\left(x, \theta\right) \middle| \theta \in \Theta \right\},$$

де $g : X \times \Theta \to Y$ - деяка фіксована функція, Θ - множина допустимих значень параметра θ, яку називають простором параметрів або простором пошуку (*search space*).

Приклад моделі 1. Задачі з n числовими ознаками.

В задачах з n числовими ознаками $f_j : X \to R,\ j = 1,2,\ldots,n$ широко використовуються лінійні моделі з вектором параметрів

$\theta = \left(\theta_1, \theta_2, \ldots, \theta_n\right),\ \theta \in \Theta,\ \Theta \subseteq R^n.$

Для задач класифікації за умови $Y = \left\{-1,1\right\}:\ g\left(x,\theta\right) = sign\sum_{j=1}^{n} \theta_j f_j\left(x\right)$

Для задач регресійної оцінки за умови $Y \subseteq R:\ g\left(x,\theta\right) = \sum_{j=1}^{n} \theta_j f_j\left(x\right)$

Ознаками можуть бути не тільки початкові вимірювання, але й функції від них.

Зокрема, багатовимірні лінійні моделі можуть використовуватися навіть у задачах з єдиною числовою ознакою.

Приклад моделі 2. Поліноміальні моделі.

Один з класичних підходів до апроксимації функцій з однією змінною за заданими точками

$$\left(x_i, y_i\right) \in R^2,\ i = 1,2,\ldots,m$$

Полягає в побудові поліноміальної моделі. Якщо ввести n ознак $f_j\left(x\right) = x^{j-1}$, то функція $g\left(x,\theta\right)$ з прикладу 1 буде визначати поліном степеня $n-1$ над початковою ознакою x.

Процес підбору оптимального параметра моделі θ по навчальній вибірці X_m називають налаштуванням навчанням(*training, learning*) алгоритму $a \in A$.

Визначення. Метод навчання (алгоритм навчання) – це відображення

$$\mu : \left(X \times Y\right)_m \to A,$$

яке довільній скінченній вибірці $X_m = \left(x_i, y_i\right)_{i=1}^{m}$ ставить в відповідність деякий алгоритм $a \in A$. Говорять також, що метод μ будує алгоритм по вибірці X_m. Метод навчання повинен мати ефективну програмну реалізацію. Отже, у задачах статистичного навчання існують два етапи.

1. Етап навчання. На цьому етапі метод μ по вибірці X_m будує алгоритм $a = \mu\left(X_m\right)$

2. Етап застосування. На етапі застосування алгоритм a для нових об’єктів x видає відповіді $y = a\left(x\right)$.

Основним є етап навчання. Як правило, він зводиться до пошуку параметрів моделі, які забезпечують оптимальне значення заданому функціоналу якості.

3.4. Функціонал якості

Визначення. Функція втрат (*loss function*) – це невід'ємна функція $\lambda\left(a,x\right)$, яка характеризує величину помилки алгоритму a на об'єкті x. Якщо $\lambda\left(a,x\right)=0$, то відповідь $a\left(x\right)$ називають коректною.

Визначення. Функціонал якості алгоритму a на вибірці X_m визначається з виразу:

$$Q\left(a,X_m\right)=\frac{1}{m}\sum_{i=1}^{m}\lambda\left(a,x_i\right)$$

Функціонал Q також називають *функціоналом середніх втрат або емпіричним ризиком*, оскільки він його розраховують за емпіричними даними

$$\left(x_i,y_i\right)_{i=1}^{m}$$

Якщо функція втрат приймає тільки значення 0 і 1, то її називають *бінарною*. У цьому випадку $\lambda\left(a,x\right)=1$ означає, що алгоритм допускає помилку на об'єкті x, а функціонал Q називається частотою помилок алгоритму на вибірці X_m.

Найчастіше використовують такі функції втрат, при $Y\subseteq R$:

1. Індикатор помилки, який застосовують в задачах класифікації:

$$\lambda\left(a,x\right)=\left|a\left(x\right)\neq y^*\left(x\right)\right|$$

2. Відхилення від правильної відповіді:

$$\lambda\left(a,x\right)=\left|a\left(x\right)-y^*\left(x\right)\right|$$

В цьому випадку функціонал $Q\left(a,X_m\right)=\frac{1}{m}\sum_{i=1}^{m}\left|a\left(x_i\right)-y^*\left(x_i\right)\right|$ називають середньою помилкою алгоритму a на вибірці X_m.

3. Квадратична функція втрат:

$$\lambda\left(a,x\right)=\left(a\left(x\right)-y^*\left(x\right)\right)^2.$$

В цьому випадку функціонал $Q\left(a,X_m\right)=\frac{1}{m}\sum_{i=1}^{m}\left(a\left(x_i\right)-y^*\left(x_i\right)\right)^2$ називають середньо квадратичною помилкою алгоритму a на вибірці X_m. Зазвичай її застосовують в задачах регресії.

Класичний метод навчання, який називають мінімізацією емпіричного ризику (*empirical risk minimization, ERM*), полягає у тому, щоб знайти в заданій моделі A алгоритм a, що забезпечує мінімальне значення функціоналу якості Q на заданій навчальній вибірці X_m:

$$\mu\left(X_m\right) = \arg\min_{a \in A} Q\left(a, X_m\right)$$

Приклад моделі 3.

В задачі регресійної оцінки ($Y \subseteq R$) з числовими ознаками $f_j : X \to R,\ j = 1,2,\ldots,n$ і квадратичною функцією втрат метод мінімізації емпіричного ризику є ніщо інше, як метод найменших квадратів:

$$\mu\left(X_m\right) = \arg\min_{\theta} \sum_{i=1}^{m} \left(g\left(x_i, \theta\right) - y_i\right)^2 .$$

3.5. Проблема перенавчання

Мінімізацію емпіричного ризику слід застосовувати з відомою долею обережності. Якщо мінімум функціонал $Q\left(a, X_m\right)$ досягається на алгоритмі a, то це ще не гарантує того, що буде добре наближувати цільову функцію на довільній контрольній вибірці $X_k = \left(x_i, y_i\right)_{i=1}^{k}$.

У випадку, якщо якість роботи алгоритму на нових об'єктах, які не входять в склад навчання, виявляється істотно гіршою, ніж у навчальній вибірці, то говорять про ефект перенавчання (*overfitting*). При вирішенні практичних задач з цим явищем приходиться зустрічатися дуже часто. Легко представити собі метод, який мінімізує емпіричний ризик до нуля, але при цьому абсолютно нездатний навчатись. Після одержання навчальної вибірки X_m, він запам'ятовує її і будує алгоритм, який порівнює новий об'єкт x з навчальними об'єктами x_i з X_m. Якщо $x = x_i$, то алгоритм видає правильну відповідь y_i. Якщо $x \neq x_i$, то в цьому випадку виникає ризик отримати будь-яку відповідь. Емпіричний ризик приймає найменше можливе значення, яке дорівнює нулю. Але такий алгоритм не може відновити залежність в точках, які не співпадають з точками навчання.

Висновок: *Для успішного навчання треба не тільки запам'ятовувати, але й потрібно узагальнювати.*

Здатність до узагальнення (*generalization ability*) методу μ характеризується величиною $Q\left(\mu\left(X_m\right), X_k\right)$ за умови, що вибірки X_m і X_k є представницькими. Для формалізації поняття «представницька вибірка» зазвичай приймають стандартне припущення, що вибірки X_m і X_k – прості, отримані з одного і того ж невідомого імовірнісного розподілу на множині X.

Визначення. Метод навчання μ називають спроможним, якщо при заданих достатньо малих значеннях ε і η справедлива нерівність:

$$P_{X_m, X_k}\left\{Q\left(\mu\left(X_m\right), X_k\right) > \varepsilon\right\} < \eta$$

Параметр ε називають точністю, а параметр $\left(1-\eta\right)$ – надійністю.

Допустиме також еквівалентне формулювання: для будь-яких простих вибірок X_m і X_k оцінка $Q\left(\mu\left(X_m\right), X_k\right) \le \varepsilon$ справедлива з ймовірністю не менше $\left(1-\eta\right)$.

Емпіричні оцінки здатності до узагальнення застосовуються в тих випадках, коли не вдається скористатися теоретичними оцінками.

Нехай дана вибірка $X_M = \left(x_i, y_i\right)_{i=1}^{M}$.

Розіб'ємо її N різними способами на дві підвибірки, що не перетинаються:

- навчальну підвибірку довжиною m: X_m
- контрольну підвибірку довжиною $k = M - m$: X_k

Для кожного розбиття $n = 1, 2, \ldots, N$ побудуємо алгоритм $a_n = \mu\left(X_{m_n}\right)$ і обчислимо значення $Q_n = Q\left(a_n, X_{k_n}\right)$.

Середнє арифметичне значень Q_n по всім розбиттям називається перехресною перевіркою (*cross-validation*, CV).

$$CV\left(\mu, X_M\right) = \frac{1}{N}\sum_{n=1}^{N} Q\left(\mu\left(X_{m_n}\right), X_{k_n}\right)$$

Можливо різні варіанти перехресних перевірок, які відрізняються способами розбиття вибірки X_M. В найпростішому варіанті розбиття генеруються випадковим чином вигляді. Число N береться в діапазоні від 20 до 100. На практиці стандартом вважають методику $t \times q$ – кратного перехресного контролю ($t \times q$ *–fold cross-validation*). У цьому випадку вибірка випадковим чином розбивається на q блоків однакової (або майже однакової) довжини, кожний блок по черзі стає контрольною вибіркою, а об'єднання решти блоків – навчальною вибіркою. Вибірка X_L по-різному t разів розбивається на q блоків В результаті отримуємо $N = tq$ розбиттів. Дана методика дає точніші оцінки за рахунок того, що всі об'єкти точно по t разів зустрічаються в контролі. Недоліками перехресних перевірок є: обчислювальна неефективність, висока дисперсія, неповне використання наявних даних для навчання через скорочення довжини навчальної вибірки з M до m.

3.6. Прикладні задачі навчання

Задачі класифікації (classification)

Медична діагностика

У задачах медичної діагностики в ролі об’єктів виступають пацієнти. Ознаки характеризують результати обстежень, симптоми захворювання і методи лікування, які застосовувалися.

Приклади бінарних ознак:

- стать,
- наявність головного болю,
- слабкості,
- нудоти, і т. д.

Приклади порядкових ознак:

- тяжкість стану (задовільний, середньої тяжкості, важкий, вкрай важке).

Приклади кількісних ознаки:

- вік,
- пульс,
- артеріальний тиск,
- вміст гемоглобіну в крові,
- доза препарату, і т. д.

Такий ознаковий опис пацієнта є, по суті справи, формалізованою історією хвороби. Накопичивши достатню кількість випадків, можна вирішувати різні завдання:

- класифікувати вид захворювання (диференціальна діагностика);
- визначати найбільш доцільний спосіб лікування;
- прогнозувати тривалість і результат захворювання;
- оцінювати ризик ускладнень;
- знаходити синдроми – найбільш характерні для даного захворювання сукупності симптомів.

Цінність такого роду систем в тому, що вони здатні миттєво аналізувати і узагальнювати величезну кількість випадків, що недоступно людині.

Задача оцінювання 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 розвинених країнах. Об’єктами в даному випадку є позичальники – фізичні або юридичні особи, які претендують на отримання кредиту. У випадку фізичних осіб опис ознак складається з анкети, яку заповнює сам позичальник, і, можливо, додаткова інформація, яку банк збирає про нього з власних джерел.

Приклади бінарних ознак:

- стать,
- наявність телефону.

Приклади номінальних ознак:

- місце проживання,
- професія,
- роботодавець.

Приклади порядкових ознак:

- освіта,
- посада на роботі.
- Кількісні ознаки:
- вік,
- стаж роботи,
- дохід сім'ї,
- розмір заборгованостей в інших банках,
- сума кредиту.

Навчальна вибірка складається з позичальників з відомої кредитною історією. У найпростішому випадку прийняття рішень зводиться до класифікації позичальників на два класи: «хороших» і «поганих». Кредити видаються тільки позичальникам першого класу. У більш складному випадку оцінюється сумарне число балів (*score*) позичальника, набраних за сукупністю інформативних ознак. Чим вище оцінка, тим надійнішим вважається позичальник. Звідси і назва: кредитний скорінг (*credit scoring*). На стадії навчання проводиться синтез і відбір інформативних ознак і визначається, скільки балів призначати за кожну ознаку, щоб ризик прийнятих рішень був мінімальний.

Наступне завдання: вирішити, на яких умовах видавати кредит:

- визначити процентну ставку,
- термін погашення,
- інші параметри кредитного договору.

Це завдання також зводиться до статистичного навчання.

Задачі регресійної оцінки (regression estimation)

Задача аналізу зросту

Термін «регресія» був введений в 1886 році антропологом Френсісом Гальтоном при вивченні статистичних закономірностей спадковості зросту. Повсякденний досвід підказує, що в середньому зріст дорослих дітей тим більше, чим вище їх батьки. Але Гальтон виявив, що сини дуже високих батьків часто мають не такий високий зріст. Він зібрав вибірку даних по 928 парах батько-син. Кількісно залежність непогано описувалась лінійною функцією $y = \frac{2}{3}x$, де $x-$ відхилення зросту батька від середнього, а $y-$ відхилення зросту сина від середнього. Гальтон назвав це явище «регресією до

пересічності», тобто, до середнього значення в популяції. Термін «регресія», або рух назад натякав також на нестандартний для того часу хід дослідження: спочатку були зібрані дані, потім по них вгадана модель залежності, тоді як традиційно поступали навпаки: дані використовувалися лише для перевірки теоретичних моделей. Це був один з перших випадків моделювання, заснованого виключно на даних. Пізніше термін, що виник в конкретній прикладній задачі, закріпився за широким класом методів відновлення залежностей. Величезна кількість регресійних задач виникає в фізичних експериментах, в промисловому виробництві, в економіці.

Задача прогнозування споживчого попиту

Ця задача вирішується сучасними супермаркетами і торговими роздрібними мережами. Для ефективного управління торговою мережею необхідно прогнозувати обсяги продаж для кожного товару на задане число днів наперед. На основі цих прогнозів здійснюється:

- планування закупівель,
- управління асортиментом,
- формування цінової політики,
- планування промоакцій (рекламних кампаній).

Специфіка задачі полягає в тому, що кількість товарів може обчислюватися десятками або навіть сотнями тисяч. Прогноз і прийняття рішень по кожному товару «вручну» просто немислимо. Початковими даними для прогнозування є часові ряди цін і обсягів продаж по товарах і по окремих магазинах. Сучасні технології дозволяють отримувати ці дані від касових апаратів і накопичувати в єдиному сховищі даних. Для збільшення точності прогнозів необхідно враховувати різні зовнішні фактори, що впливають на попит:

- рекламні кампанії,
- соціально-демографічні умови,
- активність конкурентів,
- свята,
- погодні умови.

В залежності від цілей аналізу в ролі об’єктів виступають або товари, або магазини, або пари «магазин-товар». Ще одна особливість задачі - несиметричність функції втрат. Якщо прогноз робиться з метою планування закупівель, то втрати від заниженого прогнозу, як правило, істотно вище, ніж від завищеного.

Задачі ранжування (ranking)

Задачі ранжування виникають в області інформаційного пошуку. Результатом пошуку за запитом може виявитися настільки довгий список відповідей, що користувач фізично не зможе його переглянути. Тому відповіді упорядковують за спаданням релевантності (ступеня їх відповідності до запиту). Критерій упорядкування в явному вигляді невідомий, хоча людина

легко відрізняє більш релевантні відповіді від менш релевантних або зовсім недоречні. Зазвичай для розмітки вибірки пар «запит-відповідь» залучають команду експертів (асесорів), щоб врахувати думки різних людей, які часто суперечать один з одним. Потім вирішують завдання навчання ранжирування (learning to rank).

Задача ранжування текстових документів

Задача ранжування текстових документів, які знайдені знайдених в Інтернеті за запитом користувача, вирішується всіма сучасними пошуковими машинами. Об'єктами є пари «запит-документ», а відповіді – це оцінки релевантності, зроблені асессорами.

Залежно від методології формування навчальної вибірки оцінки асессорів можуть бути бінарними (релевантний, або не релевантний) або порядковими (релевантність в балах). Ознаками є числові характеристики, які обчислюють для пари «запит-документ». Текстові ознаки засновані на підрахунку числа входжень слів запиту в документи. Можливі багаточисельні варіанти: з урахуванням синонімів або без них, з урахуванням числа входжень або без нього, в усьому документі або тільки в заголовках, і т. д. Посилальні ознаки засновані на підрахунку числа документів, що посилаються на даний. Клікові ознаки засновані на підрахунку числа звернень до даного документу.

Задачі кластеризації

Задача кластеризації (clustering) відрізняються від класифікації (classification) тим, що в них не задаються відповіді $y_i = y^*\left(x_i\right)$. Відомі тільки самі об'єкти x_i, і необхідно розбити вибірку на підмножини (кластери) так, щоб кожен кластер складався з схожих об'єктів, а об'єкти різних кластерів істотно відрізнялися. Для цього необхідно задати функцію відстані на множині об'єктів. Число кластерів також може здаватися, але частіше потрібно визначити і його.

Задача соціологічного опитування

Основним інструментом соціологічних і маркетингових досліджень є проведення опитувань. Щоб результати опитування були об'єктивні, необхідно забезпечити представництво вибірки респондентів. З іншого боку, потрібно мінімізувати вартість проведення опитування. Тому при плануванні опитувань виникає допоміжна задача: відібрати якомога менше респондентів, щоб вони утворювали репрезентативну вибірку, тобто представляли весь спектр громадської думки. Один із способів це зробити полягає в наступному. Спочатку складаються ознакові описи досить великого числа точок опитування (це можуть бути міста, райони, магазини, і т. д.). Для цього використовуються недорогі способи збору інформації – пробні опитування або фіксація деяких характеристик самих точок. Потім вирішується завдання кластеризації, і з кожного кластера відбирається по одній представницькій точці. Тільки на відібраній множині точок виробляється основне, найбільш

ресурсномістке, опитування. Задачі кластеризації, в яких частина об'єктів (як правило, незначна) розмічена по класах, називаються задачами з частковим навчанням (semi-supervised learning). Вважається, що вони не зводяться безпосередньо до класифікації або кластеризації, і для їх вирішення потрібні особливі методи.

Задача рубрикації

Задача рубрикації текстів виникає при роботі з великими колекціями текстових документів. Припустимо, є деякий ієрархічний рубрикатор, розроблений експертами для даної предметної області (наприклад, для спортивних новин), або для всіх областей (наприклад, універсальний десятковий класифікатор УДК). Є множина документів, класифікованих за рубриками вручну. Потрібно класифікувати за тими ж рубриках другої множини документів, яке може бути істотно більше першої. Для вирішення даної задачі використовується функція відстані, що порівнює тексти за складом термінів. Термінами, як правило, є спеціальні поняття предметної області, власні імена, географічні назви, і т. д. Документи вважаються схожими, якщо множини їх термінів суттєво перетинаються.

Задачі пошуку асоціацій

Задачі пошуку асоціативних правил (association rule induction) винесені в окремий клас і відносяться до задач навчання без учителя, хоча мають багато спільного з задачами класифікації.

Задача аналізу ринкових кошиків

Задача аналізу ринкових кошиків (market basket analysis) полягає у тому, щоб за даними про покупки товарів в супермаркеті (буквально, за чеками) визначити, які товари часто спільно купуються. Ця інформація може бути корисною для:

- оптимізації розміщення товарів на полицях,
- планування рекламних кампаній (промо-акцій),
- управління асортиментом і цінами.

У цій задачі об'єкти відповідають чеками, а ознаки є бінарними і відповідають товарам. Одиничне арифметичне значення ознаки $f_j\left(x_i\right)=1$ означає, що в $i-$му чеку зафіксована покупка $j-$ го товару. Завдання полягає в тому, щоб виявити всі набори товарів, які часто купують разом. Наприклад, «якщо куплено хліб, то з ймовірністю 60% буде куплено і молоко». До багатьох підручників з бізнес-аналітики увійшов приклад, коли система пошуку асоціативних правил виявила неочевидну закономірність: ввечері перед вихідними днями зростають спільні продажі памперсів і пива.

Розмістивши дорогі сорти пива поруч з памперсами, менеджери змогли збільшити продажі в масштабах всієї роздрібної мережі, що окупило впровадження системи аналізу даних. Пізніше маркетологи та соціологи

запропонували розумне пояснення цьому явищу, однак виявлено воно було саме шляхом аналізу даних.

Задача виділення термінів

Задача виділення термінів (term extraction) з текстів, яке вирішується перед задачею рубрикації, може бути зведена до пошуку асоціацій. Термінами вважаються окремі слова або стійкі словосполучення, які часто зустрічаються в невеликій підмножині документів, і рідко - у всіх інших. Множина термінів, які часто зустрічаються спільно, утворює тему, найімовірніше, відповідну до деякої рубрики.

3.7. Методика тестування алгоритмів навчання

Поки ще не створений універсальний метод статистичного навчання, здатний вирішуватися будь-які практичні завдання однаково добре. Кожен метод має свої переваги, недоліки і межі застосування. На практиці доводиться проводити чисельні експерименти, щоб зрозуміти, який метод з наявного арсеналу краще підходить для конкретного завдання. Зазвичай для цього методи порівнюються за крос-волідацією. Існує два типи експериментальних досліджень, що відрізняються цілями і методикою проведення.

Експерименти на модельних даних.

Мета експериментів на модельних даних:

- виявлення границь застосовності методу навчання;
- побудова прикладів вдалої і невдалої його роботи;
- розуміння, на що впливають параметри методу навчання.

Модельні експерименти часто використовуються на стадії налагодження методу. Модельні вибірки спочатку генеруються в двовимірному просторі, щоб роботу методу можна було наочно уявити на плоских графіках. Потім досліджується робота методу на багатовимірних даних, при різній кількості ознак. Генерація даних виконується або за допомогою датчика випадкових чисел за заданими імовірнісними розподілами, або детермінованим чином. Часто генерується не одна модельна задача, а ціла серія, параметрезованих таким чином, щоб серед завдань виявилися як завідомо «легкі» так і завідомо «складні»; при такій організації експерименту точніше виявляються межі застосування методу.

Експерименти на реальних даних

Мета експериментів на реальних даних: або вирішення конкретної прикладної задачі, або виявлення «слабких місць» і меж застосування конкретного методу. У першому випадку фіксується задача, і до неї застосовуються багато методів, або, можливо, один і той же метод при різних значеннях параметрів. У другому випадку фіксується метод, і з його допомогою вирішується велика кількість задач (зазвичай кілька десятків). Спеціально для проведення таких експериментів створюються загальнодоступні репозиторії реальних даних. Найбільш відомий репозиторій

UCI (університету Ірвіна, Каліфорнія), доступний за адресою http://archive.ics.uci.edu/ml. Він містить близько двох сотень завдань, в основному класифікації, з самих різних предметних областей.

Полігон алгоритмів класифікації

У наукових статтях по машинному навчанню прийнято наводити результати тестування запропонованого нового методу навчання в порівнянні з іншими методами на представницькому наборі задач. Порівняння повинно проводитися в рівних умовах за однаковою методикою; якщо це крос-волідація, то на одній і тій же множині розбиттів. Незважаючи на значну стандартизацію таких експериментів, результати тестування одних і тих же методів на одних і тих же задачах, отримані різними авторами, все ж можуть істотно відрізнятися. Проблема в тому, що використовуються різні реалізації методів навчання і методик тестування, а проведений кимось раніше експеримент практично неможливо відтворити у всіх деталях. Для вирішення цієї проблеми розроблено Полігон алгоритмів класифікації, доступний за адресою http://poligon.MachineLearning.ru. У цій системі реалізована уніфікована розширена методика тестування та централізоване сховище завдань. Реалізація алгоритмів класифікації, навпаки, децентралізована. Будь-який користувач Інтернету може оголосити свій комп'ютер обчислювальним сервером Полігону, які реалізують один або кілька методів класифікації. Всі результати тестування зберігаються як готові звіти в базі даних системи і можуть бути в будь-який момент видані за запитом без проведення трудомістких обчислень знову.

Конкурси з вирішення завдань аналізу даних

В останні роки компанії, зацікавився у вирішенні прикладних задач аналізу даних, все частіше стали звертатися до такої форми залучення наукового співтовариства, як відкриті конкурси з грошовими преміями для переможця. У кожному такому конкурсі публікується навчальна вибірка з відомими відповідями, тестова вибірка, відповіді на якій відомі тільки організатору конкурсу, і критерій, за яким алгоритми претендента порівнюються на даних тестової вибірки. Інформацію про поточні конкурси можна знайти на сайтах http://www.kaggle.com, http://tunedit.org. Існують також сайти, на яких можна тестувати різні алгоритми на різних наборах даних: http://mlcomp.org.

3.8. Прийоми генерації модельних даних

У ньому перераховані деякі відомості, корисні при генерації модельних вибірок даних.

Моделювання випадкових даних.

Наступні твердження дозволяють генерувати випадкові вибірки з заданими розподілами. Будемо припускати, що є стандартний спосіб отримувати рівномірно розподілені на відрізку [0,1] випадкові величини.

Твердження 1. Якщо випадкова величина r рівномірно розподілена на [0, 1], то випадкова величина $\xi = \lfloor r < p \rfloor$ набуває значення 1 з імовірністю p і значення 0 з ймовірністю $1 - p$.

Твердження 2. Якщо випадкова величина r рівномірно розподілена на [0, 1], і задана зростаюча послідовність $F_0 = 0, F_1, \ldots, F_{k-1}, F_k = 1$, то дискретна випадкова величина ξ, що визначається умовою $F_{\xi-1} \le r < F_\xi$, набуває значення $j = 1, \ldots, k$ з імовірністю $p_j = F_j - F_{j-1}$.

Твердження 3. Якщо випадкова величина r рівномірно розподілена на [0, 1], і задана зростаюча на R функція $F(x)$, $0 \le F(x) \le 1$, то випадкова величина $\xi = F^{-1}(r)$ має неперервну функцію розподілу $F(x)$.

Твердження 4. Якщо r_1, r_2 – дві незалежні випадкові величини, які рівномірно розподілені на [0,1], то перетворення Бокса-Мюллера

$$\xi_1 = \sqrt{-2 \ln r_1} \sin 2\pi r_2$$

$$\xi_2 = \sqrt{-2 \ln r_1} \cos 2\pi r_2$$

дає дві незалежні нормальні випадкові величини з нульовим математичним очікуванням і одиничною дисперсією: $\xi_1, \xi_2 \in N(0,1)$.

Невипадкові модельні дані

Невипадкові модельні дані дозволяють наочно продемонструвати, в яких випадках одні методи працюють краще за інших. Один з класичних прикладів – дві спіралі на Рис. 2.1.

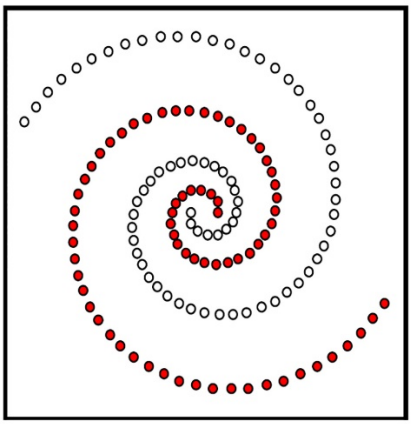


Рис. 3.1. Модельна вибірка «спіралі»

Ця вибірка добре класифікується методом найближчих сусідів, але непереборно важка для лінійних правил розділення. Якщо витки спіралей розташувати ближче один до одного, то задача стане важка і для методу найближчих сусідів. Деякі кусочно-лінійні роздільники справляються із задачею і в цьому випадку. Зазвичай при створенні модельних даних, як випадкових, так і невипадкових, вводиться параметр, плавно змінює задачу від гранично простої до надзвичайно складної. Це дозволяє досліджувати межі

застосування методу. На Рис. 3.2 показана серія модельних задач класифікації з двома класами, що має таку властивість щодо методу найближчих сусідів і деяких інших алгоритмів.

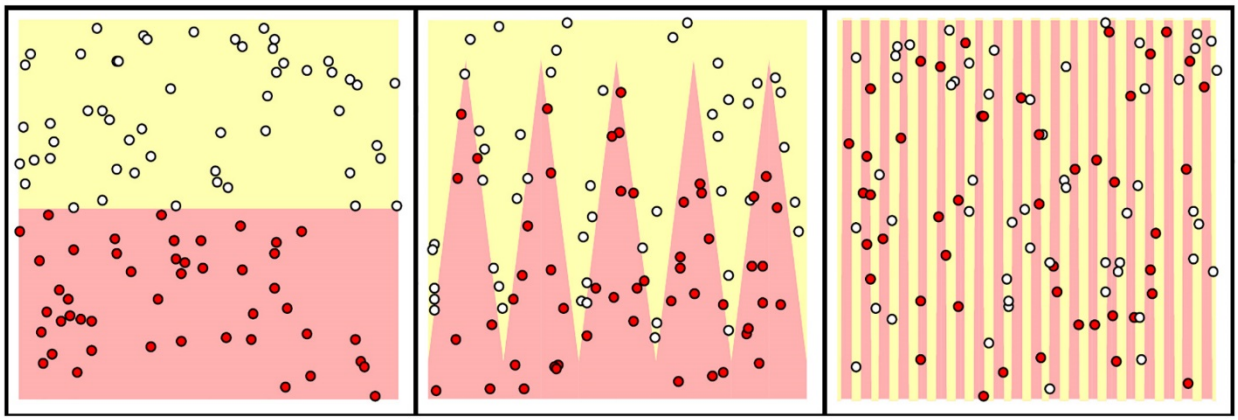


Рис. 3.2. Серія модельних вибірок «пила»

Запитання до розділу 3

1. Яким чином задають об’єкти в задачах машинного навчання?
2. Перерахуйте основні типи задач машинного навчання.
3. Що таке модель алгоритмів і які вам відомі приклади моделей алгоритмів?
4. Дайте визначення та опишіть роль, яку відіграє функціонал якості у задачах машинного навчання?
5. В чому полягає проблема перенавчання та як можна запобігти перенавчанню?

Розділ 4
Метричні методи класифікації

У багатьох прикладних задачах вимірювати ступінь подібності об'єктів суттєво простіше, ніж формувати ознакові описи. Наприклад, такі складні об'єкти, як фотографії осіб, підписи, тимчасові ряди або первинні структури білків більш природно порівнювати безпосередньо з один з одним шляхом деякого «накладення з вирівнюванням», ніж винаходити якісь ознаки й порівнювати ознакові описи. Якщо міра подібності об'єктів введена досить вдало, те, як правило, виявляється, що схожим об'єктам дуже часто відповідають схожі відповіді. У задачах класифікації це означає, що класи утворюють компактно локалізовані підмножини. Це припущення прийнято називати гіпотезою компактності. Для формалізації поняття «подібності» уводиться функція відстані в просторі об'єктів X. Методи навчання, засновані на аналізі подібності об'єктів, будемо називати метричними, навіть якщо функція відстані не задовольняє всім аксіомам метрики (зокрема, аксіомі трикутника). В англомовній літературі вживаються терміни similarity-based learning або distance-based learning.

4.1. Метод найближчого сусіда і його узагальнення

Нехай на множині об'єктів X задана функція відстані $\rho : X \times X \to \left[0, \infty\right)$. Існує цільова залежність $y^* : X \to Y$, значення якої відомі тільки на об'єктах навчальної вибірки $X_m = \left(x_i, y_i\right)_{i=1}^m$, $y_i = y^*\left(x_i\right)$. Множина класів Y скінченна. Потрібно побудувати такий алгоритм класифікації $a : X \to Y$, що апроксимує цільову залежність $y^*\left(x\right)$ на всій множині X.

4.1.1. Узагальнений метричний класифікатор

Для довільного об'єкта $u \in X$ розташуємо елементи навчальної вибірки $x_1, x_2, ..., x_m$ у порядку зростання відстаней до u:

$$\rho\left(u, x^{(1)}\right) \le \rho\left(u, x^{(2)}\right) \le ... \le \rho\left(u, x^m\right),$$

де $x^{(i)}$ – це i – й сусід об'єкта u.Відповідно, відповідь на i – му сусіді об'єкта u визначається з виразу: $y^{(i)} = y^*\left(x^{(i)}\right)$. Таким чином, будь-який об'єкт $u \in X$ породжує свою перенумерацію вибірки.

Визначення 1. Метричний алгоритм класифікації з навчальною вибіркою X_m відносить об'єкт u до того класу $y \in Y$, для якого сумарна вага найближчих навчальних об'єктів $\Gamma_y\left(u, X_m\right)$ максимальна:

$$a\left(u;X_m\right)=\arg\max_{y\in Y}\Gamma_y\left(u,X_m\right);\ \Gamma_y\left(u,X_m\right)=\sum_{i=1}^{m}\left[y_u^{(i)}=y\right]w\left(i,u\right),\qquad(3.1)$$

де вагова функція $w\left(i,u\right)$ оцінює ступінь важливості і-го сусіда для класифікації об'єкта u. Функцію $\Gamma_y\left(u,X_m\right)$ називають оцінкою близькості об'єкта u до класу y

Метричний класифікатор визначений з точністю до вагової функції $w\left(i,u\right)$. Зазвичай вона вибирається невід'ємною та такою, що не зростає по i. Це відповідає гіпотезі компактності, згідно з якою чому ближче об'єкти u та $x^{(i)}$, тим вище шанси, що вони належать одному класу.

Навчальна вибірка X_m відіграє роль параметра алгоритму a. Налаштування зводиться до запам'ятовування вибірки, і, можливо, оптимізації якихось параметрів вагової функції, однак самі об'єкти не зазнають обробки і зберігаються «як є». Алгоритм $a\left(u;X_m\right)$ будує локальну апроксимацію вибірки X_m, причому обчислення відкладаються до моменту, поки не стане відомий об'єкт u. Із цієї причини метричні алгоритми відносять до методів ледачого навчання (lazy learning), на відміну від старанного навчання (eager learning), коли на етапі навчання будується функція, що апроксимує вибірку.

Метричні алгоритми класифікації відночять також до методів міркування по прецедентах (case-based reasoning, CBR). Тут дійсно можна говорити про «міркування», тому що на запитання «чому об'єкт і був віднесений до класу y ?» алгоритм може дати зрозуміле експертам пояснення: «тому, що є схожі з ним прецеденти класу y», і пред'явити список цих прецедентів.

Вибираючи вагову функцію $w\left(i,u\right)$, можна одержувати різні метричні класифікатори, які докладно розглядаються нижче.

4.1.2. Метод *k*-найближчих сусідів

Алгоритм найближчого сусіда (nearest neighbor, NN) відносить об'єкт, що підлягає класифікації, $u\in X_m$ до того класу, якого належить найближчий навчальний об'єкт:

$$w\left(i,u\right)=\left[i=1\right];\quad a\left(u;X_m\right)=y^{(1)}$$

Цей алгоритм є, як видно, найпростішим класифікатором. Навчання NN зводиться до запам'ятовування вибірки X_m. Єдина перевага цього алгоритму – простота реалізації. Недоліків набагато більше:

1. Нестійкість до погрішностей. Якщо серед навчальних об'єктів є *викид* – об'єкт, що перебуває в оточенні об'єктів чужого класу, то не тільки він сам

буде класифікований невірно, але й ті навколишні його об'єкти, для яких він виявиться найближчим.

2. Відсутність параметрів, які можна було б налаштувати по вибірці. Алгоритм повністю залежить від того, наскільки вдало обрана метрика ρ.

3. У результаті – низька якість класифікації.

Алгоритм k найближчих сусідів (k nearest neighbors, kNN). Щоб згладити вплив викидів, будемо відносити об'єкт u до того класу, елементів якого виявиться більше серед до найближчих сусідів x^i, $i = 1,2,...k$:

$$w(i,u) = [i \le k]; \quad a(u; X_m, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y^i = y]$$

При $k = 1$ цей алгоритм збігається з попереднім, отже, нестійкий до шуму. При $k = m$, навпаки, він надмірно стійкий і вироджується в константу.
Таким чином, крайні значення k є небажаними. На практиці оптимальне значення параметра k визначають за критерієм ковзного контролю *з виключенням об'єктів по одному* (*leave-one-out, LOO*). Для кожного об'єкта $x_i \in X_m$ перевіряється, чи правильно він класифікується по своїх k найближчих сусідах.

$$LOO(k, X_m) = \sum_{i=1}^{m} [a(x_i; X_m \setminus \{x_i\}, k) \ne y_i] \to \min_k$$

Якщо об'єкт x_i, що підлягає класифікації, не виключати з навчальної вибірки, те найближчим сусідом x_i завжди буде сам x_i, і мінімальне (нульове) значення функціонала $LOO(k)$ буде досягатися при $k = 1$.

Існує й альтернативний варіант методу *kNN*: у кожному класі вибирається k найближчих до u об'єктів, і об'єкт u відносять до того класу, для якого середня відстань до k найближчих сусідів є мінімальною.

Алгоритм k зважених найближчих сусідів.

Недолік *kNN* у тому, що максимум може досягатися відразу на декількох класах. У задачах із двома класами цього можна уникнути, якщо взяти непарне k. Більш загальна тактика, яка годиться і для випадку багатьох класів – ввести строго спадну послідовність дійсних ваг w_i, які задають внесок i-го сусіда в класифікацію:

$$w(i,u) = [i \le k] w_i; \quad a(u; X_m, k) = \arg\max_{y \in Y} \sum_{i=1}^{k} [y^i = y] w_i$$

Вибір послідовності w_i є евристикою. Якщо застосувати лінійно спадні ваги $w_i = \dfrac{k+1-i}{k}$, то неоднозначності також можуть виникати, хоча й рідше

(наприклад: класів два; перший і четвертий сусід голосують за клас 1, другий і третій – за клас 2; суми голосів збігаються). Неоднозначність усувається остаточно, якщо застосувати нелінійно спадну послідовність, скажемо, геометричну прогресію: $w_i = q^i$, де знаменник прогресії $q \in (0,1)$ є параметром алгоритму. Його можна підбирати за критерієм LOO, аналогічно як і число сусідів k .

Недоліки найпростіших метричних алгоритмів типу kNN .

- Доводиться зберігати навчальну вибірку цілком. Це призводить до неефективної витрати пам'яті й надмірному ускладненню правила розв'язування. При наявності погрішностей (як у вихідних даних, так і в моделі подібності ρ) це може призводити до зниження точності класифікації поблизу границі класів. Має сенс відбирати мінімальну підмножину еталонних об'єктів, дійсно необхідних для класифікації.

- Пошук найближчого сусіда припускає порівняння об'єкта, що підлягає класифікації, з усіма об'єктами вибірки за $O(m)$ операцій. Для задач із більшими вибірками або високою частотою запитів це може виявитися накладно. Проблема вирішується за допомогою ефективних алгоритмів пошуку найближчих сусідів, що вимагають у середньому $O(\ln m)$ операцій.

- У найпростіших випадках метричні алгоритми мають вкрай бідний набір параметрів, що виключає можливість настройки алгоритму за даними.

4.1.3. Метод парзенівського вікна

Ще один спосіб задати ваги сусідам – визначити w_i як функцію від відстані $\rho\left(u, x^{(i)}\right)$, а не від рангу сусіда i. Введемо функцію ядра $K(z)$, яка незростає на $[0,\infty)$. Поклавши $w(i,u) = K\left(\frac{1}{h}\rho\left(u, x^{(i)}\right)\right)$ у загальній формулі (1), одержимо алгоритм

$$a(u; X_m, h) = \arg\max_{y \in Y} \sum_{k=1}^{m} \left[y^{(i)} = y\right] K\left(\frac{\rho\left(u, x^{(i)}\right)}{h}\right) \qquad (3.2)$$

Параметр h називається шириною вікна й відіграє приблизно ту ж роль, що й число сусідів k. «Вікно» – це сферичний окіл об'єкта u радіусом h, при попаданні в яку навчальний об'єкт x_i «голосує» за віднесення об'єкта u до класу y_i. Ми прийшли до цього алгоритму чисто евристичним шляхом, однак він має більш строге обґрунтування в байесовській теорії класифікації, і, фактично, збігається з методом парзенівського вікна.

Параметр h можна задавати апріорі або визначати по ковзному контролю. Залежність $LOO\left(h\right)$, як правило, має характерний мінімум, оскільки занадто вузькі вікна приводять до нестійкої класифікації; а занадто широкі – до виродження алгоритму в константу.

Фіксація ширини вікна h не підходить для тих задач, у яких навчальні об'єкти суттєво нерівномірно розподілені по простору X. В околі одних об'єктів може виявлятися дуже багато сусідів, а в околі інших – жодного. У цих випадках застосовується *вікно змінної ширини*. Візьмемо фінітне ядро – незростаючу функцію $K\left(z\right)$, позитивну на відрізку [0,1], і рівну нулю поза ним. Визначимо h як найбільше число, при якім рівно k найближчих сусідів об'єкта u одержують ненульові ваги: $h\left(u\right)=\rho\left(u,x^{\left(k+1\right)}\right)$. Тоді алгоритм набуває вигляду

$$a\left(u;X_m,k\right)=\arg\max_{y\in Y}\sum_{i=1}^{k}\left[y^{\left(i\right)}-y\right]K\left(\frac{\rho\left(u,x^{\left(i\right)}\right)}{\rho\left(u,x^{\left(k+1\right)}\right)}\right) \tag{3.3}$$

Помітимо, що при фінітному ядрі класифікація об'єкта зводиться до пошуку його сусідів, тоді як при нефінітному ядрі (наприклад, гаусовському) потрібен перебір усієї навчальної вибірки.

4.1.4. Метод потенційних функцій

У методі парзенівського вікна центр радіального ядра $K_h\left(u,x\right)=K\left(\frac{1}{h}\rho\left(u,x\right)\right)$ розміщується в об'єкті u, що підлягає класифікації. В силу симетричності функції відстані $\rho\left(u,x\right)$ можливий і інший, двоїстий, погляд на метричну класифікацію. Припустимо, що ядро розміщується в кожний навчальний об'єкт x_i і «притягує» об'єкт u до класу y_i, якщо він потрапляє в його окіл радіуса h_i

$$a\left(u;X_m\right)=\arg\max_{y\in Y}\sum_{i=1}^{m}\left[y^{\left(i\right)}-y\right]\gamma_i K\left(\frac{\rho\left(u,x_i\right)}{h_i}\right),\ \gamma\geq 0,h_i>0 \tag{3.4}$$

По суті, ця формула відрізняється від (3.3) тільки тим, що тут ширина вікна h_i залежить від навчального об'єкта x_i, а не від об'єкта u, який підлягає класифікації.

Алгоритм 1. Метод потенційних функцій

Вхідні дані:

X_m – навчальна вибірка;

Вихідні дані:

Коефіцієнти $\gamma_i,\ i = 1,2,\ldots,m$ в (4);

Хід роботи алгоритму:

1: Ініціалізація: $\gamma_i = 0;\ i = 1,\ldots,m$;

2: Повторювати:

3: вибрати об'єкт $x_i \in X_m$;

4: Якщо $a\left(x_i\right) \neq y_i$ то

5: $\gamma_i = \gamma_i + 1$;

6: Поки число помилок на вибірці не виявиться досить малим

Ця ідея лежить в основі методу *потенційних функцій* і має пряму фізичну аналогію з електричним потенціалом. При $Y = \left\{-1,+1\right\}$ навчальні об'єкти можна розуміти як позитивні й негативні електричні заряди; коефіцієнти γ_i – як абсолютну величину цих зарядів; ядро $K\left(z\right)$ – як залежність потенціалу від відстані до заряду; а саму задачу класифікації – як відповідь на запитання: який знак має електростатичний потенціал у заданій точці простору u. Помітимо, що в електростатиці $K\left(z\right) = \frac{1}{z}$ або $K\left(z\right) = \frac{1}{z+a}$, але для наших цілей зовсім не обов'язково брати саме таке ядро.

Алгоритм (3.4) має досить багатий набір з $2m$ параметрів y_i, m_i. Найпростіший і історично найперший метод їх налаштування представлено в *Алгоритмі* 1. Він налаштовує тільки ваги γ_i, припускаючи, що радіуси потенціалів h_i і ядро K обрані заздалегідь. Ідея дуже проста: якщо навчальний об'єкт x_i класифікується невірно, те потенціал класу y_i недостатній у точці x_i, і вага γ_i збільшується на одиницю. Вибір об'єктів на кроці 3 краще здійснювати не підряд, а випадковим чином. Цей метод не так вже і поганий, як можна було б подумати.

Переваги Алгоритму 1 у тому, що він є дуже ефективним, якщо навчальні об'єкти надходять потоком, а зберігати їх у пам'яті немає можливості або необхідності. У ті роки, коли метод потенційних функцій був придуманий, зберігання вибірки дійсно було великою проблемою. У цей час такої проблеми не існує, і тому Алгоритм 1 представляє скоріше історичний інтерес.

Недоліки Алгоритму 1. У даного алгоритму досить багато недоліків:

-він повільно сходиться;
-результат навчання залежить від порядку пред'явлення об'єктів;
-занадто грубо (із кроком 1) налаштовуються ваги γ_i;
-центри потенціалів чомусь розміщуються тільки в навчальних об'єктах;
-задача мінімізації числа потенціалів (ненульових γ_i) взагалі не ставиться;
-взагалі не налаштовуються параметри h_i .
Як результат, даний алгоритм не має високої якості класифікації.

4.2. Відбір еталонних об'єктів

Зазвичай об'єкти навчання не є рівноцінними. Серед них можуть перебувати типові представники класів – еталони. Якщо об'єкт, що підлягає класифікації, близький до еталона, те, швидше за все, він належить тому ж класу. Ще одна категорія об'єктів – неінформативні або периферійні. Вони щільно оточені іншими об'єктами того ж класу. Якщо їх вилучити з вибірки, це практично не відіб'ється на якості класифікації. Нарешті, у вибірку може потрапити деяка кількість шумових викидів – об'єктів, що перебувають «у товщі» чужого класу. Як правило, їхнє видалення тільки поліпшує якість класифікації.

Ці міркування приводять до ідеї виключити з вибірки шумові й неінформативні об'єкти, залишивши тільки мінімальну достатню кількість еталонів. Цим досягається декілька цілей одночасно – підвищується якість і стійкість класифікації, скорочується обсяг збережених даних і зменшується час класифікації, який витрачається на пошук найближчих еталонів. Крім того, виділення невеликої кількості еталонів у кожному класі дозволяє зрозуміти структуру класу.

У першу чергу введемо функцію відступу, яка дозволить оцінювати ступінь типовості об'єкта.

4.2.1. Поняття відступу об'єкта

Загальна формула (1) дозволяє ввести характеристику об'єктів, що показує, наскільки глибоко об'єкт занурений у свій клас.

Визначення 2. Відступом (margin) об'єкта $x_i \in X_m$ щодо алгоритму класифікації, який має вигляд

$$a\left(u\right) = \arg\min_{y \in Y} \Gamma_y\left(u\right),$$

називається величина

$$M\left(x_i\right) = \Gamma_{y_i}\left(x_i\right) - \max_{y \in Y \setminus y_i} \Gamma_y\left(x_i\right)$$

Відступ показує ступінь типовості об'єкта. Відступ від'ємний тоді й тільки тоді, коли алгоритм припускається помилки на даному об'єкті.

Залежно від значень відступу навчальні об'єкти умовно діляться на п'ять типів, у порядку зменшення відступу: еталонні, надійно класифіковані, приграничні, помилкові та шумові (рис. 3.1).

Еталонні. Еталонні об’єкти мають великий позитивний відступ, щільно оточені об’єктами свого класу і є найбільш типовими його представниками.

Надійно класифіковані. Надійно класифіковані об’єкти також мають позитивний відступ. Вилучення цих об’єктів з вибірки (за умови, що еталонні об’єкти залишаються), не впливає на якість класифікації. Фактично, вони не додають до еталонів ніякої нової інформації. Наявність неінформативних об’єктів характерне для вибірок надлишково великого обсягу.

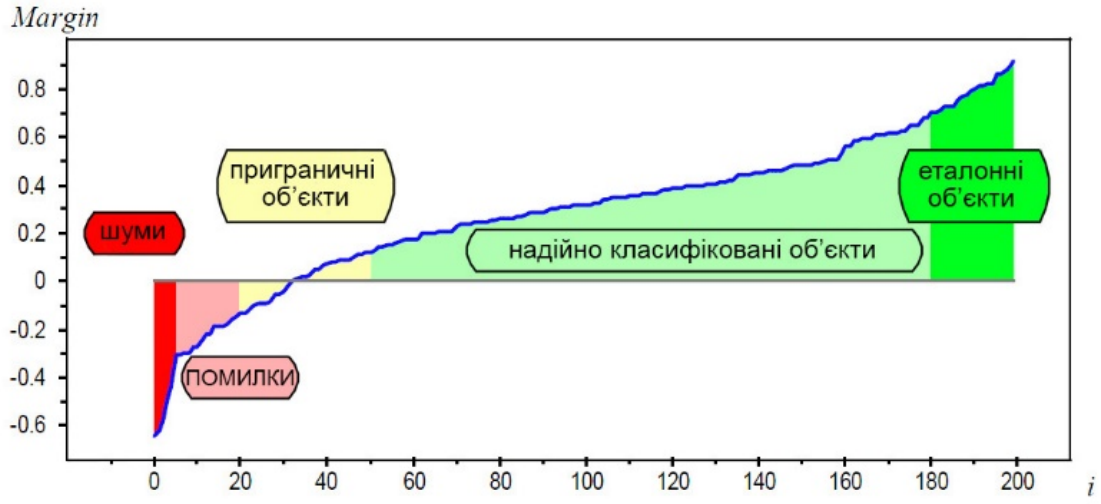


Рис. 3.1. Упорядковані за зростанням відступів M_i об’єкти вибірки, $i = 1,2,\ldots,200$. Умовний поділ об’єктів на п’ять типів.

Пригранічні об’єкти мають відступ, близький до нуля. Класифікація таких об’єктів нестійка в тому розумінні, що малі зміни метрики або складу навчальної вибірки можуть змінювати їхню класифікацію.

Помилкові об’єкти мають від’ємні відступи й класифікуються невірно. Можливою причиною може бути неадекватність алгоритмічної моделі, зокрема, невдала конструкція метрики ρ.

Шумові об’єкти або викиди – це невелике число об’єктів з великими від’ємними відступами. Вони щільно оточені об’єктами чужих класів і класифікуються невірно. Вони можуть виникати через грубі помилки або пропуски у вихідних даних, а також через відсутність важливої інформації, яка дозволила б віднести ці об’єкти до правильного класу.

Наведена типізація умовна. Не існує чіткої відмінності між «сусідніми» типами об’єктів. Зокрема, легко будуються приклади вибірок, що містять такі пари близьких об’єктів, що кожний з них може бути оголошений еталонним, а другий – надійно класифікованим.

Шумові і надійно класифіковані об’єкти доцільно видаляти з вибірки. Відповідний евристичний алгоритм буде описаний нижче.

Розподіл значень відступів у вибірці дає корисну додаткову інформацію не тільки про окремі об’єкти, але й про вибірку в цілому. Якщо основна маса об’єктів має позитивні відступи, то поділ вибірки можна вважати успішним. Якщо у вибірці занадто багато від’ємних відступів, то гіпотеза компактності

не виконується, і в даній задачі при обраній метриці застосовувати алгоритми типу kNN недоцільно. Якщо значення відступів концентруються поблизу нуля, то чекати надійної класифікації не доводиться, тому що занадто багато об'єктів виявляються в прикордонній «зоні непевності».

4.2.2. Алгоритм STOLP для відбору еталонних об'єктів

Ідея відбору еталонів реалізована в алгоритмі STOLP. Ми розглянемо його узагальнений варіант із довільною ваговою функцією $w(i,u)$. Будемо будувати метричний алгоритм $a(u;\Omega)$ виду (1), де $\Omega \subseteq X_m$ – множина еталонів

Алгоритм 2. Відбір еталонних об'єктів STOLP

Вхідні дані:

X_m – навчальна вибірка;

δ – поріг фільтрації викидів;

m_0 – припустима частка помилок;

Вихідні дані:

Множина опорних об'єктів $\Omega \subseteq X_m$;

Хід роботи алгоритму:

1: для всіх $x_i \in X_m$ перевірити, чи є x_i викидом:

2: якщо $M(x_i, X_m) < \delta$ то

3: $X_{m-1} := X_m \setminus \{x_i\}$; $m := m-1$;

4: Ініціалізація: взяти по одному еталону від кожного класу:

$$\Omega := \left\{ \arg\max_{x_i \in X_m(y)} M(x_i, X_m) \middle| y \in Y \right\};$$

5: Поки $\Omega \neq X_m$;

6: Виділити множину об'єктів, на яких алгоритм $a(u;\Omega)$ помиляється:

$$E := \{ x_i \in X_m \setminus \Omega \mid M(x_i, \Omega) < 0 \};$$

якщо $|E| < m_0$ то

8: вихід;

9: Приєднати до Ω об'єкт із найменшим відступом:

$$x_i := \arg\min_{x \in E} M(x, \Omega);\ \Omega := \Omega \cup \{x_i\};$$

Позначимо через $M\left(x_i,\Omega\right)$ відступ об'єкта x_i щодо алгоритму $a\left(x_i;\Omega\right)$. Великий негативний відступ свідчить про те, що об'єкт x_i оточений об'єктами чужих класів, отже, є викидом. Великий позитивний відступ означає, що об'єкт оточений об'єктами свого класу, тобто є або еталонним, або периферійним.

Алгоритм 2 починає з відсіву викидів (кроки 1–3). З вибірки X_m виключають всі об'єкти x_i з відступом $M\left(x_i, X_m\right)$, меншим заданого порогу δ. Якщо взяти $\delta = 0$, то об'єкти, що залишилися будуть класифіковані вірно. Замість δ можна задавати частку виключених об'єктів з найменшими значеннями відступу.

Потім формують початкове наближення - в Ω заносять по одному найбільш типовому представникові від кожного класу (крок 4).

Після цього починається процес послідовного «жадібного» нарощування множини Ω. На кожному кроці до Ω приєднують об'єкт x_i, що має мінімальне значення відступу. Так триває до тих пір, поки число помилок не виявиться менше заданого порогу m_0. Якщо покласти $m_0 = 0$, то буде побудований алгоритм $a\left(u,\Omega\right)$, який не допускає помилок на навчальних об'єктах, за винятком заздалегідь виключених викидів.

В результаті кожен клас буде представлений в Ω одним «центральним» еталонним об'єктом і масою «приграничних» еталонних об'єктів, на яких відступ набував найменших значень в процесі ітерацій. Параметр δ дозволяє регулювати ширину зазору між еталонами різних класів. Чим більше δ, тим далі від кордону класів будуть розташовуватися «приграничні» еталони, і тим більше простою, менш «порізаною» буде границя між класами.

В описаному варіанті алгоритм STOLP має відносно низьку ефективність. Для приєднання чергового еталона необхідно перебрати множину об'єктів $X_m \setminus \Omega$, і для кожного обчислити відступ щодо множини еталонів Ω. Загальне число операцій становить $O\left(\left|\Omega\right|^2 m\right)$. Для прискорення алгоритму можна додавати відразу по кілька еталонів, не перераховуючи відступів. Якщо при цьому вибирати, еталони, що додаються, досить далеко один від одного, то додавання одного з них практично не буде впливати на відступи інших. Аналогічно, на етапі відсівання викидів можна обчислити відступи тільки один раз, і потім відкинути всі об'єкти з відступами нижче δ. Реалізація цих ідей не показано в Алгоритмі 2, щоб не захаращувати його технічними подробицями.

Результатом роботи алгоритму STOLP є розбивка навчальних об'єктів на три категорії: шумові, еталонні й неінформативні. Якщо гіпотеза компактності вірна й вибірка досить велика, то основна маса навчальних

об’єктів виявиться неінформативною і буде відкинута. Фактично, цей алгоритм виконує стискання вихідних даних.

Запитання до розділу 4

1. Дайте визначення метричного алгоритму класифікації.

2. Які основні принципи побудови алгоритму за методом k - найближчих сусідів?

3. Чим відрізняється алгоритм k зважених найближчих сусідів від алгоритму k - найближчих сусідів?

4. З якою метою застосовують метод парзенівського вікна?

5. Дайте визначення відступу об’єкта при застосуванні алгоритмів класифікації. Що показує даний відступ об’єкта?

Розділ 5
Лінійні методи класифікації

Лінійні моделі широко використовуються в машинному навчанні завдяки їх відносній простоті, у деяких випадках гарній інтерпретації та наявності глибоко пророблених чисельних методів. Найпростішим обґрунтуванням лінійного класифікатора служить його аналогія з нервовою клітиною – нейроном. Персептронні принципи навчання, спочатку запозичені з нейрофізіології, потім знайшли математичні обґрунтування й з погляду градієнтних методів мінімізації емпіричного ризику, і з погляду байесовської теорії класифікації, і з погляду статистичних оцінок узагальнюючої здатності.

Нервова клітина і модель МакКаллока-Питтса

Лінійний класифікатор є найпростішою математичною моделлю нервової клітини – нейрона. Рис. 5.1.

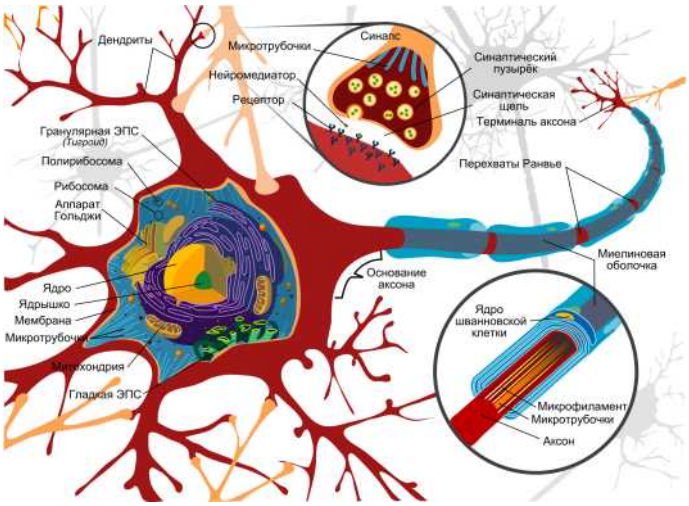


Рис. 5.1. Нервова клітина

Нейрон має велику кількість розгалужених відростків – дендритів, і одне довге тонке волокно – аксон, на кінці якого перебувають синапси, що примикають до дендритів інших нервових клітин. Нервова клітина може перебувати у двох станах: звичайному й збудженому. Клітина збуджується, коли в ній накопичується достатня кількість позитивних зарядів. У збудженому стані клітина генерує електричний імпульс величиною близько 100 мВ і тривалістю близько 1 мс, який проходить по аксону до синапсів. Синапс при приході імпульсу виділяє речовину, що сприяє проникненню позитивних зарядів усередину сусідньої клітини, яка пов’язана з даним синапсом. Синапси мають різну здатність концентрувати цю речовину, причому деякі навіть перешкоджають її виділенню – вони називаються гальмуючими. Після збудження в клітині наступає період релаксації – якийсь час вона не здатна генерувати нові імпульси.

Нервову клітину можна розглядати як обладнання, яке на кожному такті своєї роботи приймає заряди величиною $x^j = f_j\left(x\right)$ від n входів – синапсів, що примикають до її дендритів. Заряди, які надходять, додаються з вагами w_j. Якщо вага w_j позитивна, то j-й синапс збуджуючий, якщо негативний, то гальмуючий.

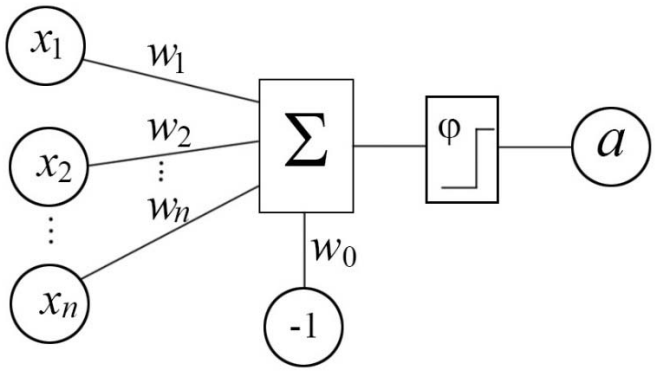


Рис. 5.2. Блок-схема штучного нейрона

Якщо сумарний заряд перевищує поріг активації w_0, то нейрон збуджується й видає на виході +1, інакше видається –1.

$$a\left(x\right) = \varphi\left(\sum_{j=1}^{n} w_j x_j - w_0\right) \tag{5.1}$$

Функцію $\varphi\left(z\right) = sign\left(z\right)$, що перетворює значення сумарного імпульсу у вихідне значення нейрона, називають функцією активації. У загальному випадку це не обов'язково порогова функція. Використовують також «згладжену» граничну функцію – гіперболічний тангенс $\varphi\left(z\right) = th\left(z\right)$ і інші (рис. 5.3).

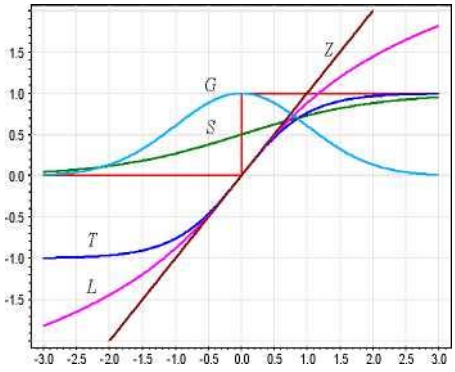


Рис. 5.3. Стандартні функції активації $\varphi\left(z\right)$:

$\theta\left(z\right) = \left[z \geq 0\right]$ – порогова функція Хевісайда;

$\sigma\left(z\right)$ – сигмоїдна функція (S);

$th\left(z\right) = 2\sigma\left(2z\right) - 1$ – гіперболічний тангенс (T);

$\ln\left(z+\sqrt{z^{2+1}}\right)$ – логарифмічна функція (L);

$\exp\left(-z^2/2\right)$ – гаусова функція (G);

z – лінійна функція (Z);

Таким чином, лінійний класифікатор (5.1) є математичною моделлю нейрона. Цю модель запропонували в 1943 році МакКаллок і Піттс.

Конективізм і нейронні мережі.

Нервова система складається з величезного числа зв'язаних один з одним нейронів, що поширюють спрямовані хвилі імпульсів. Швидкість поширення імпульсів становить приблизно 100 м/с. Людина здатна вирішувати складні задачі розпізнавання й прийняття рішень за десяті частки секунди, звідки випливає, що необхідні для цього нейрообчислення виконуються не більш ніж за 10^2 послідовних тактів. Кора головного мозку людину містить порядку 10^{11} нейронів, і кожний нейрон має синаптичні зв'язки з 10^3–10^4 інших нейронів. Нейрообчислення виконуються з великим ступенем паралелізму, і для прийняття одного рішення може бути задіяне величезне число нейронів.

Є гіпотеза, що, з'єднавши велику кількість елементарних класифікаторів (скажемо, лінійних, але обов'язково через нелінійні функції активації), можливо створити універсальну машину, здатну навчатися розв'язку будь-яких задач, подібно тому, як це робить людський мозок. На основі цієї ідеї, висловленої ще в 50-і роки й названої принципом конективізму, будуються штучні нейронні мережі.

Насправді механізми функціонування нервових кліток набагато складніше описаних вище. У нейрофізіології відомі десятки різних типів нейронів, і багато з них функціонують інакше. Однак у теорії штучних нейронних мереж не ставиться задача максимально точного відтворення функцій біологічних нейронів. Мета полягає у тому, щоб підглянути деякі принципи в живій природі й використовувати їх для побудови обладнання, яке має властивість до навчання.

Ми почнемо з лінійних класифікаторів, а до задач навчання штучних нейронних мереж повернемося пізніше.

5.1. Апроксимація й регуляризація емпіричного ризику

Розглянемо задачу класифікації із двома класами, $Y=\{-1,+1\}$.

Нехай модель алгоритмів являє собою параметричне сімейство відображень $a(x,w)=\varphi(f,w)$, де w – вектор параметрів. Функція $f(x,w)$ називається дискримінантною функцією. Функція φ- функція активації штучного нейрону. Наприклад, якщо $a(x,w)=sign(f,w)$, $f(x,w)>0$, то алгоритм a відносить об'єкт x до класу +1, інакше до класу -1. Рівняння $f(x,w)=0$ описує поверхню розділення.

Як зазвичай, задача навчання класифікатора $a\left(x,w\right)$ полягає в тому, щоб настроїти вектор параметрів w, маючи навчальну вибірку пар $X_m = \left(x_i, y_i\right)_{i=1}^m$.

Визначення 1. Величина $M_i\left(w\right) = y_i f\left(x_i w\right)$ називається відступом (margin) об'єкта x_i щодо алгоритму класифікації $a\left(x,w\right) = sign\left(f\left(x,w\right)\right)$.

Якщо $M_i\left(w\right) < 0$, то алгоритм $a\left(x,w\right)$ припускається помилки на об'єкті x_i. Чим більше відступ $M_i\left(w\right)$, тим вірніше й надійніше класифікація об'єкта x_i.

Мінімізація аппроксимированного емпіричного ризику. Визначимо функцію втрат виду $\lambda\left(M_i\left(w\right)\right)$, де $\lambda\left(M\right)$-монотонно незростаюча функція відступу, яка апроксимує зверху порогову функцію втрат: $\left[M < 0\right] \le \lambda\left(M\right)$. Тоді мінімізацію сумарних втрат можна розглядати як наближений метод мінімізації емпіричного ризику – числа помилок на навчальній вибірці:

$$Q\left(w, X_m\right) = \sum_{i=1}^{m}\left[M_i\left(w\right) < 0\right] \le Q\left(w, X_m\right) = \sum_{i=1}^{m}\lambda\left(M_i\left(w\right)\right) \to \min_w \qquad (5.2)$$

Деякі застосовувані на практиці функції втрат показані на рис. 5.4.

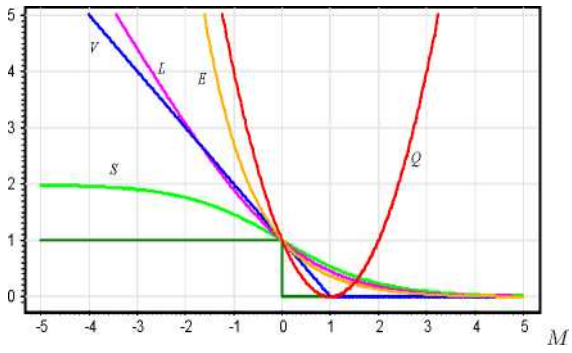


Рис. 5.4. Неперервні апроксимації порогової функції втрат $\left[M < 0\right]$.

$Q\left(M\right) = \left(1 - M\right)^2$ – квадратична,

$V\left(M\right) = \left(1 - M\right)_+$ – кусочно-лінійна,

$S\left(M\right) = 2\left(1 + e^M\right)^{-1}$ – сигмоїдна,

$L\left(M\right) = \log_2\left(1 + e^{-M}\right)$ – логістична,

$E\left(M\right)$ – експоненціальна.

Квадратична функція втрат не є монотонною, проте, вона відповідає лінійному дискримінанту Фішера. Кусочно-лінійна функція втрат відповідає

методу опорних векторів (SVM), сигмоїдна використовується в нейронних мережах, логістична – у логістичній регресії, експонентна – в алгоритмі бустинга AdaBoost. Кожний з перерахованих методів має свою розумну мотивацію, однак усі вони приводять до різних функцій втрат.

Виходячи з евристичного принципу максимізації відступів, неможливо відповісти на низку питань: які ще функції $\lambda\left(M\right)$ припустимі, які з них більш кращі та у яких ситуаціях.

5.2. Лінійна модель класифікації

Нехай X – простір об'єктів; $Y = \left\{-1, +1\right\}$ – множина припустимих відповідей; об'єкти описуються n числовими ознаками $f_j : X \to R,\ j = 1,2,\ldots,n$. Вектор $x = \left(x^1, x^2, \ldots, x^n\right) \in R^n$, де $x^j = f_j\left(x\right)$, називають ознаковим описом об'єкта x.

Якщо дискримінантна функція визначається як скалярний добуток вектора x і вектора параметрів $w \in R^n$, то виходить *лінійний класифікатор*:

$$a\left(x, w\right) = \varphi\left(\left\langle w, x\right\rangle - w_0\right) = sign\left(\sum_{j=1}^{n} w_j f_j\left(x\right) - w_0\right), \qquad (5.3)$$

де $\varphi\left(z\right) = sign\left(z\right)$ – функція актиівції,

w_j – вагові коефіцієнти синаптичних зв'язків,

w_0 – поріг активації,

$w, x \in R^{n+1}$, за умови введення константної ознаки $f_0\left(x\right) \equiv -1$

Рівняння $\left\langle w, x\right\rangle = 0$ задає гіперплощину, яка розділяє класи в просторі R^n. Якщо вектор x перебуває по одну сторону гіперплощини з її напрямним вектором w, то об'єкт x відносять до класу +1, інакше – до класу –1.

Параметр w_0 іноді опускають. Іноді вважаються, що серед ознак є константа, $f_j\left(x\right) \equiv -1$, і тоді роль вільного коефіцієнта w_0 відіграє параметр w_j.

5.3. Метод стохастичного градієнта

Нехай задана навчальна вибірка $X_m = \left\{\left(x_i, y_i\right)\right\}_{i=1}^{m},\ x_i \in R^n,\ y_i \in \left\{-1, +1\right\}$. Потрібно знайти вектор ваг $w \in R$, при якім досягається мінімум апроксимованого емпіричного ризику:

$$Q\left(w\right) = \sum_{i=1}^{m} \lambda\left(w\right) \to \min_{w} \qquad (5.4)$$

Застосуємо для мінімізації $Q\left(w\right)$ метод градієнтного спуску. У цьому методі вибирається деяке початкове наближення для вектора ваг w, потім запускається ітераційний процес, на кожному кроці якого вектор w змінюється в напрямку найбільш швидкого зменшування значення функціонала Q. Цей напрямок протилежний вектору градієнта

$$Q'\left(w\right)=\left(\frac{\partial Q\left(w\right)}{\partial w_j}\right)_{j=1}^{n}$$

$$w^{(t+1)}:=w^{(t)}-\eta Q'\left(w^{(t)}\right),$$

де $\eta>0$ – величина кроку в напрямку антиградієнта, називана також темпом навчання (learning rate). Припускаючи, що функція втрат λ дифференційована, розпишемо градієнт:

$$w^{(t+1)}=w^{(t)}-\eta\sum_{i=1}^{m}\lambda'\left(w^{(t)}\right) \tag{5.5}$$

Прискорення збіжності

Кожний прецедент $\left(x_i,y_i\right)$ вносить адитивний внесок у зміну вектора w, але вектор w змінюється тільки після перебору всіх m об'єктів. Збіжність ітераційного процесу можна поліпшити, якщо вибирати прецеденти $\left(x_i,y_i\right)$ по одному, для кожного робити градієнтний крок і відразу обновляти вектор ваг:

$$w^{(t+1)}:=w^{(t)}-\eta\lambda_a'\left(w^{(t)}\right) \tag{5.6}$$

У методі стохастичного градієнта (stochastic gradient, SG) прецеденти перебираються у випадковому порядку, (Алгоритм 1). Якщо ж об'єкти пред'являти в деякому фіксованому порядку, процес може зациклитися або розійтися.

Ініціалізація ваг може проводитися різними способами. Стандартна рекомендація – брати невеликі випадкові значення, $w_j:=random\left(-\frac{1}{2n},\frac{1}{2n}\right)$. Іноді ваги ініціалізують нулем. Іноді беруть оцінки

$$w_j:=\frac{\left\langle y,f_j\right\rangle}{\left\langle f_i,f_j\right\rangle},\quad j=1,2,\ldots,n \tag{5.7}$$

де $f_j=\left(f_j\left(x_i\right)\right)_{i=1}^{m}$ – вектор значень $j-$ї ознаки, $y=\left(y_i\right)_{i-1}^{m}$ – вектор відповідей. Ці оцінки є точними в одному нереалістичному окремому випадку – коли функція втрат квадратична й ознаки статистично незалежні.

Критерій останову в Алгоритмі 1 заснований на приблизній оцінці функціонала Q методом експонентної ковзної середньої. Обчислення точного значення по всім m об'єктам занадто обчислювально трудомістко. Коли градієнтний метод підходить до області мінімуму, оцінка ковзного середнього стабілізується й наближається до точного значення функціонала. Параметр λ можна покласти рівним $\frac{1}{m}$. У випадку надлишково довгої вибірки його рекомендується збільшувати.

Алгоритм 1. Метод стохастичного градієнта.
Вхідні дані:
X_m – навчальна вибірка;
η – темп навчання;
r – темп забування.
Вихідні дані:
Синаптичні ваги $w_j,\ j = 1,2,\ldots,n$;
Хід роботи алгоритму
1: Ініціалізувати ваги $w_j,\ j = 1,2,\ldots,n$;
2: Ініціалізувати поточну оцінку функціонала:

$$\bar{Q} = \frac{1}{m}\sum_{i=1}^{m}\lambda\left(w\right);$$

3: **повторювати**
вибрати об'єкт x_i з X_m (наприклад, випадковим чином);
обчислити вихідне значення алгоритму $a\left(x_i, w\right)$
обчислити помилку: $\varepsilon_i := \lambda\left(\left\langle w, x_i\right\rangle y_i\right)$;
зробити крок градієнтного спуску: $w := w - \eta\lambda'\left(w\right)$;
оцінити значення функціонала: $\bar{Q} := \left(1-r\right)\bar{Q} + \lambda\varepsilon_i$;
8: **поки** значення $\bar{Q}$ не стабілізується і/або ваги w не перестануть змінюватися;

Переваги методу SG
-Метод легко реалізується і легко узагальнюється на нелінійні класифікатори і на нейронні мережі – суперпозиції лінійних класифікаторів.
-Метод підходить для динамічного навчання, коли навчальні об'єкти поступають потоком, і вектор ваг оновлюється при появі кожного об'єкта.
-Метод дозволяє налаштовувати ваги на надлишково великих вибірках, за рахунок того, що випадкової підвибірки може виявитися досить для навчання.

Недоліки методу SG

-Функціонал Q, як правило, має багато екстремумів, і процес може сходитися до локального мінімуму, сходитися дуже повільно або сходитися зовсім.

-При великій розмірності простору n або малої довжині вибірки m можливе зворотне перенавчання. При цьому різко зростає норма вектору ваг, з'являються великі за абсолютною величиною позитивні і негативні ваги, класифікація стає нестійкою – малі зміни навчальної вибірки, початкового наближення, порядку пред'явлення об'єктів або параметрів алгоритму η, λ можуть сильно змінити результуючий вектор ваг, збільшується ймовірність помилкової класифікації нових об'єктів.

-Якщо функція втрат має горизонтальні асимптоти, то процес може попасти в стан «паралічу». Чим більше значення скалярного добутку $\langle w, x_i \rangle$, тим ближче значення похідної λ' до нуля, тим менше приріст у виразі

$$w := w - \eta \lambda_a'(w).$$

Якщо ваги w_j потрапили в область більших значень, то у них практично не залишається шансів вибратися з цієї «мертвої зони».

Алгоритм 2. Метод стохастичного середнього градієнта.

Вхідні дані:

X_m – навчальна вибірка;

η – темп навчання;

r – темп забування.

Вихідні дані:

Синаптичні ваги $w_j,\ j = 1,2,\ldots,n$;

Хід роботи алгоритму

1: Ініціалізувати ваги $w_j,\ j = 1,2,\ldots,n$;

2: Ініціалізувати поточну оцінку функціонала:

3: Ініціалізувати градієнти: $G_i := \lambda_i'(w),\ i = 1,2,\ldots,m$

4: Ініціалізувати оцінку функціонала: $\bar{Q} = \frac{1}{m}\sum_{i=1}^{m} \lambda(w)$;

3: **повторювати**

вибрати об'єкт x_i з X_m (наприклад, випадковим образом);

обчислити вихідне значення алгоритму $a(x_i, w)$

обчислити функцію втрат: $\varepsilon_i := \lambda(w)$;

обчислити градієнт: $G_i := \lambda_i'(w)$,

зробити крок градієнтного спуску: $w := w - \eta \frac{1}{m}\sum_{i=1}^{m} G_i$;

оцінити значення функціонала: $\bar{Q} := (1-r)\bar{Q} + \lambda\varepsilon_i$;

8: **поки** значення $\bar{Q}$ не стабілізується і/або ваги w не перестануть змінюватися.

5.3.1. Класичні окремі випадки

Адаптивний лінійний елемент. Розглянемо випадок, коли функція втрат квадратична, $\lambda(M) = (M-1)^2$. Тоді правило відновлення ваг на кожній ітерації методу стохастичного градієнта матиме вигляд:

$$w := w - \eta(\langle w, x_i \rangle - y_i)x_i \tag{5.8}$$

Це правило запропоноване Видроу й Хоффом в 1960 році й називається дельта-правилом (delta-rule), а сам лінійний нейрон – *адаптивним лінійним елементом* або ADALINE.

Персептрон Розенблатта. В 1957 році Розенблатт запропонував евристичний алгоритм навчання нейрона, заснований на принципах нейрофізіології. Експериментально було встановлено, що при синхронному збудженні двох зв'язаних нервових клітин синаптичний зв'язок між ними підсилюється. Чим частіше синапс угадує правильна відповідь, тем сильніше стає зв'язок. Своєрідне тренування зв'язку приводить до поступового запам'ятовування інформації. Якщо ж синапс починає часто помилятися або взагалі перестає використовуватися, зв'язок слабшає, інформація починається забуватися. Таким чином, пам'ять реалізується в синапсах. У математичній моделі нейрона, роль пам'яті відіграє вектор синаптичних ваг w.

Дане правило навчання неважко формалізувати. Ознаки будемо поки вважати бінарними, $f_j(x) \in \{0,1\}$. Відповіді також приймають тільки два значення, $y_i \in \{-1,1\}$. Припустимо, що відразу після одержання класифікації об'єкта $a(x_i)$ стає відома правильна відповідь y_i. Можливі три випадки.

1. Якщо відповідь $a(x_i)$ збігається з y_i, то вектор ваг змінювати не потрібно.

2. Якщо $a(x_i) = -1$ і $y_i = 1$, то вектор ваг w збільшується. Збільшувати має сенс тільки ті ваги w_j, для яких $f_j(x_i) \neq 0$; зміна інших компонент не вплине на результат. Покладемо $w := w + \eta x_i$, де $\eta > 0$ - темп навчання.

3.Якщо $a(x_i) = 1$ і $y_i = -1$, то вектор ваг зменшується: $w := w - \eta x_i$.

Ці три випадки поєднуються в так зване правило Хэбба:

$$\textbf{якщо } \langle w, x_i \rangle y_i < 0 \textbf{ то } w := w + \eta x_i y_i \tag{5.9}$$

Легко перевірити, що воно точно відповідає градиєнтному кроку (5), якщо брати кусочно-лінійну функцію втрат $\lambda\left(M\right)=\left(-M\right)_+$. Однак формула (5) вірна для довільних ознак, не обов'язково бінарних.

Для правила Хэбба доведена теорема збіжності, яка також слушна для довільних дійсних ознак.

Теорема 1 (Новиков, 1962). Нехай $X=R^n, Y=\left\{-1,1\right\}$, і вибірка X_m лінійно роздільна – існує вектор $\tilde{w}$ і позитивне число δ такі, що $\left\langle \tilde{w}, x_i \right\rangle y_i > \delta$ для всіх $i=1,2,\ldots,m$. Тоді Алгоритм 1 с правилом Хэбба (8) за скінченне число виправлень знаходить вектор ваг, що розділяє навчальну вибірку без помилок, причому з будь-якого початкового наближення w^0, при будь-якому $\eta>0$, незалежно від порядку пред'явлення об'єктів. Якщо $w^0=0$, то достатнє число виправлень вектору ваг не перевершує

$$t_{\max}=\left(\frac{D}{\delta}\right)^2, \quad \text{де} \quad D=\max_{x\in X_m}\left\|x\right\|$$

5.3.2. Евристики для поліпшення градієнтних методів навчання

Розглянемо евристичні прийоми і рекомендації, що компенсують недоліки градієнтних методів навчання. Усі вони повною мірою відносяться до навчання нейронних мереж, включаючи широко відомий метод зворотного поширення помилок. Різних тонкощів настільки багато, що застосування градієнтного навчання по праву вважається мистецтвом.

Нормалізація даних. Градієнтний метод чутливий до масштабу вимірювання ознак. Якщо норма вектору об'єкта $\left\|x_i\right\|$ набуває великих значень, а функція втрат має горизонтальні асимптоти, то ітераційний процес може виявитися «паралізованим». Тому загальною практикою є попередня нормалізація ознак:

$$x^j := \frac{x^j - x^j_{\min}}{x^j_{\max} - x^j_{\min}} \quad \text{або} \quad x^j := \frac{x^j - x^j_c}{x^j_{ск}}, \; j=1,\ldots,n$$

де $x^j_{\min}, x^j_{\max}, x^j_c, x^j_{ск}$ – відповідно мінімальне, максимальне, середнє значення й середньоквадратичне відхилення j-ї ознаки.

Порядок пред'явлення об'єктів. Крім стандартної рекомендації брати об'єкти у випадковому порядку, є ще наступні міркування.

1. Найбільший зсув ваг очікується для того об'єкта, який найменш схожий на об'єкти, пред'явлені до нього. У загальному випадку досить важко знайти об'єкт, максимально інформативний на даному кроці навчання. Проста евристика полягає в тому, щоб поперемінно пред'являти об'єкти з різних

класів, оскільки об’єкти одного класу з більшою ймовірністю містять схожу інформацію. Ця техніка називається перетасуванням об’єктів (shuffling).

2. Ще одна евристика полягає в тому, щоб частіше пред’являти ті об’єкти, на яких була допущена помилка. Для цього ймовірність появи кожного об’єкта встановлюється пропорційно величині помилки на даному об’єкті. Цю евристику рекомендується застосовувати тільки в тих випадках, коли вихідні дані не містять викидів, інакше процес навчання може зосередитися на шумових об’єктах, які взагалі варто було б виключити з навчальної вибірки.

3. Проста для реалізації евристика полягає в тому, щоб зрівняти величину помилки на пред’явленому об’єкті з деяким порогом. Якщо помилка виявиться менше порогу, вектор ваг не модифікується. Логіка та ж, що в персептрона Розенблатта: якщо об’єкт непогано класифікується, то міняти ваги не потрібно. При цьому збільшується й швидкість налаштування.

Квадратична регуляризация

Можливі причини перенавчання

1. Надто багато об’єктів, але надто мало ознак.
2. Лінійна залежність ознак:

Нехай маємо класифікатор $a\left(x,w\right)=sign\left\langle w,x\right\rangle$;

Мультколініарність: $\exists u\in R^{n+1}:\forall x_i\in X_m\ \left\langle u,x_i\right\rangle=0$;

Неодиничний розв’язок: $\forall\gamma\in R\ a\left(x,w\right)=sign\left\langle w+\gamma u,x\right\rangle$

Прояви перенавчання:

1. Надто великі ваги різних знаків.
2. Нестійка дискримінантна функція $\left\langle w,x\right\rangle$
3. $Q\left(X_m\right)<<Q\left(X_k\right)$

Основний спосіб зменшення перенавчання: регуляпизація (скорочення ваг weight decay).

Штраф за збільшення норми вектора ваг:

$$\tilde{\lambda}_i\left(w\right)=\lambda_i\left(w\right)+\frac{\tau}{2}\left\|w\right\|^2=\lambda_i\left(w\right)+\frac{\tau}{2}\sum_{j=1}^{n}w_j^2\rightarrow\min_w$$

Градієнт: $\lambda_i'\left(w\right)=\lambda_i\left(w\right)+\tau w$

Модифікація градієнтного кроку:

$w:=w\left(1-\eta\tau\right)-\eta\lambda_i'\left(w\right)$

В теорії нейронних мереж називається також *скороченням ваг* (weights decay). Щоб обмежити ріст абсолютних значень ваг, до функціоналу $Q\left(w\right)$, який мінімізують, додається штрафний доданок:

$$Q_\tau\left(w\right)=Q\left(w\right)+\frac{\tau}{2}\left\|w\right\|^2$$

Це приводить до появи адитивної поправки в градієнті:

$$Q'_\tau\left(w\right) = Q'_\tau\left(w\right) + \tau w$$

У результаті правило відновлення ваг набуває вигляду:

$$w := w\left(1 - \eta\tau\right) - \eta Q'\left(w\right)$$

Таким чином, уся модифікація зводиться до появи невід'ємного множника $\left(1 - \eta\tau\right)$, що приводить до постійного зменшення ваг. Регуляризація запобігає паралічу, підвищує стійкість ваг у випадку мультиколінеарності, сприяє підвищенню узагальнюючої здатності алгоритму й зниженню ризику перенавчання. Керуючий параметр τ дозволяє знайти компроміс між точністю налаштування на конкретну вибірку й стійкістю ваг.

Недолік методу в тому, що параметр τ доводиться підбирати в режимі ковзного контролю, що пов'язане з великими обчислювальними витратами.

Вибір величини кроку

1. Відомо, що градієнтні методи сходяться до локального мінімуму, якщо величину кроку η зменшувати із числом ітерацій t. Точніше, збіжність гарантується при $\eta_t \to 0,\ \sum_{t=1}^{\infty}\eta_t = \infty,\ \sum_{t=1}^{\infty}\eta_t^2 < \infty$, зокрема можна покласти:

$$\eta_t = \frac{1}{t}$$

2. У методі якнайшвидшого градієнтного спуску вибирається *адаптивний крок* η, який є розв'язком одномірної задачі $Q\left(w - \eta Q'\left(w\right)\right) \to \min_\eta$. У багатьох випадках цю задачу вдається розв'язати аналітично. Зокрема, для алгоритму ADALINE із квадратичною функцією втрат: $\eta = \left\|x_i\right\|^{-2}$.

Вибивання з локальних мінімумів необхідне для запобігання збіжності до недостатньо гарних локальних розв'язків. Один з найпростіших способів полягає в тому, щоб при кожній стабілізації функціонала робити випадкові модифікації вектора ваг у досить великому околі поточного значення й запускати процес градієнтного спуску з нових точок. Це прийом називають *струшуванням коефіцієнтів* (jog of weights). По суті справи, він є симбіозом градієнтного методу й *випадкового локального пошуку* (stochastic local search).

Ранній останов. Надмірна оптимізація може вести до перенавчання. Вузький глобальний мінімум функціонала $Q\left(w, X_m\right)$ гірше більш широкого, але стійкого локального мінімуму. Для запобігання попаданню в такі «ущелини» застосовується *ранній останов* (early stopping): у ході ітерацій обчислюється який-небудь *зовнішній критерій*, наприклад, середня втрата на

незалежній контрольній вибірці, і якщо він починає зростати, процес налаштування припиняється.

Запитання до розділу 5

1.Зобразіть блок-схему штучного нейрона та запишіть формулу його функціонування.

2. Яка відмінність лінійної моделі класифікатора ?

3. В чому полягає метод стохастичного градієнта?

4. Які переваги та недоліки методу стохастичного градієнта?

5. Які можливі причини перенавчання?

Розділ 6
Логістична регресія та метод опорних векторів

Метод *логістичної регресії* заснований на досить сильних імовірнісних припущеннях, які мають відразу кілька цікавих наслідків. Перш за все, лінійний алгоритм класифікації виявляється оптимальним байесовским класифікатором. Окрім того, однозначно визначається функція втрат. Також, виникає цікава додаткова можливість поряд із класифікацією об'єкта одержувати чисельні оцінки ймовірності його приналежності кожному із класів.

6.1. Обґрунтування логістичної регресії

У нормальному дискримінантному аналізі доводиться, що якщо щільності класів нормальні й мають однакові матриці коваріфації, то оптимальний байєсовський класифікатор лінійний. Виникає питання: а чи тільки в цьому випадку? Виявляється, що ні – він залишається лінійним при менш жорстких припущеннях.

Базові припущення. Нехай класів два, $Y = \{-1, +1\}$, об'єкти описуються n числовими ознаками $f_j : X \to R,\ j = 1,\ldots,n$. Будемо вважати, що $X = R^n$, ототожнюючи об'єкти з їх ознаковими описами: $x \equiv \left(f_1(x),\ldots,f_n(x)\right)$.

Гіпотеза 1. Множина прецедентів $X \times Y$ є імовірнісним простором. Вибірка прецедентів $X_m = \left(x_i, y_i\right)_i^m$ отримана випадково й незалежно згідно з імовірнісним розподілом із щільністю $p(x,y) = P_y p_y(x) = P(y/x)\,p(x)$, де P_y – апріорні ймовірності, $p_y(x)$ – функції правдоподібності, $P(y/x)$ – апостеріорні ймовірності класів $y \in Y$.

Визначення 2. Щільність розподілу $p(x),\ x \in R^n$ називається експонентною, якщо $p(x) = \exp\left(c(\delta)\langle\theta, x\rangle + b(\delta,\theta) + d(x,\delta)\right)$, де параметр $\theta \in R^n$ називають зсувом, параметр δ називають розкидом, b, c, d – довільні числові функції.

Клас експонентних розподілів дуже широкий. До нього відносять багато безперервних і дискретних розподілів: рівномірний, нормальний, гіпергеометричний, пуасонівський, біноміальний, Г-розподіл, і інші.

Гіпотеза 2. Функції правдоподібності класів $p_y(x)$ належать експонентному сімейству плотностей, мають рівні значення параметрів d і δ, але відрізняються значеннями параметра зсуву θ_y.

Основна теорема. Нагадаємо, що оптимальний байєсовський класифікатор має вигляд $a(x) = \arg\max\limits_{y \in Y} \lambda_y$, λ_y – штраф за помилку на об'єктах класу y.

У випадку двох класів

$$a\left(x\right) = sign\left(\lambda_{+}P\left(+1 / x\right) - \lambda\_P\left(-1 / x\right)\right) = sign\left(\frac{P\left(+1 / x\right)}{P\left(-1 / x\right)} - \frac{\lambda\_}{\lambda_{+}}\right)$$

Теорема 4.2. Якщо слушні гіпотези 1, 2, і серед ознак $f_1\left(x\right),\ldots,f_n\left(x\right)$ є константа, то:

1) байєсовський класифікатор є лінійним: $a\left(x\right) = sign\left(\left\langle w,x\right\rangle - w_0\right)$, де $w_0 = \ln\left(\frac{\lambda_{-1}}{\lambda_{+}}\right)$, а вектор w не залежить від штрафів $\lambda\_,\lambda_{+}$;
2) апостеріорна ймовірність приналежності довільного об'єкта $x \in X$ класу $y \in \left\{-1,+1\right\}$ може бути обчислена за значенням дискримінантної функції: $P\left(y / x\right) = \sigma\left(\left\langle w,x\right\rangle y\right)$, де $\sigma\left(z\right) = \frac{1}{1+e^{-z}}$ – сигмоїдна функція.

6.2. Метод стохастичного градієнту для логістичної регресії

Принцип максимуму правдоподібності. Для настроювання вектора ваг w по навчальній вибірці X_m будемо максимізувати логарифм правдоподібності вибірки:

$$L\left(w,X_m\right) = \log_2 \prod_{i=1}^{m} p\left(x_i,y_i\right) \to \max_w$$

Згідно з визначенням умовної ймовірності, $p\left(x,y\right) = P\left(y / x\right)p\left(x\right)$, де щільності розподілу об'єктів $p\left(x\right)$ не залежать від вектора параметрів w. Апостеріорні ймовірності виражають згідно з Теоремою 2 через лінійну дискримінантну функцію: $P\left(y / x\right) = \sigma\left(\left\langle w,x\right\rangle y\right)$. Таким чином,

$$L\left(w,X_m\right) = \sum_{i=1}^{m} \log_2 \sigma\left(\left\langle w,x_i\right\rangle y_i\right) + const\left(w\right) \to \max_w$$

Максимізація правдоподібності $L\left(w,X_m\right)$ еквівалентна мінімізації функціонала $\tilde{Q}\left(w,X_m\right)$, який гладко апроксимує емпіричний ризик:

$$\tilde{Q}\left(w,X_m\right) = \sum_{i=1}^{m} \log_2\left(1+\exp\left(-\left\langle w,x_i\right\rangle y_i\right)\right) \to \min_w \qquad (6.1)$$

Таким чином, логістична функція втрат $L\left(M\right) = \log_2\left(1+e^{-M}\right)$ є наслідком экспонентності класів і принципу максимуму правдоподібності.

Градієнтний крок. Запишемо градієнт функціонала $\tilde{Q}\left(w\right)$, скориставшись виразом для похідної сигмоїдної функції

$\sigma'(z) = \sigma(z)(1-\sigma(z)) = \sigma(z)\sigma(-z)$, і одержимо логістичне правило відновлення ваг для градієнтного кроку в методі стохастичного градієнта:

$$w := w + \eta y_i x_i \sigma\left(-\langle w, x_i \rangle y_i\right) \qquad (6.2)$$

де (x_i, y_i) – прецедент, який пред'являють, n – темп навчання.

Аналогія із правилом Хебба. Логістична функція втрат є згладженим варіантом кусочно-лінійної функції втрат, відповідної до правила Хебба, рис. 6.1.

Нагадаємо, що правило Хебба має вигляд:

якщо $\langle w, x_i \rangle y_i < 0$ **то** $w := w + \eta x_i y_i$

Тому й логістичне правило (6.2) виявляється, у свою чергу, згладженим варіантом правила Хебба, рис. 6.2:

$$w := w + \eta y_i x_i \left[\langle w, x_i \rangle y_i < 0\right]$$

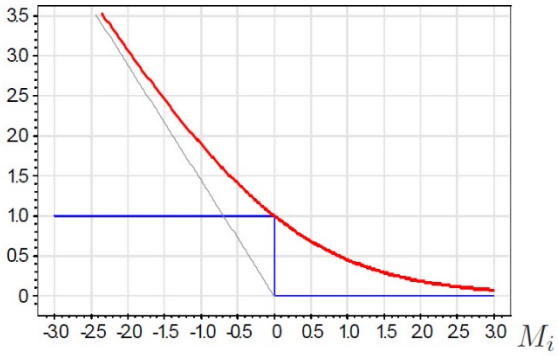


Рис. 6.1. Логарифмічна функція втрат $\log_2\left(1+e^{-M_i}\right)$ та її похила компонента

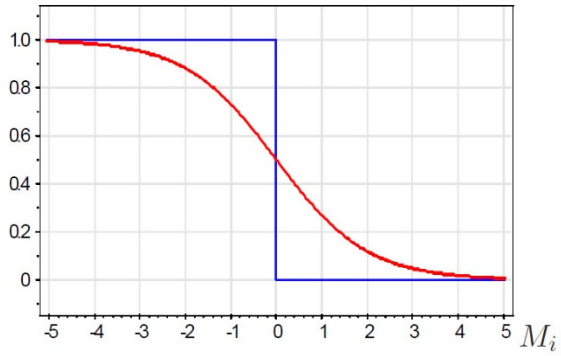


Рис. 6.2. Правило Хебба: порогове $\left[M_i < 0\right]$ та зглажене $\sigma(-M_i)$

У правилі Хебба зсув ваг відбувається тільки коли на об'єкті x_i допускається помилка. У логістичнім правилі зсув тим більше, чим менше відступ $M_i(w) = \langle w, x_i \rangle y_i$, тобто чим серйозніше помилка. Навіть якщо помилки немає, але об'єкт близький до границі класів, ваги модифікуються так, щоб границя пройшла якнайдалі від об'єкта. Тим самим градієнтна мінімізація реалізує стратегію збільшення зазору (margin) між навчальними об'єктами й поділяючою поверхнею, що сприяє поліпшенню узагальнюючої здатності.

Переваги логістичної регресії.

1. Як правило, логістична регресія дає кращі результати в порівнянні з лінійним дискримінантом Фішера (оскільки вона заснована на менш жорстких гіпотезах), а також у порівнянні з дельта-правилом і правилом Хебба (оскільки вона використовує «більш правильну» функцію втрат).
2. Можливість оцінювати апостеріорні ймовірності й ризики.

Недоліки логістичної регресії.
1. Оцінки ймовірностей і ризиків можуть виявитися неадекватними, якщо не виконуються припущення Теореми 2.
2. Градиентный метод навчання логістичної регресії успадковує всі недоліки методу стохастичного градієнта. Практична реалізація повинна передбачати стандартизацію даних, відсівання викидів, регуляризацію (скорочення ваг), відбір ознак, і інші евристики для поліпшення збіжності.

6.3. Скоринг і оцінювання апостеріорних імовірностей

Скоринг. У випадку бінарних ознак, $X = \{0,1\}^n$, можна вважати, що функції правдоподібності класів $p_y(x)$ описуються біноміальними розподілами, отже, є експонентними. Це міркування служить додатковим обґрунтуванням бінаризації ознак, коли кожна небінарна початкова ознака замінюється одною або декількома бінарними.

У бінарному випадку обчислення лінійної дискримінантної функції зручно розглядати як підрахунок балів (score): якщо $f_j(x) = 1$, тобто ознака f_j спостерігається в об'єкта x, то до суми балів додається вага w_j. Класифікація проводиться шляхом порівняння набраної суми балів із граничним значенням w_0.

Вік	до 25	5
	25-40	10
	40-50	15
	50 і більше	10
Власність	власник	20
	співвласник	15
	наймач	10
	інше	5
Робота	керівник	15
	менеджер середньої ланки	10
	Службовець	5
	інше	0
Стаж	1/безробітний	0
	1…3	5
	3…10	10
	10 і більше	15
Робота чоловіка / дружини	немає/домогосподарка	0
	Керівник	10
	менеджер середньої ланки	5
	Службовець	1

Рис. 6.3. Фрагмент скорингової карти для задач прийняття кредитних рішень

Завдяки своїй простоті підрахунок балів або скоринг (scoring) користується великою популярністю в таких областях, як медицина, геологія, банківська справа, соціологія, маркетинг, і ін. Абсолютне значення ваги w_j можна інтерпретувати як ступінь важливості ознаки f_j, а знак $sign\left(w_j\right)$ показує, на користь якого класу свідчить наявність даної ознаки. Це важлива додаткова інформація про ознаки, що допомагає експертам краще розуміти і задачу, і класифікатор.

Після бінаризації ознак класифікатор представляється у вигляді так званої скорингової карти (score card), у якій перелічуються всі початкові ознаки, для кожного початкового – усі побудовані по ньому бінарні ознаки, для кожного бінарного – його вага. Маючи таку карту, класифікацію можна проводити за допомогою стандартної електронної таблиці або навіть вручну. Рис. 6.3.

Імовірнісний вихід і оцінювання ризиків.

Логістична функція σ переводить значення лінійної дискримінантної функції $\left\langle w,x\right\rangle$ в оцінку апостеріорної ймовірності того, що об'єкт x належить класу +1: $P\left(+1/x\right)=\sigma\left\langle w,x\right\rangle$. Ця властивість використовується в тих додатках, де поряд із класифікацією об'єкта x потрібно оцінити пов'язаний з ним ризик як математичне очікування втрат:

$$R\left(x\right)=\sum_{y\in Y}\lambda_y P\left(y/x\right)=\sum_{y\in Y}\lambda_y\sigma\left(\left\langle w,x\right\rangle y\right)$$

де λ_y – величина втрати при помилковій класифікації об'єкта класу y.

У практичних ситуаціях до оцінок апостеріорної ймовірності слід ставитися з обережністю. Теорема 2 гарантує, що $P\left(y/x\right)=\sigma\left(\left\langle w,x\right\rangle y\right)$ тільки для експонентних класів з рівними параметрами розкиду. В інших випадках оцінка ймовірності носить евристичний характер. На практиці експонентність рідко коли перевіряється, а гарантувати рівність розкиду взагалі не представляється можливим.

Імовірнісне калібрування

дозволяє перерахувати значення дискримінантної функції в оцінку апостеріорної ймовірності, коли умови теореми 2 не виконуються, і навіть коли класифікатор $a\left(x\right)=\mathsf{sign}\,f\left(x,w\right)$ не є лінійним. Передбачається, що апостеріорна ймовірність $P\left(+1/x\right)$ монотонно залежить від значення дискримінантної функції $f\left(x,w\right)$. Існує багато способів ввести модель цієї залежності, але ми розглянемо тільки один – калібрування Платта, яка основана на лінійно-сигмоїдальній моделі:

$$P\left(+1 / x\right) = \sigma\left(\alpha f\left(x, w\right) + \beta\right)$$

Для настроювання невідомих параметрів $\left(\alpha, \beta\right) \in R^2$ вирішується задача максимізації правдоподібності

$$\sum_{i=1}^{m} \log P\left(y_i / x_i\right) = \sum_{i=1}^{m} \log \sigma\left(\alpha y_i f\left(x_i, w\right) + \beta y_i\right) \to \max_{\alpha,\beta}$$

будь-яким стандартними чисельними методами оптимізації.

6.4. Метод опорних векторів

Ми розглядали персептрон, набір нейронів МакКаллока і Пітса, розташованих в одному шарі. Ми визначили метод, за допомогою якого ми могли б змінити ваги для того, щоб кожен нейрон навчився краще класифікувати об’єкти. Легко дізнатися, що персептрон не є універсальним класифікатором. Він може виділяти лише групи даних, якщо між ними можливо намалювати пряму лінію (або провести гіперплощину для вищих розмірностей). Це означає, що він не може навчитися розрізняти дві класи з логічною функцію XOR в 2D.

Алгоритм, який ми розглядаємо у цьому розділі, вирішує цю проблему. Такий алгоритм називають методо опорних векторів або Support Vector Machine (SVM). Це один з найпопулярніших алгоритмів сучасного машинного навчання. Він створений у 1992 році і з тих має велику популярність головним чином тому, що він часто (але не завжди) забезпечує дуже вражаючу ефективність класифікації на відповідних розмірах наборів даних.

SVM не працює з надзвичайно великими наборами даних, оскільки (як ми побачимо) обчислення не дуже добре масштабуються на великі навчальні вибірки. У цих випадках кількість обчислень стрімко зростає.

Розглянемо простий SVM, використовуючи cvxopt, вільно доступний обчислювач з інтерфейсом Python, щоб зробити важку роботу. Існує кілька різних реалізацій SVM доступних в Інтернеті. Деякі з них мають обгортки, щоб їх можна було використовувати з Python.

Метод SVM має декілька чудових властивостей.

По-перше, навчання SVM зводиться до задачі квадратичного програмування, що має єдиний розв’язок, який обчислюється досить ефективно навіть на вибірках у сотні тисяч об’єктів.

По-друге, розв’язок має властивість розрідженості: положення оптимальної поділяючої гіперплощини залежить лише від невеликої частки навчальних об’єктів. Вони й називаються опорними векторами; інші об’єкти фактично не задіються. Нарешті, за допомогою витонченого математичного прийому – введення функції ядра – метод узагальнюється на випадок нелінійних поділяючих поверхонь. Питання про вибір ядра, оптимального для даної прикладної задачі, дотепер залишається відкритою теоретичною проблемою.

6.4.1. Лінійно роздільна вибірка

Розглянемо задачу класифікації на два класи, які не перетинаються, у якій об'єкти описуються n-вимірними дійсними векторами: $X = R^n,\ Y = \{-1, +1\}$. Будемо будувати лінійний пороговий класифікатор:

$$a(x) = sign\left(\sum_{j=1}^{n} w_j x^j - w_0\right) = sign\left(\langle w, x\rangle - w_0\right), \qquad (6.3)$$

де $x = \left(x^1, \ldots, x^n\right)$ – ознаковий опис об'єкта x; вектор $w = \left(w^1, \ldots, w^n\right) \in R^n$ і скалярний поріг $w_0 \in R$ є параметрами алгоритму. Рівняння $\langle w, x\rangle = w_0$ описує гіперплощину, що розділяє класи в просторі R^n.

Припустимо, що вибірка $X_m = \left(x_i, y_i\right)_{i=1}^{m}$ лінійно роздільна й існують значення параметрів $w,\ w_0$, при яких функціонал числа помилок

$$Q\left(w, w_0\right) = \sum_{i=1}^{m}\left[y_i\left(\langle w, x_i\rangle - w_0\right) \le 0\right]$$

набуває нульового значення. Але тоді поділяюча гіперплощина не єдина. Можна вибрати інші її положення, що реалізують таке ж розбиття вибірки на два класи. Ідея методу полягає в тому, щоб розумним чином розпорядитися цією свободою вибору.

Оптимальна поділяюча гіперплощина. Зажадаємо, щоб поділяюча гіперплощина максимально далеко відстояла від найближчих до неї точок обох класів. Спочатку даний принцип класифікації виник з евристичних міркувань: цілком природно вважатися, що максимізація зазору (margin) між класами повинна сприяти більш надійній класифікації.

Нормування. Помітимо, що параметри лінійного порогового класифікатора визначені з точністю до нормування: алгоритм $a(x)$ не зміниться, якщо w і w_0 одночасно помножити на одну і ту ж позитивну константу. Зручно вибрати цю константу таким чином, щоб виконувалася умова

$$\min_{i=1,\ldots,m} y_i\left(\langle w, x_i\rangle - w_0\right) = 1 \qquad (6.4)$$

Множина точок $\left\{x \,\middle|\, -1 \le \langle w, x\rangle - w_0 \le 1\right\}$ описує смугу, що розділяє класи, див. рис. 6.4. Жоден з об'єктів навчальної вибірки не попадає всередину цієї смуги. Границями смуги служать дві паралельні гіперплощини з вектором нормалі w. Поділяюча гіперплощина проходить рівно по середині між ними. Об'єкти, найближчі до поділяючої гіперплощини, лежать на границях смуги, і саме на них досягається мінімум (6.4). У кожному із класів є хоча б один такий

об’єкт, а якщо ні, то поділяючу смугу можна було б ще трохи розширити й порушувався б принцип максимального зазору.

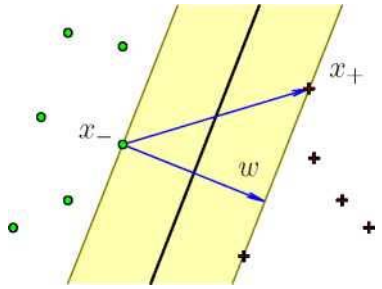


Рис. 6.4. Лінійно роздільна вибірка. Навчальні об’єкти x_- і x_+ перебувають на границі поділяючої смуги. Вектор нормалі w до поділяючої гіперплощини визначає ширину смуги.

Ширина поділяючої смуги. Щоб поділяюча гіперплощина якнайдалі відстояла від крапок вибірки, ширина смуги повинна бути максимальної. Нехай x_- і x_+ – два навчальні об’єкти класів –1 і +1 відповідно смуги, що лежать на границі. Тоді ширина смуги є

$$\left\langle \left(x_+ - x_-\right), \frac{w}{\|w\|} \right\rangle = \frac{\langle w, x_+ \rangle - \langle w, x_- \rangle}{\|w\|} = \frac{\left(w_0 + 1\right) - \left(w_0 - 1\right)}{\|w\|} = \frac{2}{\|w\|}$$

Ширина смуги максимальна, коли норма вектора w мінімальна.

Отже, у випадку лінійно роздільної вибірки одержуємо задачу квадратичного програмування: потрібно знайти значення параметрів w і w_0, при яких виконуються m обмежень-нерівностей і норма вектора w мінімальна:

$$\begin{cases} \langle w, w \rangle \to \min; \\ y_i \left(\langle w, x_i \rangle - w_0 \right) \geq 1, \ \ i = 1, \ldots, m \end{cases} \tag{6.5}$$

На практиці лінійно роздільні класи зустрічаються досить рідко. Тому постановку задачі (6.5) необхідно модифікувати так, щоб система обмежень була сумісна в будь-якій ситуації.

6.4.2. Лінійно нероздільна вибірка

Щоб узагальнити постановку задачі на випадок лінійно нероздільної вибірки, дозволимо алгоритму допускати помилки на навчальних об’єктах, але при цьому постараємося, щоб помилок було поменше. Введемо додаткові змінні $\xi_i \geq 0$ такі, що характеризують величину помилки на об’єктах $x_i, \ i = 1, \ldots, m$. Послабимо в (6.5) обмеження-нерівності й одночасно введемо в функціонал, який мінімізуємо, штраф за сумарну помилку:

$$\begin{cases} \frac{1}{2}\langle w, w\rangle + C\sum_{i=1}^{m}\xi_i \to \min\limits_{w, w_0, \xi}; \\ y_i\left(\langle w, x_i\rangle - w_0\right) \geq 1 - \xi_i,\ i = 1, \ldots, m; \\ \xi_i \geq 0,\ i = 1, \ldots, m \end{cases} \tag{6.6}$$

Позитивна константа C є керуючим параметром методу й дозволяє знаходити компроміс між максимізацією ширини поділяючої смуги й мінімізацією сумарної помилки.

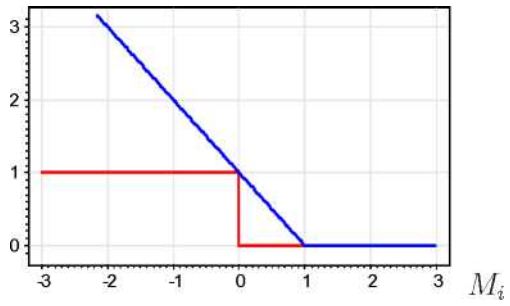


Рис. 6.5. Кусочно-лінійна апроксимація порогової функції втрат:

$$\left[M_i < 0\right] \leq \left(1 - M_i\right)_+$$

Регуляризація емпіричного ризику.

У задачах із двома класами $Y = \{-1, +1\}$ відступом (margin) об'єкта x_i від границі класів називається величина

$$M_i\left(w, w_0\right) = y_i\left(\langle w, x_i\rangle - w_0\right)$$

Алгоритм (14) припускається помилки на об'єкті x_i тоді й тільки тоді, коли відступ M_i негативний. Якщо $M_i \in (-1, +1)$, то об'єкт хі попадає усередину поділяючої смуги. Якщо $M_i > 1$, то об'єкт x_i класифікується правильно, і перебуває на деякім видаленні від поділяючої смуги.

Згідно (6.6) помилку ξ_i виражають через відступ M_i. Дійсно, з обмежень-нерівностей випливає, що $\xi_i \geq 0$ і $\xi_i \geq 1 - M_i$. В силу вимоги мінімізації суми $\sum_i \xi_i$ одна із цих нерівностей обов'язково повинна перетворитися в рівність. Отже, $\xi_i = \left(1 - M_i\right)_+$. Таким чином, задача (6.6) виявляється еквівалентною безумовній мінімізації функціонала Q, що не залежить від змінних ξ_i:

$$Q\left(w, w_0\right) = \sum_{i=1}^{m}\left(1 - M_i\left(w, w_0\right)\right)_+ + \frac{1}{2C}\left\|w\right\|^2 \to \min_{w, w_0} \tag{6.7}$$

Через те, що справедлива нерівність $\left[M_i < 0\right] \le \left(1 - M_i\right)_+$, рис. 6.5, функціонал (6.7) можна розглядати як верхню оцінку емпіричного ризику (числа помилкових класифікацій об'єктів навчальної вибірки), до якого доданий регуляризатор $\|w\|^2$, помножений на параметр регуляризації $\frac{1}{2C}$.

Заміна граничної функції втрат $\left[M < 0\right]$ кусочно-лінійною верхньою оцінкою $L\left(M\right) = \left(1 - M\right)_+$ робить функцію втрат чутливою до величини помилки. Функція втрат $L\left(M\right)$ штрафує об'єкти за наближення до границі класів.

Введення регуляризатора підвищує стійкість розв'язку w. У випадках, коли мінімум емпіричного ризику досягається на множині векторів w, регуляризація вибирає з них вектор з мінімальною нормою. Тим самим усувається проблема мультиколінеарності, підвищується стійкість алгоритму, поліпшується його узагальнююча здатність. Таким чином, принцип оптимальної поділяючої гіперплощини або максимізації ширини поділяючої смуги тісно пов'язаний з регуляризацією некоректно поставлених задач по А. Н. Тихонову.

Задача (6.7) відповідає принципу максимуму спільної правдоподібності, якщо прийняти модель щільності

$p\left(x_i, y_i \,/\, w\right) = z_1 \exp\left(-\left(1 - M_i\left(w, w_0\right)\right)_+\right)$ гаусівську модель апріорного розподілу вектора параметрів w

$$p\left(w; C\right) = z_2 \exp\left(-\frac{\|w\|^2}{2C}\right)$$

і не накладати ніяких обмежень на параметр w_0. Тут z_1, z_2 – нормувальні константи, C – гіперпараметр.

Розуміння ролі функції втрат і регуляризатора необхідно для того, щоб більш вільно поводитися з постановкою задачі, при необхідності модифікувати її, одержуючи методи, схожі на SVM, але такі, що володіють необхідними властивостями.

Двоїста задача

Запишемо функцію Лагранжа для задачі (6.6).

$$L\left(w, w_0, \xi, \lambda, \eta\right) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i - \sum_{i=1}^{m}\lambda_i\left(M_{\ i}\left(w, w_0\right) - 1 + \xi_i\right) - \sum_{i=1}^{m}\xi_i\eta_i =$$

$$= \frac{1}{2}\|w\|^2 - \sum_{i=1}^{m}\lambda_i\left(M_i\left(w, w_0\right) - 1\right) - \sum_{i=1}^{m}\xi_i\left(\lambda_i + \eta_i - C\right)$$

Де $\lambda = \left(\lambda_1, \ldots, \lambda_m\right)$ - вектор змінних, які є двоїстими до w;

$\eta = (\eta_1,\ldots,\eta_m)$ – вектор змінних, які є двоїстими до $\xi = (\xi_1,\ldots,\xi_m)$.

Згідно з теоремою Куна-Таккера задача (6.6) еквівалентна двоїстій задачі пошуку сідлової точки функції Лагранжа:

$$\begin{cases} L(w, w_0, \xi; \lambda, \eta) \to \min\limits_{w, w_0, \xi} \max\limits_{\lambda, \eta}; \\ \xi_i \geq 0, \lambda_i \geq 0, \eta_i \geq 0,\ i = 1,\ldots,m; \\ \lambda_i = 0 \text{ або } M_i(w, w_0) = 1 - \xi_i,\ i = 1,\ldots,m; \\ \eta_i = 0 \text{ або } \xi_i = 0, \quad i = 1,\ldots,m. \end{cases}$$

В останніх двох рядках записані умови доповнюючої нежорсткості. Необхідною умовою сідлової точки функції Лагранжа є те, що в цій точці її похідні мають дорівнювати нулю. Звідси випливають три корисних співвідношення:

$$\frac{\partial L}{\partial w} = w - \sum_{i=1}^{m} \lambda_i y_i x_i = 0 \quad \Rightarrow \quad w = \sum_{i=1}^{m} \lambda_i y_i x_i \tag{6.8}$$

$$\frac{\partial L}{\partial w_0} = -\sum_{i=1}^{m} \lambda_i y_i = 0 \quad \Rightarrow \quad \sum_{i=1}^{m} \lambda_i y_i = 0 \tag{6.9}$$

$$\frac{\partial L}{\partial \xi_i} = -\lambda_i - \eta_i + C = 0 \quad \Rightarrow \quad \eta_i + \lambda_i = C \quad i = 1,\ldots,m \tag{6.10}$$

З (6.8) випливає, що шуканий вектор ваг w є лінійною комбінацією векторів навчальної вибірки x_i, причому тільки тих, для яких $\lambda_i > 0$.

Визначення 3. Якщо $\lambda_i > 0$, то об'єкт навчальної вибірки x_i називається опорним вектором (support vector). З третього співвідношення (6.10) і нерівності $\eta_i > 0$ випливає, що $0 \leq \lambda_i \leq C$. Звідси та з умов доповнюючої нежорсткості випливає, що можливі тільки три допустимих поєднання значень змінних ξ_i, λ_i, η_i та відступів M_i .

Відповідно, всі об'єкти $x_i,\ i = 1,\ldots,m$ діляться на наступні три типи:

1. $\lambda_i = 0;\ \eta_i = C;\ M_i \geq 1.$

Об'єкт x_i класифікується правильно і не впливає на розв'язок w. Такі об'єкти будемо називати *периферійними або неінформативними*.

2. $0 < \lambda_i < C;\ 0 < \eta_i < C;\ \xi_i = 0;\ M_i = 1.$

Об'єкт x_i класифікується правильно і лежить точно на границі, яка поділяючої смуги. Такі об'єкти будемо називати *опорними граничними*.

3. $\lambda_i = C; \eta_i = 0; \xi_i > 0;\ M_i < 1$

Об'єкт x_i або лежить всередині поділяючої смуги, але класифікується правильно $\left(0 < \xi_i < 1,\ 0 < M_i < 1\right)$, або потрапляє на кордон класів $\left(\xi_i > 1,\ M_i < 0\right)$, або взагалі відноситься до чужого класу $\left(\xi_i > 1,\ M_i < 0\right)$. У всіх цих випадках об'єкт x_i будемо називати *опорним порушником*.

В силу співвідношення (6.10) в лагранжіані обнуляються всі члени, що містять змінні ξ_i і η_i, і він виражається тільки через двоїсті змінні λ_i.

$$\begin{cases} -L\left(\lambda\right) = -\sum\limits_{i=1}^{m}\lambda_i + \dfrac{1}{2}\sum\limits_{i=1}^{m}\sum\limits_{j=1}^{m}\lambda_i\lambda_j y_i y_j \left\langle x_i, x_j \right\rangle \to \min\limits_{\lambda}; \\ 0 \le \lambda_i \le C,\ i = 1,\ldots,m; \\ \sum\limits_{i=1}^{m}\lambda_i y_i = 0. \end{cases}$$

Тут мінімізується квадратичний функціонал, що має невід'ємну визначену квадратичну форму, отже, опуклий. Область, яка визначається обмеженнями нерівностями і одним рівнянням, також опукла. Отже, дана двоїста задача має єдине рішення. Припустимо, ми вирішили це завдання. Тоді вектор w обчислюється за формулою (6.8). Для визначення порогу w_0 достатньо взяти довільний опорний граничний вектор x_i і виразити w_0 з рівності $w_0 = \left\langle w, x_i \right\rangle - y_i$. На практиці для підвищення чисельної стійкості рекомендується брати медіану множини значень w_0, які обчислені по всіх граничних опорних векторах.

$$w_0 = med\left\{\left\langle w, x_i \right\rangle - y_i \,\middle|\, \lambda_i > 0, M_i = 1, i = 1,\ldots,m\right\} \qquad (6.11)$$

В результаті алгоритм класифікації має вигляд:

$$a\left(x\right) = sign\left(\sum_{i=1}^{m}\lambda_i y_i \left\langle x_i, x\right\rangle - w_0\right)$$

Звернемо увагу, що додавання йде не по всій вибірці, а тільки по опорних векторах, для яких $\lambda_i \neq 0$. Класифікатор $a\left(x\right)$ не зміниться, якщо всю решту об'єктів виключити з вибірки. Цю властивість називають *розрідженістю* (sparsity); саме вона і відрізняє SVM від інших лінійних класифікаторів – дискримінанта Фішера, логістичної регресії і одношарового персептрона. Ненульовими λ_i володіють не тільки граничні опорні об'єкти, але і об'єкти-порушники. Це говорить про недостатню робастності (стійкості

до шуму) SVM. Порушниками можуть бути об'єкти, що з'явилися в результаті помилкових спостережень; їх треба було б виключити з вибірки, а не будувати по них розв'язок.

Про підбір параметра регуляризації.

Константу C зазвичай вибирають по критерію ковзного контролю. Це трудомісткий спосіб, оскільки задачу доводиться розв'язувати знову при кожному значенні C. Добре те, що рішення, як правило, не дуже чутливе до вибору C, і занадто точна його оптимізація не потрібна. Якщо є підстави вважати, що вибірка майже лінійно роздільна, і лише об'єкти-викиди класифікуються невірно, то можна застосувати фільтрацію викидів. Спочатку завдання вирішується при деякому C, і з вибірки видаляється невелика частка об'єктів, що мають найбільшу величину помилки ξ_i . Після цього задачу вирішують знову з усіченою вибіркою. Можливо, доведеться виконати кілька таких ітерацій, поки ті об'єкти що залишилися, не виявляться лінійно роздільними.

6.4.3. Ядра і спрямляючі простори

Існує ще один підхід до розв'язку проблеми лінійної нероздільності. Це перехід від початкового простору ознакових описів об'єктів X до нового простору H за допомогою деякого перетворення $\psi : X \to H$. Якщо простір H має досить високу розмірність, то можна сподіватися, що в ньому вибірка виявиться лінійно роздільною (легко показати, що якщо вибірка X_m не суперечлива, то завжди найдеться простір розмірності не більше m , у якому вона буде лінійно роздільна). Простір H називають спрямляючим.

Якщо припустити, що ознаковими описами об'єктів є вектори $\psi(x_i)$, а не вектори x_i , то побудова SVM проводиться точно так само, як і раніше. Єдина відмінність полягає в тому, що скалярний добуток $\langle x, x' \rangle$ у просторі X усюди заміняється скалярним добутком $\langle \psi(x), \psi(x') \rangle$ у просторі H. Звідси випливає природня вимога: простір H повинен бути наділеним скалярним добутком, зокрема, підійде будь-який евклідовий, а в загальному випадку й гільбертовий, простір.

Визначення. Функцію $K : X \times X \to R$ називають ядром (kernel function), якщо її можна представити у вигляді $K(x, x') = \langle \psi(x), \psi(x') \rangle$ при деякому відображенні $\psi : X \to H$ де H – простір зі скалярним добутком.

Теорема Мерсера.

Важливо зрозуміти, чи довільна функція двох аргументів $K(x, x')$ може виконувати роль ядра? Наступна теорема дає вичерпну відповідь на це питання й показує, що клас допустимих ядер досить широкий.

Теорема 3 (Мерсер). Функція $K\left(x,x'\right)$ є ядром тоді й тільки тоді, коли вона симетрична, $K\left(x,x'\right)=K\left(x',x\right)$, невід'ємно визначена: $\int\limits_X\int\limits_X K\left(x,x'\right)g\left(x\right)g\left(x'\right)dxdx'\geq 0$ для будь-якої функції $g:X\to R$.

Невід'ємна визначеність може ще бути представленою таким чином:

Визначення 5. Функція $K\left(x,x'\right)$ невід'ємно визначена, якщо для будь-якої скінченної вибірки $X_p=\left(x_1,\ldots,x_p\right)$ з X матриця $K=\left\|K\left(x_i,y_j\right)\right\|$ розміру $p\times p$ невід'ємно визначена: $z^TKz\geq 0$ для будь-якого $z\in R^p$.

Перевірка невід'ємної визначеності функції в практичних ситуаціях може виявитися нетривіальною справою. Часто обмежуються перебором скінченного числа функцій, про які відомо, що вони є ядрами. Серед них вибирається краща, як правило, за критерієм ковзного контролю. Очевидно, що цей не оптимальний розв'язок. На сьогоднішній день проблема вибору ядра, оптимального для даної конкретної задачі, залишається відкритою.

Конструктивні способи побудови ядер

Наступні правила породження дозволяють будувати ядра в практичних задачах.

1. Довільний скалярний добуток $K\left(x,x'\right)=\left\langle x,x'\right\rangle$ є ядром.

2. Константа $K\left(x,x'\right)=1$ є ядром.

3. Добуток ядер $K\left(x,x'\right)=K_1\left(x,x'\right)K_2\left(x,x'\right)$ є ядром.

4. Для будь-якої функції $\psi:X\to R$ добуток $K\left(x,x'\right)=\psi\left(x\right)\psi\left(x'\right)$ – ядро.

5. Лінійна комбінація ядер з невід'ємними коефіцієнтами $K\left(x,x'\right)=\alpha_1K_1\left(x,x'\right)+\alpha_2K_2\left(x,x'\right)$ є ядром.

6. Композиція довільної функції $\varphi:X\to X$ і довільного ядра K_0 є ядром: $K\left(x,x'\right)=K_0\left(\varphi\left(x\right),\varphi\left(x'\right)\right)$.

7. Якщо $s:X\times X\to R$ – довільна симетрична інтегрована функція, то $K\left(x,x'\right)=\int\limits_X s\left(x,z\right)s\left(x',z\right)dz$ є ядром.

8. Функція виду $K\left(x,x'\right)=k\left(x-x'\right)$ є ядром тоді й тільки тоді, коли Фур'є-образ $F\left[k\right]\left(\omega\right)=\left(2\pi\right)^{\frac{n}{2}}\int\limits_X e^{-i\left\langle\omega,x\right\rangle}k\left(x\right)dx$ невід'ємний.

9. Ліміт рівномірно збіжної послідовності ядер – ядро.

10. Композиція довільного ядра K_0 і довільної функції $f : R \to R$, яку можна представити у вигляді збіжного степеневого ряду з невід'ємними коефіцієнтами $K\left(x,x'\right) = f\left(K_0\left(x,x'\right)\right)$, є ядром. Зокрема, функції $f\left(z\right) = e^z$ і $f\left(z\right) = \dfrac{1}{1-z}$ від ядра є ядрами.

Приклади ядер

Існує кілька «стандартних» ядер, які при ближчому розгляді приводять до вже відомих алгоритмів: до поліноміальних поділяючих поверхонь, до двошарових нейронних мереж, до потенційних функцій (RBF-мережі) та до інших. Таким чином, ядра претендують на роль універсальної мови для опису широкого класу алгоритмів навчання за прецедентами.

Спостерігається парадоксальна ситуація. З одного боку, ядра – одне з найкрасивіших винаходів у машиннім навчанні. З іншого боку, дотепер не знайдено ефективного загального підходу до їхнього добору в конкретних задачах.

Приклад 2. Візьмемо $X = R^2$ і розглянемо ядро $K\left(u,v\right) = \left\langle u,v\right\rangle^2$, де $u = \left(u_1,u_2\right)$,$v = \left(v_1,v_2\right)$. Спробуємо зрозуміти, який спрямляючий простір і перетворення ψ йому відповідають. Розкладемо квадрат скалярного добутку:

$$K\left(u,v\right) = \left\langle u,v\right\rangle^2 = \left\langle \left(u_1,u_2\right),\left(v_1,v_2\right)\right\rangle^2 =$$
$$\left(u_1v_1 + u_2v_2\right)^2 = u_1^2v_1^2 + u_2^2v_2^2 + 2u_1v_1u_2v_2 =$$
$$\left\langle \left(u_1^2,u_2^2,\sqrt{2}u_1u_2\right),\left(v_1^2,v_2^2,\sqrt{2}v_1v_2\right)\right\rangle$$

Ядро K представляється у вигляді скалярного добутку в просторі $H = R^3$. Перетворення $\psi : R^2 \to R^3$ має вигляд

$$\psi : \left(u_1,u_2\right) \mapsto \left(u_1^2,u_2^3,\sqrt{2}u_1u_2\right) .$$

Лінійній поверхні в просторі H відповідає квадратична поверхня у вихідному просторі X. Дане ядро дозволяє розділити внутрішню й зовнішню частину довільного еліпса, що неможливо у початковому двовимірному просторі.

Приклад 3. Ускладнимо ситуацію. Нехай тепер $X = R^n$,

$$K\left(u,v\right) = \left\langle u,v\right\rangle^d .$$

Тоді компонентами вектора $\psi\left(u\right)$ є різні добутки $\left(u_1\right)^{d_1}\ldots\left(u_n\right)^{d_n}$ при всіляких цілих невід'ємних $d_1,d_2,\ldots,d_n$, що задовольняють умові $d_1 + d_2 + \ldots. + d_n = d$. Число таких мономів, а отже й розмірність простору

H, дорівнює C_{n+d-1}^{d}. Простір H ізоморфний простору всіх поліномів, що складаються з мономів степеню d від змінних $u_1,...,u_n$.

Приклад 4. Якщо $X = R^n$,

$$K\left(u,v\right) = \left(\left\langle u,v\right\rangle + 1\right)^d ,$$

то H – простір усіх мономів степеню не вище d від змінних $u_1,...,u_n$. У цьому випадку простір H ізоморфний простору всіх поліномів степеню d. Лінійна роздільність множин у цьому просторі еквівалентна поліноміальній роздільності множин у початковому просторі X.

SVM як двошарова нейронна мережа

Розглянемо структуру алгоритму $a\left(x\right)$ після заміни в

$$a\left(x\right) = sign\left(\sum_{i=1}^{m}\lambda_i y_i \left\langle x_i, x\right\rangle - w_0\right),$$

$$w_0 = med\left\{\left\langle w, x_i\right\rangle - y_i \,\middle|\, \lambda_i > 0, M_i = 1, i = 1,...,m\right\}$$

скалярного добутку $\left\langle x_i, x\right\rangle$ ядром $K\left(x_i, x\right)$. Перенумеруємо об'єкти так, щоб перші h об'єктів виявилися опорними. Оскільки $\lambda_i = 0$ для всіх неопорних об'єктів, $i = h+1,...,m$, алгоритм $a\left(x\right)$ прийме вид

$$a\left(x\right) = sign\left(\sum_{i=1}^{h}\lambda_i y_i K\left(x_i, x\right) - w_0\right).$$

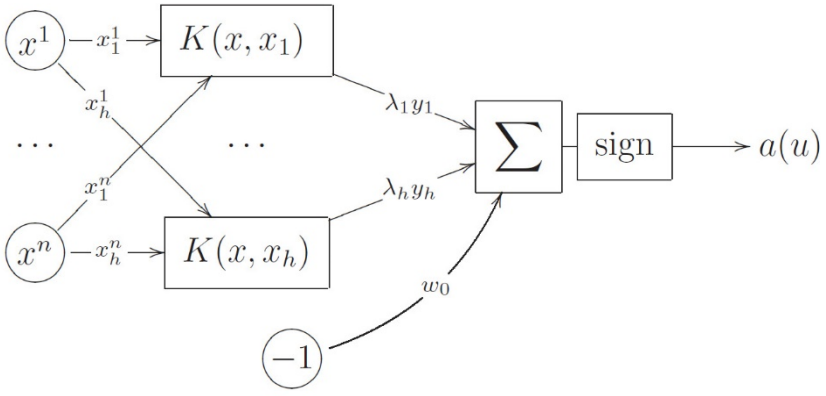


Рис. 6.6. Машина опорних векторів (SVM) як двошарова нейромережа.

Якщо $X = R^n$ то алгоритм $a\left(x\right)$ можна розглядати як суперпозицію, яку називають двошаровою нейронною мережею. Перший шар утворюють

ядра, другий шар – власне лінійний класифікатор. Така суперпозиція, побудована методом SVM, має кілька чудових особливостей.

По-перше, число нейронів першого шару визначається автоматично, тоді як у нейронних мережах визначення числа нейронів є окремою проблемою.

По-друге, прояснюється зміст двоїстих змінних: λ_i – це ступінь важливості ядра $K\left(x, x_i\right)$, фактично, важливості або «опорності» об'єкта x_i.

Приклад 5. Класичну нейронну мережа з сигмоїдними функціями активації одержимо, якщо в якості ядра взяти функцію

$$K\left(u, v\right) = th\left(k_0 + k_1 \left\langle u, v \right\rangle\right).$$

Дана функція задовольняє умовам Мерсера не при всіх значеннях параметрів k_0 і k_1. Зокрема, вона їм не задовольняє при $k_0 < 0$ або $k_1 < 0$. Однак це не перешкоджає її успішному практичному застосуванню. Замість гіперболічного тангенса $th\left(z\right)$ часто використовують також логістичну функцію $\sigma\left(z\right) = \dfrac{1}{1+e^{-z}}$.

Що поганого відбудеться, якщо функція $K\left(u, v\right)$ не буде задовольняти умовам Мерсера? Постановка задачі квадратичного програмування

$$\begin{cases} -L\left(\lambda\right) = -\sum\limits_{i=1}^{m} \lambda_i + \dfrac{1}{2} \sum\limits_{i=1}^{m} \sum\limits_{j=1}^{m} \lambda_i \lambda_j y_i y_j \left\langle x_i, x_j \right\rangle \to \min\limits_{\lambda}; \\ 0 \le \lambda_i \le C,\ i = 1, \ldots, m; \\ \sum\limits_{i=1}^{m} \lambda_i y_i = 0. \end{cases}$$

залишиться такою ж і в цьому випадку. Однак квадратична форма втратить властивість невід'ємної визначеності, а функціонал, який мінімізуємо, уже не буде опуклим, і розв'язок може виявитися не єдиним. Найнеприємніше те, що на границях гіперпаралелепіпеда $0 \le \lambda_i \le C$ виникне величезна кількість локальних мінімумів, і пошук розв'язку серед них у загальному випадку вимагає повного перебору. У цій ситуації багато методів квадратичного програмування будуть видавати якийсь локальний мінімум, зовсім не обов'язково гарний.

Приклад 6. Нейронну мережу з *радіальними базисними функціями* (radial basis functions, RBF) одержимо, якщо взяти гаусівське ядро

$$K\left(u, v\right) = \exp\left(-\beta \left\| u - v \right\|^2\right),$$

де β – параметр. Ядро $K\left(x_i, x\right)$ обчислює оцінку близькості об'єкта x до опорного об'єкта x_i. Чим ближче об'єкти, тим більше значення ядра.

Вихідний нейрон додає всі ці оцінки, та множить їх на коефіцієнти $\lambda_i y_i$. При цьому близькості до опорних об'єктів класу +1 додаються з додатними вагами, а до об'єктів класу –1 – з від'ємними. Вихідний нейрон робить голосування, порівнюючи сумарні близькості розпізнаваного об'єкта x до обом класів.

Переваги SVM

1. Задача квадратичного програмування має єдиний розв'язок, для знаходження якого розроблені досить ефективні методи.
2. Автоматично визначається складність суперпозиції – число нейронів першого шару, дорівнює числу опорних векторів.
3. Максимізація зазору між класами поліпшує узагальнюючу здатність.

Недоліки SVM

1. Нестійкість до шуму у початкових даних. Об'єкти-викиди є опорними й суттєво впливають на результат навчання.
2. Дотепер не розроблені загальні методи добору ядер під конкретну задачу. На практиці «цілком розумні» ядра, побудовані з урахуванням специфіки задачі, можуть і не мати властивість невід'ємної визначеності.
3. Добір параметра С вимагає багаторазового розв'язку задачі.

Запитання до розділу 6

1. Назвіть базові припущення для застосування методу логістичної регресії.
2. Переваги та недоліки логістичної регресії.
3. Що таке скоринг в методах логістичної регресії.
4. Яку вибірку називають лінійно роздільною?
5. Які переваги та недоліки методу SVM?

Розділ 7
Байєсівська теорія класифікації. Основні положення. Ймовірнісна постановка задачі. Непараметрична класифікація.

7.1. Основні положення теорії ймовірностей, які будуть використовуватися

7.1.1. Функція розподілу ймовірностей дискретної випадкової величини та її властивості

Визначення. Розглянемо функцію $F\left(x\right)$, визначену на всій числовій осі Ox

Для кожного x значення $F\left(x\right)$ дорівнює ймовірності того, що дискретна випадкова величина ξ приймає значення менші за x

Тобто $F\left(x\right)=P\left(\xi<x\right)$.

Ця функція називається ***функцією розподілу ймовірностей* (*the probability distribution function)***, або ***функцією розподілу.***

Приклад.

Випадкова величина ξ – число очок, що випали при однократному киданні грального кубика.

Ряд розподілу має вигляд:

ξ	1	2	3	4	5	6
$P\left(\xi\right)$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$	$1/6$

Знайти функцію розподілу $F\left(x\right)$. При $x<0$ функція $F\left(x\right)=0$, тому що ξ не приймає значень менше 1.

Якщо:$1\le x<2$, то $F\left(x\right)=p\left(\xi<2\right)=p\left(\xi=1\right)=\frac{1}{6};$

$2\le x<3$ то $F\left(x\right)=p\left(\xi<3\right)=p\left(\xi=1\right)+p\left(\xi=2\right)=\frac{1}{6}+\frac{1}{6}=\frac{1}{3};$

$3\le x<4$,то $F\left(x\right)=p\left(\xi<4\right)=p\left(\xi=1\right)+p\left(\xi=2\right)+p\left(\xi=3\right)=\frac{1}{6}+\frac{1}{6}+\frac{1}{6}=\frac{1}{2};$

Далі аналогічно :

$4\le x<5$ то $F\left(x\right)=\frac{1}{6}\cdot 4=\frac{4}{6}=\frac{2}{3}$;

$5\le x<6$ то $F\left(x\right)=\frac{1}{6}\cdot 5=\frac{5}{6};$

$6\le x$ то $F\left(x\right)=\frac{1}{6}\cdot 6=1;$

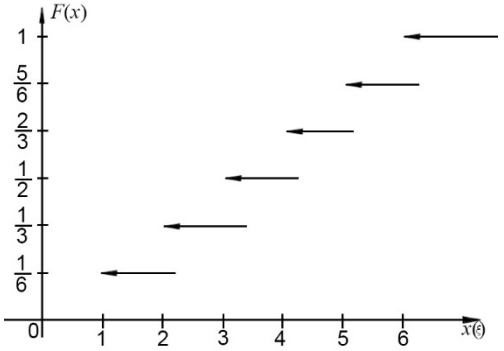


Рис. 7.1. Функція розподілу імовірності $F\left(x\right)$

Знаючи функцію розподілу $F\left(x\right)$ легко знайти імовірність того, що випадкова величина ξ задовольняє нерівності $x' \le \xi < x''$.

З теореми про суму подій і ймовірностей одержимо:

$P\left(\xi < x''\right) = P\left(\xi < x'\right) + P\left(x' \le \xi < x''\right)$. Звідси

$P\left(x' \le \xi < x''\right) = P\left(\xi < x''\right) - P\left(\xi < x'\right).$

За визначенням функції розподілу

$P\left(x' \le \xi < x''\right) = F\left(\xi < x''\right) - F\left(\xi < x'\right).$

Імовірність попадання дискретної випадкової величини в інтервал $x' \le \xi < x''$ дорівнює приросту (the growth) функції розподілу на цьому інтервалі.

Розглянемо **основні властивості функції розподілу.**

1. Функція розподілу є неспадною.

Нехай $x' < x''$, $P\left(x' \le x < x''\right) \ge 0$. Із знайденої формули отримаємо:

$F\left(x''\right) - F\left(x'\right) \ge 0 \Rightarrow F\left(x''\right) \ge F\left(x'\right)$

2. Значення функції задовольняють нерівність:

$0 \le F\left(x\right) \le 1,\ F\left(-\infty\right) = \lim\limits_{x \to -\infty} F\left(x\right) = 0$

$F\left(+\infty\right) = \lim\limits_{x \to +\infty} F\left(x\right) = 1$, це випливає з означення функції $F\left(x\right)$.

3. Імовірність того, що дискретна випадкова величина ξ приймає одне із можливих значень x , дорівнює стрибку функції розподілу в точці x

Цю властивість наочно видно з наведеного вище прикладу.

7.1.2. Неперервні випадкові величини

Крім дискретних випадкових величин, які приймають окремі числові значення і утворюють скінченну або нескінченну послідовність чисел, часто також зустрічаються випадкові величини, можливі значення яких заповнюють деякий інтервал.

Прикладом такої випадкової величини може бути відхилення від номіналу певного розміру деталі при правильно налагодженому процесі. Такі випадкові величини не можуть бути задані за допомогою закону розподілу

ймовірностей $P\left(x\right)$ не можна кожному значенню поставити у відповідність імовірність.

Функція розподілу $F\left(x\right)$ задається аналогічно: $F\left(x\right)=P\left(\xi<x\right)$

Функція $F\left(x\right)$ задана для всіх $x\in\left(-\infty,+\infty\right)$ і її значення в точці x дорівнює імовірності того, що випадкова величина матиме значення, менше x.

Всі рівності і властивості для функції розподілу справедливі і в цьому випадку. Доведення аналогічне випадку дискретної величини. Випадкова величина ξ називається ***неперервною*** **(*continuous*)**, якщо для неї існує невід'ємна, кусково-неперервна функція $f\left(x\right)$ така, що для всіх $x\in\left(-\infty,+\infty\right)$ виконується рівність:

$$F\left(x\right)=\int_{-\infty}^{+\infty}f\left(x\right)dx \qquad (7.1)$$

Функцію $f\left(t\right)$ називають ***густиною*** **(*щільністю*) (*the density*)** розподілу ймовірностей або коротко ***густиною розподілу*** **(*the density distribution*)**. Якщо $x_1<x_2$, то одержимо:

$$P\left(x_1\le\xi<x_2\right)=F\left(x_2\right)-F\left(x_1\right)=\int_{-\infty}^{x_2}f\left(x\right)dx-\int_{-\infty}^{x_1}f\left(x\right)dx=\int_{x_1}^{x_2}f\left(x\right)dx \quad (1)$$

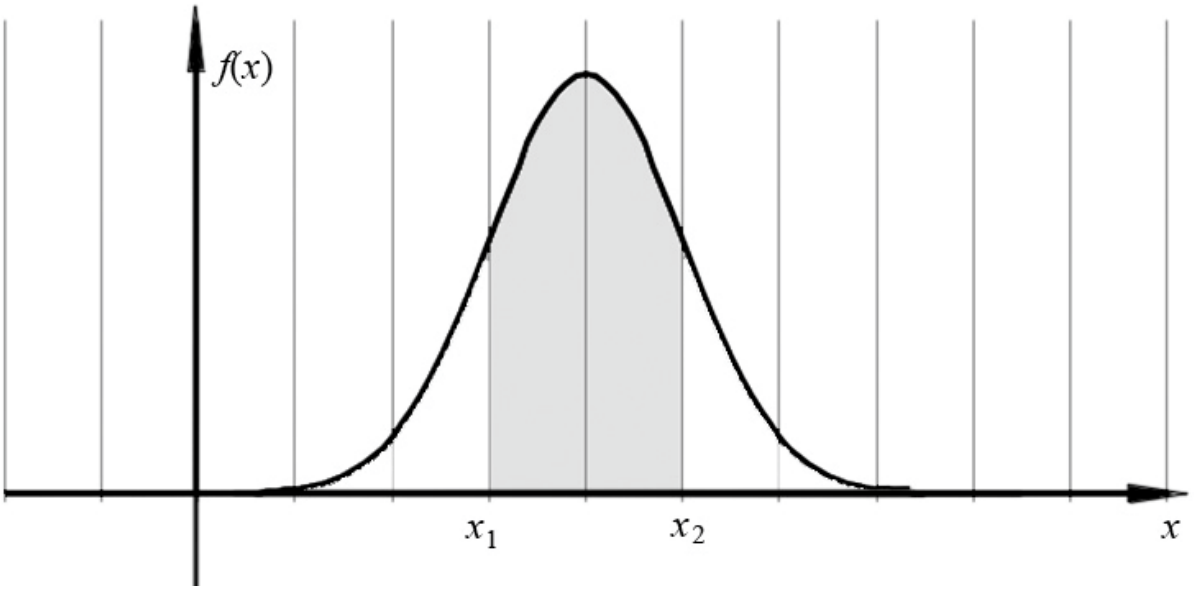


Рис. 7.2. Функція густини розподілу

Виходячи з геометричного змісту інтеграла як площі, можна сказати, що ймовірність виконання нерівності $x_1\le\xi<x_2$ рівна площі криволінійної трапеції з основою $\left[x_1,x_2\right)$, обмеженою зверху кривою $y=f\left(\alpha\right)$.

Далі $F\left(+\infty\right)=P\left(\xi<+\infty\right)=1$, тоді $F\left(+\infty\right)=\int\limits_{-\infty}^{+\infty}f\left(x\right)dx=1$.

Звідси слідує: $\lim\limits_{t\to+\infty}f\left(x\right)=\lim\limits_{t\to-\infty}f\left(x\right)=0$

Знайдемо $F\left(x\right)$ як похідну інтеграла за змінною верхньою границею, вважаючи $f\left(t\right)$ неперервною $F'\left(x\right)\frac{d}{dx}\int\limits_{-\infty}^{x}f\left(x\right)dx=f\left(x\right)$

Для неперервної випадкової величини функція $F\left(x\right)$ неперервна в довільній точці *x,* де функція $f\left(x\right)$ неперервна. Це випливає з того, що $F\left(x\right)$ в цих точках диференційовна. Часто $F\left(x\right)$ називають ***інтегральною функцією розподілу (integral distribution function)***, а $f\left(x\right)$ ***диференціальною функцією розподілу (the differential distribution functiony).***

Із формули (1), приймаючи $x_1=x,\ x_2=x+\Delta x$ і в силу неперервності $F\left(x\right)$ маємо:

$$\lim_{x\to0}P\left(x\le\xi<x+\Delta x\right)-\lim_{x\to0}\Delta F\left(x\right)=P\left(\xi=x\right)=0$$

Таким чином, імовірність того, що неперервна випадкова величина може прийняти окреме числове значення *x*, дорівнює нулю:

$$P\left(\xi=x_1\right)=P\left(\xi=x_2\right)=0$$

Отже,

$$P\left(x_1\le\xi<x_2\right)=P\left(\xi=x_1\right)+P\left(x_1<\xi<x_2\right)=P\left(x_1<\xi<x_2\right)=P\left(x_1\le\xi<x\right.$$

Умовна імовірність

Імовірність події *A*, визначена за умови, що подія B відбулася, називається ***умовною*** і позначається $P\left(A\ /\ B\right)$ або $P_B\left(A\right)$

Значення умовної імовірності визначається з виразу:

$$P\left(A\ /\ B\right)=\frac{P\left(A\cap B\right)}{P\left(B\right)},\ P\left(B\right)>0 \qquad (7.2)$$

Приклад 1. У регіоні є 50 фірм і тільки 10 із них користуються кредитами банку. Зі списку фірм навмання вибирають дві підряд. Нехай подія *A* – друга фірма не користується кредитами банків, подія *B* – перша фірма користується кредитами банків. Обчислити ймовірності $P\left(A\right)$, $P\left(B\right)$, $P\left(A\cap B\right)$, $P\left(A\ /\ B\right)$

Розв'язання. У даній задачі простір Ω складається з $50\cdot49=2450$ елементів, з яких$49\cdot40=1960$ елементарних результатів сприяють події *A*,

$10 \cdot 49 = 490$ – сприяють події B і $10 \cdot 40 = 400$ – сприяють події $A \cap B$. Користуючись класичним означенням імовірності випадкової події, отримуємо:

$$P\left(B\right) = \frac{490}{2450} 0.2, \;\; P\left(A\right) = \frac{1960}{2450} = 0.8 \;\; P\left(A \cap B\right) = \frac{400}{2450} = \frac{8}{49} = 0.163$$

Коли відомо, що подія B відбулася, то, замість 2450 можливих елементарних результатів, може відбутися лише один із $10 \cdot 49 = 490$

результатів, з яких $10 \cdot 40 = 400$ сприяють події A. Тому

$$P\left(A \,/\, B\right) = \frac{400}{490} = 0.816$$

Відзначимо ще один факт, який використовується для обчислення умовної ймовірності:

$$P\left(A \,/\, B\right) = \frac{400}{490} = \frac{400/2450}{490/2450} = \frac{P\left(A \cap B\right)}{P\left(B\right)}$$

Нехай $\Omega -$ простір елементарних подій, і $A \subset \Omega$, $B \subset \Omega$. Тоді з (7.2) випливають такі властивості умовної ймовірності:

1) $P\left(A \,/\, B\right) \geq 0$

2) $P\left(\Omega \,/\, B\right) = 1$

3) $P\left(B \,/\, B\right) = 1$

З (7.2) також випливає правило множення імовірностей:

$$P\left(A \cap B\right) = P\left(B\right) \cdot P\left(A \,/\, B\right) = P\left(A\right) \cdot P\left(B \,/\, A\right), \qquad (7.3)$$

тобто ймовірність добутку двох подій дорівнює добуткові ймовірності однієї з них на умовну ймовірність другої за умови, що перша подія відбулася. Практичне значення рівності (3) полягає в тому, що вона дозволяє визначити ймовірність складеної події $A \cap B$ за відомими ймовірностями подій-співмножників, обминаючи побудову для події $A \cap B$ простору елементарних подій Ω .

Нехай, наприклад, потрібно визначити ймовірність добутку подій A і B у наведеному вище прикладі 1. У цьому випадку $P\left(B\right) = \frac{10}{50} = 0.2$

Перед другою спробою для вибору залишається тільки 49 фірм. Якщо перша вибрана фірма користується кредитами банків (подія B відбулася), то

$P\left(A \,/\, B\right) = \frac{40}{49}$. Застосовуючи формулу (3), отримаємо, що

$$P\left(A \cap B\right) = \frac{1}{5} \cdot \frac{40}{49} = \frac{8}{49} = 0.163$$

7.1.3. Повна імовірність

Нехай в умовах експерименту подія A з'являється спільно з однією з групи несумісних подій H_i, де $i = 1,2,...n$. Згадані події H_i, утворюють повну групу подій, тобто $\sum_{i=1}^{n} P\left(H_i\right) = 1$.

Нехай відомі всі апріорні імовірності $P\left(H_i\right)$. Нехай також відомі всі умовні імовірності $P\left(A / H_i\right)$. Тоді ймовірність події A визначається за формулою повної ймовірності:

$$P\left(A\right) = \sum_{i=1}^{n} P\left(H_i\right) P\left(A / H_i\right)$$

Наведена формула називається **формулою повної ймовірності.**

Приклад 2. В магазині три холодильники, в яких закінчується морозиво. У першому - 4 білих морозива і 6 шоколадних, у другому - 2 білих і 8 шоколадних, у третьому - 3 білих і 7 шоколадних. Навмання вибирають холодильник і виймають з нього морозиво, визначити ймовірність того, що воно біле.

Розв'язування. Позначимо події наступним чином: H_i – вибрано і-й холодильник, А – вибрано біле морозиво. Тоді ймовірності витягнути морозиво з кожного холодильника однакові і дорівнюють 1/3:

$$P\left(H_1\right) = P\left(H_2\right) = P\left(H_3\right) = \frac{1}{3}$$

Ймовірності витягнути біле морозиво будуть рівні кількості морозива в кожному з холодильників розділеній на загальну кількість морозива

$$P\left(A / H_1\right) = \frac{4}{4+6} = \frac{4}{10} = 0.4$$

$$P\left(A / H_2\right) = \frac{2}{2+8} = \frac{2}{10} = 0.2$$

$$P\left(A / H_3\right) = \frac{3}{3+7} = \frac{3}{10} = 0.3$$

Використовуючи формулу повної ймовірності, знаходимо

$$P\left(A\right) = \frac{1}{3} \cdot 0.4 + \frac{1}{3} \cdot 0.2 + \frac{1}{3} \cdot 0.3 = 0.3$$

7.1.4. Формула Байєса

Нехай події H_i $(i = 1,2,\ldots,n)$ утворюють повну групу несумісних подій, тобто $\sum_{i=1}^{n} P\left(H_i\right) = 1$ і нехай подія A відбувається обов'язково з однією з подій H_i.

Із деяких міркувань відомі ймовірності подій $P\left(H_i\right),\ i-1,2,\ldots,n$.

Відомими вважаються також умовні ймовірності $P\left(A\ /\ H_i\right),\ i=1,2,\ldots,n$

Далі припустимо, що проведено стохастичний експеримент, у результаті якого відбулася подія *A*. Тоді виникає питання про переоцінку ймовірностей подій з позиції їх сприяння появі події *A*, тобто обчислення ймовірностей $P\left(H_i\ /\ A\right),\ i=1,2,\ldots,n$.

Теорема. *Нехай набір подій* $H_1,H_2,\ldots,H_n$ *утворює повну групу попарно несумісних подій, до того ж* $P\left(H_i\right)>0$ *для кожного i* = 1, 2, ..., *n. Тоді для*

будь-якої випадкової події A такої, що P(*A*)>0*, виконується рівність:*

$$P\left(H_i\ /\ A\right)=\frac{P\left(H_i\right)P\left(A\ /\ H_i\right)}{\sum_{k=1}^{n}P\left(H_k\right)P\left(A\ /\ H_k\right)} \qquad (7.4)$$

Доведення.

Для доведення (4) використаємо формулу множення і формулу повної ймовірності. Отримаємо:

$$P\left(H_i\ /\ A\right)=\frac{P\left(H_i\cap A\right)}{P\left(A\right)}=\frac{P\left(H_i\right)P\left(A\ /\ H_i\right)}{\sum_{k=1}^{n}P\left(H_k\right)P\left(A\ /\ H_k\right)}$$

Формулу (4) називають *формулою Байєса. Вона визначає імовірність подій* H_i *за умови, що подія A відбулася.*

Приклад 3. Деталі, які виготовляє завод, можуть бути перевірені щодо їх стандартності одним із двох контролерів. Імовірність того, що деталь буде перевірена першим контролером, становить 0,6, другим контролером – 0,4. Імовірність того, що деталь буде визнана стандартною першим контролером, – 0,94, другим контролером – 0,98. Після перевірки деталь визнана стандартною (подія *A*). Знайти ймовірність того, що деталь перевірив перший контролер.

Розв'язання. До проведення експерименту (перевірки деталі) можливі дві події: H_1 або H_2 – деталь надійде на перевірку до 1-го або 2 контролера, відповідно. Імовірності подій: $P\left(H_1\right)=0.6$; $P\left(H_2\right)=0.4$. Умовні ймовірності події *A:* $P\left(A\ /\ H_1\right)=0.94$; $P\left(A\ /\ H_2\right)=0.98$. Шукану ймовірність $P\left(H_1\ /\ A\right)$ обчислимо за формулою (7.4):

$$P\left(H_1 / A\right) = \frac{P\left(H_1\right)P\left(A / H_1\right)}{P\left(H_1\right)P\left(A / H_1\right) + P\left(H_2\right)P\left(A / H_2\right)} = \frac{0.6 \cdot 0.94}{0.6 \cdot 0.94 + 0.4 \cdot 0.98} = 0.59$$

Отже, імовірність того, що деталь, яка була визнана стандартною, перевірив перший контролер, становить 0.59.

7.1.5. Імовірнісна постановка задачі класифікації

Нехай $X -$ множина об'єктів, $Y -$ скінченна множина імен класів, множина $X \times Y -$ є імовірнісним простором з щільністю розподілу $p\left(x, y\right) = P\left(y\right) p\left(x / y\right)$ Ймовірності появи об'єктів, що належать кожному з класів: $P_y = P\left(y\right)$ називають апріорними ймовірностями класів. Щільності розподілу $p_y\left(x\right) = p\left(x / y\right)$ називають функціями правдоподібності класів. Імовірнісна постановка задачі класифікації поділяється на дві незалежні підзадачі.

Задача1. Нехай існує проста вибірка $X_m = \left(x_i, y_i\right)_{i=1}^m$ з невідомого розподілу $p\left(x, y\right) = P_y p_y\left(x\right)$. Потрібно побудувати емпіричні оцінки апріорних ймовірностей $\widehat{P}_y$ і функцій правдоподібності $\widehat{p}_y\left(x\right)$ для кожного з класів $y \in Y$.

Задача 2. За відомими щільностями розподілу $p_y\left(x\right)$ і апріорними імовірностями P_y всіх класів $y \in Y$ побудувати алгоритм $a\left(x\right)$, що мінімізує ймовірність помилкової класифікації.

Друга задача вирішується відносно легко, і ми можемо її вирішити зараз. Перша задача має безліч рішень, оскільки багато розподілів $p\left(x, y\right)$ могли б породити одну і ту ж вибірку X_m. Доводиться залучати різні припущення про щільність, що і призводить до великої різноманітності байесовських методів.

Функціонал середнього ризику

Знання функцій правдоподібності дозволяє знаходити ймовірності подій виду

«$x \in \Omega$ за умови, що x належить класу y»:

$$P\left(\Omega / y\right) = \int_{\Omega} p_y\left(x\right) dx,\ \Omega \subseteq X$$

Розглянемо довільний алгоритм $a : X \to Y$. Він розбиває множину X на області, які не перетинаються $A_y = \left\{x \in X \middle| a\left(x\right) = y\right\},\ y \in Y$. Імовірність

того, що з'явиться об'єкт класу y і алгоритм a віднесе його до класу s, дорівнює $P_y P\left(A_s / y\right)$.

Кожній парі $\left(y,s\right) \in Y \times Y$ поставимо у відповідність величину втрати λ_{ys} при віднесенні об'єкта класу y до класу S. Зазвичай вважають $\lambda_{yy} = 0$, $\lambda_{y,s} > 0$ при $y \neq s$.Співвідношення втрат на різних класах, як правило, відомі заздалегідь.

Примітка. Великою буквою P будемо позначати ймовірності, а малою p щільності розподілу.

Символами з кришечкою (наприклад, $\widehat{P}$) позначатимем вибіркові (емпіричні) оцінки імовірностей, функцій розподілу або випадкових величин, що обчислюються за вибіркою.

Визначення. Функціоналом середнього ризику називають очікувану величину втрати при класифікації об'єктів алгоритмом a:

$$R\left(a\right) = \sum_{y \in Y} \sum_{s \in Y} \lambda_{ys} P_y P\left(A_s / y\right)$$

Якщо величина втрат однакова для помилок будь-якого роду, $\lambda_{ys} = \left[y \neq s\right]$, то середній ризик $R\left(a\right)$ збігається з ймовірністю помилки алгоритму a.

Аргументи максимізації та мінімізації

Аргумент максимізації (argmax або arg max) – значення аргументу, при якому даний вираз сягає максимуму. Іншими словами, $\arg\max_x f\left(x\right)$ – є значення x , при якому $f\left(x\right)$ досягає свого максимального значення.

Є розв'язком задачі максимізації функції зі скінченною кількістю аргументів

$$\arg\max_x f\left(x\right) \in \left\{x \middle| \forall y : f\left(y\right) \leq f\left(x\right)\right\}.$$

Приклад.

$\arg\max_{x \in R}\left(x\left(10 - x\right)\right) = 5$ оскільки максимум функції $\cos x$ дорівнює 1 при $x = 0, 2\pi, 4\pi$

$\arg\max_{x \in [0,4\pi]} \cos\left(x\right) \in \left\{0, 2\pi, 4\pi\right\}$ оскільки максимум функції

дорівнює 25 при $x = 5$

7.1.6. Оптимальний класифікатор Байєса

Теорема. Якщо відомі апріорні ймовірності $P\left(y\right)$ і функції правдоподібності $p\left(x / y\right)$, то мінімум середнього ризику R (a) досягається алгоритмом:

$$a\left(x\right) = \arg\min_{s\in Y} \sum_{y\in Y} \lambda_{ys} P\left(y\right) p\left(x / y\right) \qquad (7.5)$$

Часто можна вважати, що величина втрати залежить тільки від істинної класифікації об'єкта, але не від того, до якого класу він був помилково віднесений. В цьому випадку формула оптимального алгоритму спрощується:

Теорема. Якщо $P\left(y\right)$ і $p\left(x / y\right)$ відомі, і до того ж $\lambda_{yy} = 0$, $\lambda_{ys} = \lambda_y$ для всіх $y, s \in Y$, то мінімум середнього ризику $R\left(a\right)$ досягається алгоритмом:

$$a\left(x\right) = \arg\max_{y\in Y} \lambda_y P\left(y\right) p\left(x / y\right) \qquad (7.6)$$

Поверхня розділення

Поверхня розділення між класами s і t – це геометричне місце точок $x \in X$ таких, що максимум в (7.6) досягається одночасно при $y = s$ і $y = t$:

$$\lambda_t P\left(t\right) p\left(x / t\right) = \lambda_s P\left(s\right) p\left(x / s\right)$$

Об'єкти x, що задовольняють цьому рівнянню, можна відносити до будь-якого з двох класів s ,t , що не вплине на середній ризик $R\left(a\right)$.

Апостеріорна ймовірність

Апостеріорна ймовірність класу y для об'єкта x. Це умовна ймовірність $P\left(y / x\right)$. Вона може бути обчислена за формулою Байеса, якщо відомі $P\left(y\right)$ і $p\left(x / y\right)$

$$P\left(y / x\right) = \frac{p\left(x, y\right)}{p\left(x\right)} = \frac{P\left(y\right) p\left(x / y\right)}{\sum_{s\in Y} P\left(s\right) p\left(x / s\right)}$$

У багатьох додатках важливо не тільки класифікувати об'єкт x, але і сказати, з якою ймовірністю $P\left(y / x\right)$ він належить кожному з класів. Через апостеріорні ймовірності виражається величина очікуваних втрат на об'єкті x

$$R\left(x\right) = \sum_{y\in Y} \lambda_y P\left(y / x\right)$$

Принцип максимуму апостеріорної ймовірності.

Оптимальний алгоритм класифікації (7.6) можна переписати через апостеріорні ймовірності:

$$a\left(x\right) = \arg\max_{y\in Y} \lambda_y P\left(y / x\right)$$

Тому вираз (6) називають Байєсовим правилом розв'язування.

Мінімальне значення середнього ризику $R(a)$, що досягається Байєсовим правилом розв'язування, називається Байєсовим ризиком або Байєсовим рівнем помилки. Якщо класи рівнозначні ($\lambda_y \equiv 1$), то Байєсве правило називають також принципом максимуму апостеріорної ймовірності. Якщо класи ще й рівноімовірні ($P(y) \equiv \frac{1}{|Y|}$), То об'єкт x просто відноситься до класу y з найбільшим значенням щільності розподілу $p(x/y)$ в точці x.

Завдання відновлення щільності розподілу

Перейдемо до задачі 1. Потрібно оцінити, якою могла б бути щільність імовірнісного розподілу $p(x,y) = P(y)\,p(x/y)$ при генерації вибірки X_m.

Позначимо підвибірку елементів класу y через

$$X_m(y) = \left\{ (x_i, y_i)_{i=1}^m \,\middle|\, y_i = y \right\}$$

Найпростіше оцінити апріорні ймовірності класів $P(y)$. Відповідно до закону великих чисел, частота появи об'єктів кожного з класів

$$\widehat{P}(y) = \frac{m(y)}{m},\ m(y) = \left|X_m(y)\right|,\ \ y \in Y \qquad (7.7)$$

сходиться по ймовірності до $P(y)$ при $m(y) \to \infty$. Чим більше довжина вибірки, тим точніше вибіркова оцінка $\widehat{P}(y)$. Оцінка (7.7) є незміщеною лише в тому випадку, якщо всі без винятку об'єкти спостереження заносилися в навчальну вибірку. На практиці застосовуються і інші принципи формування даних. Наприклад, в задачах з незбалансованими класами (unbalanced classes) один з класів може зустрічатися в тисячі разів рідше за інших; це може ускладнювати побудову алгорітмов, тому вибірку формують невипадковим чином, щоб об'єкти всіх класів були представлені порівну. Можлива також ситуація, коли навчальна вибірка формується в ході планованого експерименту, а застосовувати побудований алгоритм класифікації передбачається в реальному середовищі з іншими апріорними можливостями класів. У всіх подібних ситуаціях оцінка $P(y)$ повинна робитися не по частці навчальних об'єктів (7.6), а з інших змістовних міркувань.

Задача відновлення щільності має самостійне значення, тому ми сформулюємо її в більш загальному вигляді, позначаючи вибірку через X_k замість $X_m(y)$, що дозволить дещо спростити позначення.

Задача 3. Задано множину об'єктів $X_k = \{x_1, x_2, ..., x_k\}$, які обрані випадково і незалежно згідно невідомому розподілу $p(x)$. Потрібно побудувати емпіричну оцінку щільності – функцію $\hat{p}(x)$, що наближає $p(x)$ на всьому X.

«Наївний» байесовський класифікатор

Припустимо, що об'єкти $x \in X$ описуються n числовими ознаками $f_j : X \to R,\ j = 1,...,n$. Позначимо через $x = (\xi_1, \xi_2, ..., \xi_n)$ довільний елемент простору об'єктів $X \subset R^n$, де $\xi_j = f_j(x)$.

Гіпотеза 1. Ознаки $f_1(x), ..., f_n(x)$ є незалежними випадковими величинами. Отже, функції правдоподібності класів представимо у вигляді

$$p(x / y) = p(\xi_1 / y) \cdot p(\xi_1 / y) \cdot ... \cdot p(\xi_1 / y),\ y \in Y \qquad (7.8)$$

де $p(\xi_j / y)$ - щільність розподілу значень j -ї ознаки для класу y, тобто це міра правильного розпізнавання класу y за ознакою ξ_i

Тоді імовірність того, що об'єкт x належить класу y:

$$P(y / x) = P(y) \prod_{i=1}^{n} p(\xi_i / y),$$

де $P(y)$ – апріорна імовірність того що об'єкт належить класу y.

Мета такої класифікації – знайти найкращий клас y для об'єкта x. У даному методі найкращим є найбільш імовірний клас

$$a(x) = \arg\max_{y \in Y} \hat{P}(y / x) = \arg\max_{y \in Y} \hat{P}(y) \prod_{i=1}^{n} \hat{p}(\xi_1 / y) \qquad (7.9)$$

Припущення про незалежність істотно спрощує завдання, тому що оцінити n одновимірних щільностей набагато простіше, ніж одну n -вимірну щільність. Але воно вкрай рідко виконується на практиці, тому алгоритми класифікації, які використовують (8), називаються наївними байесовськими.

Примітка. Ми пишемо $\hat{P}$, $\hat{p}$, а не P, p , тому що не знаємо справжніх параметрів $P(y)$ і $p(\xi_j / y)$, а можемо лише оцінити їх за допомогою навчальних множин.

У рівності (7.9) перемножуються кілька умовних імовірностей, по одній для кожного значення $\xi_j, 1 \le j \le n$. Це може призвести до переповнення машинної пам'яті. Отже, краще замінити добуток імовірностей додаванням їх логарифмів. Клас з найбільшим значенням логарифма імовірності залишається найбільш ймовірним, тому що

$\ln ab = \ln a + \ln b$ і логарифмічна функція монотонна. Отже, у наївному методі Байеса насправді потрібно знайти точку максимуму наступної функції.

$$a\left(x\right) = \arg\max_{y \in Y} \left(\ln \lambda_y \widehat{P}\left(y\right) + \sum_{j=1}^{n} \ln \widehat{p}\left(\xi_j \ / \ y\right) \right) \qquad (7.10)$$

Рівність (9) допускає просту інтерпретацію. Кожен логарифм умовної імовірності $\ln \widehat{P}\left(\xi_j \ / \ y\right)$ – це вага, що вказує, наскільки важлива ознака ξ_j для класу y. Аналогично, апріорна імовірність $\ln \widehat{P}\left(y\right)$ – це вага, що характеризує відносну частоту класу y. Ті класи, що зустрічаються більш часто, частіше є правильними, ніж рідкісні. Таким чином, ця сума логарифмів імовірностей і ваг ознак характеризує кількість свідчень того, що об'єкт належить класу, а рівність ідентифікує клас, якому відповідає найбільша кількість доказів.

Основні його переваги – простота реалізації і низькі обчислювальні витрати при навчанні та класифікації. У тих рідкісних випадках, коли ознаки (майже) незалежні, наївний байесовський класифікатор (майже) оптимальний.

Основний його недолік – низька якість класифікації. Він використовується або як еталон при експериментальному порівнянні алгоритмів, або як елементарний «будівельний» блок в алгоритмічних композиціях.

7.1.7. Непараметрична класифікація

Непараметричні методи класифікації засновані на локальному оцінюваанні щільності розподілу класів $p\left(x \ / \ y\right)$ в околі об'єкта $x \in X$, який підлягає класифікації. Для класифікації об'єкта x застосовується основна формула (7.6).

Непараметричні оцінки щільності

Локальне оцінювання спирається на саме визначення щільності. Почнемо з найпростіших одновимірних оцінок. Вони вже можуть виявитися корисними на практиці, зокрема, при побудові «наївних» байесовських класифікаторів (7.9). Крім того, вони підкажуть нам ідеї узагальнення на багатовимірний випадок.

Дискретний випадок.

Нехай X-скінченна множина, і $\left|X\right| << k$. Оцінкою щільності служить гістограма значень x_i, які зустрілися в вибірці $X_k = \left(x_i\right)_{i=1}^{k}$:

$$\widehat{p}\left(x\right) = \frac{1}{k} \sum_{i=1}^{k} \left[x_i = x\right] \qquad (7.10)$$

Одновимірний неперервний випадок.

Нехай $X = R$ Згідно з визначенням щільності

$p(x) = \lim\limits_{h \to 0} \frac{1}{2h} P[x-h, x+h]$, де $P[a,b]$ - імовірнісна міра відрізка [a, b].

Відповідно, емпірична оцінка щільності визначається як частина точок вибірки, що лежать всередині відрізка$[x-h, x+h]$, де $h-$ невід'ємний параметр, який називають шириною вікна:

$$\hat{p}(x/h) = \frac{1}{2kh} \sum_{i=1}^{k} [|x - x_i| < h] \qquad (7.11)$$

Функція $\hat{p}(x/h)$ є кусочно-постійною, що призводить до появи широких зон невпевненості, в яких максимум (7.6) досягається одночасно для декількох класів $y \in Y$. Проблема вирішується за допомогою локальної непараметричної оцінки Парзена-Розенблатта:

$$\hat{p}(x/h) = \frac{1}{dh} \sum_{i=1}^{d} K\left(\frac{x - x_i}{h}\right) \qquad (7.12)$$

де $K(z) -$ функція, яка називається ядром, парна і нормована $\int K(z)dz = 1$

Функція $\hat{p}(x/h)$ має той же ступінь гладкості, що і ядро $K(z)$, і, завдяки нормуванню, дійсно може інтерпретуватися як щільність імовірності: $\int \hat{p}(x/h)dx = 1$ за довільного h.

На практиці часто використовуються ядра, що показані на рис. 7.3.

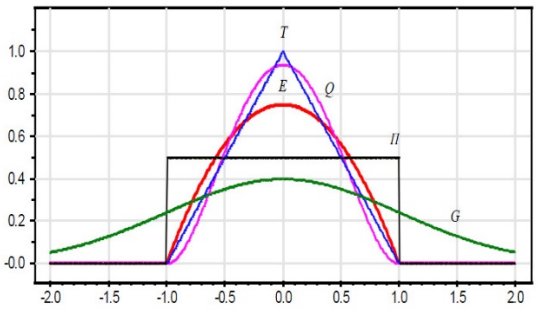


Рис. 7.3. Ядра, що застосовуються на практиці:

E-експоненціальне; Q-квадратичне; T-трикутне; G-гаусівське;П-прямокутне.

Прямокутне ядро $K(z) = \frac{1}{2}[|z| < 1]$ відповідає найпростішій оцінці (7.11). Точкове ядро $K(z) = [z = 0]$ при h = 1 відповідає дискретному випадку (7.10).

Багатовимірний неперервний випадок.

Нехай об'єкти описуються n числовими ознаками $f_j : X \to R,\ j = 1,\ldots,n$ Тоді непараметрична оцінка щільності в точці $x \in X$ записується в наступному вигляді:

$$\hat{p}\left(x / h\right) = \frac{1}{d}\sum_{i=1}^{d}\prod_{j=1}^{n}\frac{1}{h_j}K\left(\frac{f_j\left(x\right) - f_j\left(x_i\right)}{h_j}\right) \qquad (7.13)$$

Таким чином, в кожній точці x_i багатовимірна щільність представляється у вигляді добутку одновимірних щільностей. Зауважимо, що це ніяк не пов'язано з «наівним» Байєсовим припущенням про незалежність ознак. При «наївному» підході щільність представлялася б як добуток одновимірних парзеновскіх оцінок (7.12), тобто як добуток сум, а не як сума добутків.

7.1.8. Метод парзенівського вікна

Запишемо парзенівську оцінку щільності (13) для кожного класу $y \in Y$:

$$\hat{p}\left(x / y, h\right) = \frac{1}{m_y V\left(h\right)}\sum_{i=1}^{m}\left[y_i = y\right]K\left(\frac{\rho\left(x, x_i\right)}{h}\right), \qquad (7.14)$$

де $K -$ядро, $h -$ ширина вікна. Якщо нормуючий множник $V\left(h\right)$ не залежить від y, то в класифікаторі Байєса (6) його можна прибрати з-під знака argmax і взагалі не обчислювати. Підставами оцінку щільності (14) і оцінку апріорної ймовірності класів $\hat{P}\left(y\right) = \frac{m\left(y\right)}{m}$ в формулу (6):

$$a\left(x; X_m, h\right) = \arg\max_{y \in Y}\lambda_y\sum_{i=1}^{m}\left[y_i = y\right]K\left(\frac{\rho\left(x, x_i\right)}{h}\right) \qquad (7.15)$$

Вибірка X_m зберігається «як є» і грає роль параметра алгоритму. Якщо метрика ρ фіксована, то навчання парзеновского класифікатора (7.15) зводиться до підбору ширини вікна h і виду ядра K.

Ширина вікна h істотно впливає на якість відновлення щільності. При h $\to$ 0 щільність концентрується поблизу навчальних об'єктів, і функція $\hat{p}\left(x / h\right)$ має різкі скачки. При h $\to \infty$ щільність надмірно згладжується і вирджується в константу. Отже, має існувати компромісне значення ширини вікна h^*. На практиці його знаходять використовуючи Leave-one-out cross-validation

$$LOOV\left(h, X_m\right) = \sum_{i=1}^{m}\left[a\left(x_i; X_m \setminus x_{i,h}\right) \neq y_i\right] \to \min$$

де $a\left(x; X_m \setminus x_i, h\right)$ алгоритм класифікації, побудований за навчальною вибіркою X_m без об'єкта x_i. Зазвичай залежність $LOOV\left(h\right)$ має характерний мінімум, відповідний оптимальній ширині вікна h *.

Запитання до розділу 7

1. Дайте визначення функції розподілу ймовірностей дискретної випадкової величини та її властивості.

2. Запишіть вираз для визначення умовної імовірності.

3. За якою формулою обчислюється повна ймовірність?

4. Сформулюйте теорему та наведіть формулу Байєса.

5. Сформулюйте теорему про оптимальний класифікатор Байєса.

Розділ 8
Нормальний дискримінантний аналіз

У параметричному підході передбачається, що щільність розподілу вибірки $X_m = \{x_1, x_2, ..., x_m\}$ відома з точністю до параметра, $p(x) = \varphi(x;\theta)$, де φ – фіксована функція. Вектор параметрів θ оцінюється по вибірці X_m за допомогою *принципу максимуму правдоподібності* (maximum likelihood).

Нормальний дискримінантний аналіз – це спеціальний випадок байєсовської класифікації, коли передбачається, що щільності всіх класів $p_y(x)$, $y \in Y$ є багатовимірними нормальними. Цей випадок цікавий і зручний тим, що задача оцінювання параметрів розподілу за вибіркою вирішується аналітично.

8.1. Багатовимірний нормальний розподіл

Нехай $X = \mathbb{R}^n$, тобто об'єкти описуються n числовими ознаками.

Визначення. Імовірнісний розподіл з щільністю

$$\aleph(x;\mu,\Sigma) = (2\pi)^{-\frac{n}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(x-\mu)^{\mathrm{T}} \Sigma^{-1}(x-\mu)\right), x \in \mathbb{R}^n,$$

називають n-вимірним нормальним (гаусовським) розподілом з математичним очікуванням (центром) $\mu \in \mathbb{R}^n$ *і коваріаційною матрицею* $\Sigma \in \mathbb{R}^{n\times n}$. *Передбачається, що матриця* Σ *симетрична, невироджена, додатна.*

Інтегруючи по $\mathbb{R}^n$, можна переконатися в тому, що це дійсний розподіл, а параметри μ і Σ виправдовують свою назву:

$$\int \aleph(x;\mu,\Sigma)\,dx = 1$$

$$\mathrm{E}x = \int x\aleph(x;\mu,\Sigma)\,dx = \mu$$

$$\mathrm{E}(x-\mu)(x-\mu)^{\mathrm{T}} = \int (x-\mu)(x-\mu)^{\mathrm{T}} \aleph(x;\mu,\Sigma)\,dx = \Sigma$$

Геометрична інтерпретація нормальної щільності. Якщо ознаки некорельовані, $\Sigma = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_n^2)$, то лінії рівня щільності розподілу мають форму еліпсоїдів з центром μ і осями, паралельними лініям координат. Якщо ознаки мають однакові дисперсії, $\Sigma = \sigma^2 I_n$, то еліпсоїди є сферами.

Якщо ознаки корельовані, то матриця Σ не діагональна й лінії рівня мають форму еліпсоїдів, осі яких повернені відносно початкової системи координат. Дійсно, як будь-яка симетрична матриця, Σ має спектральне розкладання $\Sigma = VSV^{\mathrm{T}}$, де $V = (v_1, v_2, ..., v_n)$ – ортогональні власні вектори матриці Σ, відповідні до власних значень $\lambda_1, \lambda_2, ..., \lambda_n$, матриця S діагональна, $S = diag(\lambda_1, \lambda_2, ..., \lambda_n)$. Тоді $\Sigma^{-1} = VS^{-1}V^{\mathrm{T}}$, отже,

$$(x-\mu)^{\mathrm{T}}\Sigma^{-1}(x-\mu) = (x-\mu)^{\mathrm{T}}VS^{-1}V^{\mathrm{T}}(x-\mu) = (x'-\mu')^{\mathrm{T}}S^{-1}(x'-\mu').$$

Це означає, що в результаті ортогонального перетворення координат $x' = V^{\mathrm{T}} x$ осі еліпсоїдів стають паралельні лініям координат. У нових координатах коваріаційна матриця S є діагональною. Тому лінійне перетворення V називають *декорелюючим*. У початкових координатах осі еліпсоїдів спрямовані вздовж власних векторів матриці Σ.

8.2. Квадратичний дискримінант

Розглянемо задачу класифікації з довільним числом класів.

Теорема 8.1. *Якщо класи мають n-вимірні нормальні щільності розподілу* $p_y(x) = \aleph(x; \mu_y, \Sigma_y), y \in Y$, *то байєсовський класифікатор задає квадратичну поділяючу поверхню. Вона вироджується в лінійну, якщо коваріаційні матриці класів рівні.*

Геометрія поділяючих поверхонь. Розглянемо найпростіший випадок, коли класи *s,t* рівноімовірні й рівнозначні $(\lambda_s P_s = \lambda_t P_t)$, коваріаційні матриці рівні, ознаки некорельовані й мають однакові дисперсії $(\Sigma_s = \Sigma_t = \sigma I_n)$. Це означає, що класи мають однакову сферичну форму. У цьому випадку поділяюча гіперплощина проходить посередині між класами, ортогонально лінії, що з'єднує центри класів. Нормаль гіперплощини має властивість оптимальності – це та пряма, в одномірній проекції на яку класи розділяються найкраще, тобто з найменшим значенням байєсовського ризику *R(a)*.

Якщо ознаки корельовані $(\Sigma_s = \Sigma_t \neq \sigma I_n)$, то ортогональність зникає, однак поділяюча гіперплощина як і раніше проходить посередині між класами, дотично до ліній рівня обох розподілів.

Якщо класи не рівноімовірні або не рівнозначні $(\lambda_s P_s \neq \lambda_t P_t)$, то поділяюча гіперплощина відсувається далі від більш значимого класу.

Якщо коваріаційні матриці не діагональні й не рівні, то поділяюча поверхня стає квадратичною й «прогинається» так, щоб менш щільний клас «охоплював» більш щільний.

У деяких випадках більш щільний клас «розріже» менш щільний на дві незв'язні області. Це може приводити до парадоксальних ситуацій. Наприклад, може виникнути область, у якій об'єкти будуть відноситися до менш щільного класу, хоча об'єкти більш щільного класу перебувають ближче.

Якщо кількість класів перевищує 2, то поділяюча поверхня є кусочно-квадратичною, а при рівних коваріаційних матрицях – кусочно-лінійною.

Відстань Махаланобиса. Якщо класи рівноімовірні й рівнозначні, коваріаційні матриці рівні, то рівняння поділяючої поверхні має вигляд

$$(x - \mu_s)^{\mathrm{T}} \Sigma^{-1} (x - \mu_s) = (x - \mu_t)^{\mathrm{T}} \Sigma^{-1} (x - \mu_t);$$

$$\|(x - \mu_s)\|_{\Sigma} = \|(x - \mu_t)\|_{\Sigma};$$

де $\|u-v\|_{\Sigma} \equiv \sqrt{(u-v)^{\mathrm{T}}\Sigma^{-1}(u-v)}$ – метрика в $\mathbb{R}^n$, названа *відстанню Махаланобіса*. Поділяюча поверхня є геометричним місцем точок, рівновіддалених від центрів класів у сенсі відстані Махаланобиса.

Якщо ознаки незалежні й мають однакові дисперсії, то відстань Махаланобіса збігається з евклідовою метрикою. У цьому випадку оптимальним (байєсовським) вирішальним правилом є *класифікатор найближчого середнього* (nearest mean classifier), що відносить об'єкт до класу з найближчим центром.

Принцип максимуму правдоподібності становить основу параметричного підходу. Нехай задана множина об'єктів $X_m = \{x_1, x_2, ..., x_m\}$, вибраних незалежно один від одного з імовірнісного розподілу з щільністю $\varphi(x;\theta)$. Функцією правдоподібності називають спільну щільність розподілу всіх об'єктів вибірки:

$$p(X_m;\theta) = p(x_1, x_2, ..., x_m;\theta) = \prod_{i=1}^{m} \varphi(x_i;\theta).$$

Значення параметра θ, при якому вибірка максимально правдоподібна, тобто функція $p(X_m;\theta)$ набуває максимального значення, називають *оцінкою максимуму правдоподібності*. Як відомо з математичної статистики, ця оцінка має ряд оптимальних властивостей.

Замість максимізації правдоподібності зручніше максимізувати його логарифм:

$$L(X_m;\theta) = \sum_{i=1}^{m} \ln \varphi(x_i;\theta) \to \max_{\theta} \tag{8.1}$$

Для розв'язку цієї задачі можна використовувати стандартні методи оптимізації. У деяких випадках розв'язок виписується в явному виді, виходячи з необхідної умови оптимуму (якщо функція $\varphi(x;\theta)$ достатньо гладка за параметром θ):

$$\frac{\partial}{\partial\theta} L(X_m;\theta) = \sum_{i=1}^{m} \frac{\partial}{\partial\theta} \ln \varphi(x_i;\theta) = 0 \tag{8.2}$$

Вибіркові оцінки параметрів нормального розподілу. У випадку гаусовської щільності з параметрами $\theta \equiv (\mu, \Sigma)$ задача максимізації правдоподібності має аналітичний розв'язок, який можна отримати з рівнянь $\frac{\partial}{\partial\theta} L(X_m;\theta) = \sum_{i=1}^{m} \frac{\partial}{\partial\theta} \ln \varphi(x_i;\theta) = 0$.

Теорема 8.2. Нехай задана незалежна вибірка об'єктів $X_m = (x_1, x_2, ..., x_m)$. Тоді оцінки параметрів гаусовської щільності $\varphi(x;\theta) = \aleph(x;\mu,\Sigma)$, що надають максимум функціоналу правдоподібності $L(X_m;\theta) = \sum_{i=1}^{m} \ln \varphi(x_i;\theta) \to \max_{\theta}$, мають вигляд

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} x_i ; \quad \hat{\Sigma} = \frac{1}{m}\sum_{i=1}^{m} (x_i - \hat{\mu})(x_i - \hat{\mu})^{\mathrm{T}}.$$

Поправка на зміщення. Природньою вимогою до оцінки параметра розподілу є її незміщеність.

Озн. 8.3. Нехай X_m – вибірка випадкових незалежних спостережень, отримана згідно з розподілом $\varphi(x;\theta)$ при фіксованому $\theta = \theta_0$. Оцінку $\hat{\theta}(X_m)$ параметра θ, обчислену по вибірці X_m, називають незміщеною, якщо $\mathrm{E}_{X_m}\hat{\theta}(X_m) = \theta_0$.

Оцінка $\hat{\mu}$ є незміщеною, $\mathrm{E}\hat{\mu} = \mu$. Однак оцінка $\hat{\Sigma}$ уже зміщена: $\mathrm{E}\hat{\Sigma} = \frac{m}{m-1}\Sigma$. Це пов'язано з тим, що при обчисленні $\hat{\Sigma}$ замість невідомого точного значення математичного сподівання μ доводиться підставляти його вибіркову оцінку $\hat{\mu}$. Для обчислення цього невеликого зміщення вводиться *поправка на зміщення:*

$$\hat{\Sigma} = \frac{1}{m-1}\sum_{x=1}^{m} (x_i - \hat{\mu})(x_i - \hat{\mu})^{\mathrm{T}} \tag{8.3}$$

Підстановочний алгоритм. Оцінимо параметри функцій правдоподібності $\hat{\mu}_y$ та $\hat{\Sigma}_y$ за частинами навчальної вибірки $X_m^y = \{x_i \in X_m \mid y_i = y\}$ для кожного класу $y \in Y$. Потім ці вибіркові оцінки підставимо у формулу $a(x) = \arg\max_{y \in Y} \lambda_y P_y p_y(x)$. Одержимо байєсовський нормальний класифікатор, який називають також підстановочним (plug-in).

При $m_y \to \infty$ оцінки $\hat{\mu}_y$ та $\hat{\Sigma}_y$ мають ряд оптимальних властивостей: вони не зміщені, спроможні і ефективні. Однак оцінки, зроблені по коротких вибірках, можуть бути не досить точними.

Недоліки підстановочного алгоритму випливають із декількох надмірно сильних базових припущень, які на практиці часто не виконуються.

- Функції правдоподібності класів можуть суттєво відрізнятися від гаусовських. Зокрема, коли є ознаки, що приймають дискретні значення, або коли класи розпадаються на ізольовані згустки.
- Якщо довжина вибірки менша за розмірність простору, $m_y < n$, або серед ознак є лінійно залежні, то матриця $\hat{\Sigma}_y$ стає виродженою. У цьому випадку обернена матриця не існує й метод взагалі не застосовний.
- На практиці зустрічаються задачі, у яких ознаки «майже лінійно залежні». Тоді матриця $\hat{\Sigma}_y$ є *погано обумовленою*, тобто близькою до деякої виродженої матриці. Це так звана *проблема мультиколінеарності*, яка призводить до нестійкості як самої оберненої матриці $\hat{\Sigma}_y^{-1}$, так і вектора

$\hat{\Sigma}_y^{-1}\left(s-\mu_{st}\right)$. Вони можуть непередбачувано й сильно змінюватися при незначних варіаціях вхідних даних, наприклад, пов'язаних з погрішностями вимірів. Нестійкість знижує якість класифікації.

- Вибіркові оцінки чутливі до порушень нормальності розподілів, зокрема, до нечисленних великих викидів.

Наївний нормальний байєсовський класифікатор. Припустимо, що всі ознаки $f_j\left(x\right)$ незалежні й нормально розподілені з математичним сподіванням μ_{yj} і дисперсією σ_{yj}, які відрізняються для різних класів:

$$p_{yj}\left(\xi\right)=\frac{1}{\sqrt{2\pi}\sigma_{yj}}\exp\left(-\frac{\left(\xi-\mu_{yj}\right)^2}{2\sigma_{yj}^2}\right), y\in Y, j=1,2,...,n$$

Тоді, як неважко переконатися, коваріаційні матриці Σ_y і їх вибіркові оцінки $\hat{\Sigma}_y$ будуть діагональні. У цьому випадку проблеми виродженості й мультиколінеарності не виникають. Метод навчання простий і зводиться до обчислення параметрів $\hat{\mu}_{yj}$ та $\hat{\sigma}_{yj}$ для всіх $y\in Y$ і всіх ознак $j=1,2,...,n$.

8.3. Лінійний дискримінант Фішера

В 1936 р. Р. Фишер запропонував просту евристику, що дозволяє збільшити число об'єктів, по яких оцінюється коваріаційна матриця, підвищити її стійкість і заодно спростити алгоритм навчання. Припустимо, що коваріаційні матриці класів однакові й рівні Σ. Оцінимо $\hat{\Sigma}$ по всіх m навчальних об'єктах. З урахуванням виправлення на зміщення,

$$\hat{\Sigma}=\frac{1}{m-\left|Y\right|}\sum_{i=1}^{m}\left(x_i-\hat{\mu}_{y_i}\right)\left(x_i-\hat{\mu}_{y_i}\right)^{\mathrm{T}}.$$

Згідно з теоремою 8.1, поділяюча поверхня линійна, якщо класів два, і кусочно-лінійна, якщо класів більше. Запишемо підстановочний алгоритм:

$$a\left(x\right)=\arg\max_{y\in Y}\left(\lambda_y P_y p_y\left(x\right)\right)=$$

$$=\arg\max_{y\in Y}\left(\underbrace{\ln\left(\lambda_y P_y-\frac{1}{2}\hat{\mu}_y^{\mathrm{T}}\hat{\Sigma}^{-1}\hat{\mu}_y\right)}_{\beta_y}+x^{\mathrm{T}}\underbrace{\hat{\Sigma}^{-1}\hat{\mu}_y}_{\alpha_y}\right)= \tag{8.4}$$

$$=\arg\max_{y\in Y}\left(x^{\mathrm{T}}\alpha_y+\beta_y\right)$$

Цей алгоритм називають *лінійним дискримінантом Фішера* (ЛДФ). Він непогано працює, коли форми класів дійсно близькі до нормальних і не занадто сильно різняться. У цьому випадку лінійне вирішальне правило

близьке до оптимального байєсовського, але суттєво більш стійке, ніж квадратичне, і часто має кращу узагальнюючу здатність.

Імовірність помилки лінійного дискримінанта Фішера виражають через відстань Махаланобіса між класами, у випадку, коли класів два:

$$R(a)=\Phi\left(-\frac{1}{2}\|\mu_1-\mu_2\|_\Sigma\right),$$

де $\Phi(r)=\aleph(x;0,1)$ – функція стандартного нормального розподілу.

Регуляризація коваріаційної матриці. Евристика Фішера не є радикальним розв'язком проблеми мультиколінеарності: загальна коваріаційна матриця класів $\hat{\Sigma}$ також може виявитися погано обумовленою (близькою до виродженої). Ознакою поганої обумовленості є наявність у матриці власних значень, близьких до нуля. Тому один зі способів розв'язку проблеми – модифікувати матрицю $\hat{\Sigma}$ так, щоб усі її власні значення λ збільшилися на задане число τ, а всі власні вектори v збереглися. Для цього достатньо додати до матриці одиничну, помножену на τ:

$$\left(\hat{\Sigma}+\tau I_n\right)v=\lambda v+\tau v=\left(\lambda+\tau\right)v.$$

Відомі й інші способи розв'язування проблеми поганої обумовленості. Можна пропорційно зменшувати недіагональні елементи – замість $\hat{\Sigma}$ брати матрицю $(1-\tau)\hat{\Sigma}+\tau diag\hat{\Sigma}$. Можна зануляти недіагональні елементи матриці, які відповідають тим парам ознак, коваріації яких дещо відрізняються від нуля; при цьому матриця стає розрідженою, і для її обернення можуть застосовуватися спеціальні, більш ефективні, алгоритми. Можна розбивати множину ознак на групи й вважати, що ознаки з різних груп не корельовані. Тоді матриця $\hat{\Sigma}$ набуває блочно-діагонального вигляду. Для таких матриць також існують спеціальні ефективні алгоритми обернення.

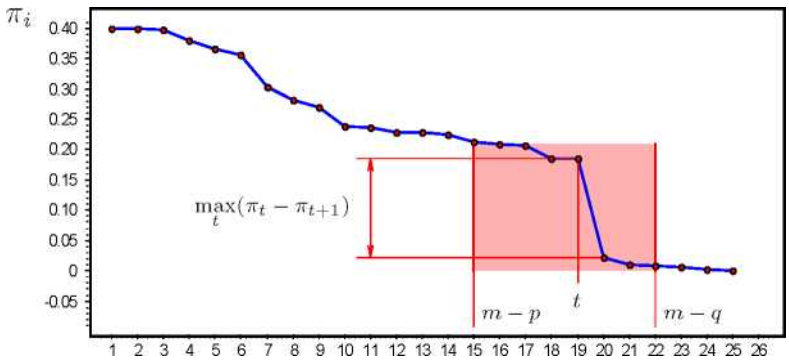


Рис. 8.1. Експерт припустив, что у вибірці довжини m = 25 міститься від q=3 до p=10 викидів. Більш точну кількість викидів (6) вдалося визначити за критерієм «крутого схилу».

Відбір і перетворення ознак. Інший спосіб усунення мультиколінеарності полягає в тому, щоб відкинути найменш значимі ознаки. Звертаємо увагу на, здається, парадокс: інформація відкидається, але якість класифікації підвищується.

Існує простий «жадібний» метод відбору ознак, у якому початкова *n*-вимірна задача зводиться до серії двовимірних задач. Це метод *редукції розмірності* А. М. Шуригіна. Спочатку шукають дві ознаки, у підпросторі яких лінійний дискримінант Фишера щонайкраще розділяє класи (тобто байєсовський ризик *R(a)* набуває найменшого значення). Позначимо його параметри через $\left\{\alpha_y^{(2)}, \beta_y^{(2)} : y \in Y\right\}$, де у векторі $\alpha_y^{(2)} \in \mathbb{R}^n$ лише два коефіцієнти не дорівнюють нулю. Функція проекції на нормаль поділяючої гіперплощини приймається за нову ознаку: $\psi(x) = x^{\mathrm{T}} \alpha_y^{(2)}$. З тих *n*–2 ознак, що залишилися, вибирається та, яка у парі з $\psi(x)$ щонайкраще розділяє класи. Знову будується двовимірний лінійний дискримінант і визначаються параметри $\left\{\alpha_y^{(3)}, \beta_y^{(3)} : y \in Y\right\}$, де у векторі $\alpha_y^{(3)} \in \mathbb{R}^n$ уже три коефіцієнти не дорівнюють нулю. Так триває доти, поки приєднання нових ознак поліпшує якість класифікації (знижує байєсовський ризик). *Переваги* методу редукції – можливість відкинути неінформативні ознаки й обійтися без обернення коваріаційних матриць, більших за 2x2. У деяких прикладних задачах він перевершує інші методи класифікації. *Недоліком* є відсутність строгого теоретичного обґрунтування. Як і будь-яка «жадібна» стратегія, він знаходить неоптимальний набір ознак. Цей метод підходить для тих задач, у яких є невелике число ознак, суттєво більш інформативних, ніж інші.

Ще один спосіб скорочення розмірності полягає в тому, щоб з наявного набору ознак побудувати новий набір, що складається з найменшого числа максимально інформативних ознак. Оптимальне лінійне перетворення простору ознак будується за допомогою методу головних компонентів (principal component analysis).

Робастні методи оцінювання. Оцінки, стійкі відносно нечастих великих викидів або інших невідповідностей реальних даних модельному розподілу $\varphi(x;\theta)$, називають робастними (robust – надійний).

Найпростіший метод фільтрації викидів полягає у розв'язуванні задачі двічі. Спочатку обчислюється оцінка максимуму правдоподібності для параметра 0 по всій вибірці X_m. Для кожного об'єкта $x_i \in X_m$ обчислюється значення правдоподібності $\pi_i = \varphi\left(x_i;\hat{\theta}\right)$, і об'єкти сортуються за спаданням правдоподібностей: $\pi_1, \pi_2, ..., \pi_m$. Об'єкти, що опинилися наприкінці цього ряду, вважають нетиповими (викидами) і видаляють з вибірки. Для цього може застосовуватися критерій «крутого схилу»: задають два параметри *p* і *q*, і шукають значення $t \in \{m-p, ..., m-q-1\}$, для якого стрибок правдоподібності $\pi_t - \pi_{t+1}$ максимальний. Потім останні *(m–t)* об'єктів видаляються з вибірки, див. рис. 1. Після цього оцінка $\hat{\theta}$ обчислюється вдруге по відфільтрованій вибірці. Якщо викидів багато, то, можливо, доведеться провести декілька таких фільтрацій підряд.

Помітимо, що видалення об'єкта може не вимагати повного переобчислення оцінки $\hat{\theta}$. Зокрема, оцінки нормального розподілу $\hat{\mu}, \hat{\Sigma}$ адитивні по об'єктах, і для них достатньо відняти доданок, який відповідає об'єкту, що видаляється.

8.4. Поділ суміші розподілів

У тих випадках, коли «форму» класу не вдається описати яким-небудь одним розподілом, можна спробувати описати її сумішшю розподілів.

Гіпотеза 8.4. *Щільність розподілу на X має вигляд суміші k розподілів:*

$$p(x) = \sum_{j=1}^{k} w_j p_j(x), \quad \sum_{j=1}^{k} w_j = 1, \quad w_j \geq 0$$

де $p_j(x)$ *– функція правдоподібності j-го компоненти суміші,* w_j *– її апріорна ймовірність. Функції правдоподібності належать параметричному сімейству розподілів* $\varphi(x;\theta)$ *і відрізняються тільки значеннями параметра,* $p_j(x) = \varphi(x;\theta_j)$.

Іншими словами, «вибрати об'єкт *x* із суміші *p(x)*» означає спочатку вибрати *j*-ий компонент суміші з дискретного розподілу $\{w_1, w_2, ..., w_k\}$, потім вибрати об'єкт *x* згідно із щільністю $p_j(x)$.

Задача поділу суміші полягає в тому, щоб, маючи вибірку X_m випадкових і незалежних спостережень із суміші *p(x)*, знаючи число *k* й функцію φ, оцінити вектор параметрів $\Theta = (w_1, ..., w_k, \theta_1, ..., \theta_k)$.

8.4.1 EM-алгоритм

На жаль, спроба розділити суміш, використовуючи принцип максимуму правдоподібності «напряму», призводить до занадто громіздкої оптимізаційної задачі. Обійти ці труднощі дозволяє алгоритм EM (expectation-maximization). Ідея алгоритму полягає в наступному. Штучно вводиться допоміжний вектор *прихованих* (hidden) змінних *G*, що має дві чудові властивості. З одного боку, він може бути обчислений, якщо відомі значення вектора параметрів Θ. З іншого боку, пошук максимуму правдоподібності значно спрощується, якщо відомі значення схованих змінних.

EM-алгоритм складається з ітераційного повторення двох кроків.

На E-кроці обчислюють очікуване значення (expectation) вектора прихованих змінних *G* за поточним наближенням вектора параметрів Θ.

На M-кроці вирішують задачу максимізації правдоподібності (maximization) і знаходять наступне наближення вектора Θ за поточними значеннями векторів *G* і Θ.

Алгоритм 8.1. Загальна ідея ЕМ-алгоритму
1: Обчислити початкове наближення вектора параметрів Θ;
2: **повторювати**
3: G := Estep(Θ);
4: Θ := Mstep(Θ, G);
5: **поки** Θ і G не стабілізуються.

Цей алгоритм був запропонований і досліджений М. І. Шлезінгером як інструмент для *мимовільної (спонтанної) класифікації образів*. Через дванадцять років він був відкритий заново під назвою *ЕМ-алгоритму*. Область його застосування надзвичайно широка – дискримінантний аналіз, кластеризація, відновлення пропусків у даних, обробка сигналів і зображень. Тут ми розглядаємо його як інструмент поділу суміші розподілів.

Е-крок (expectation). Позначимо через $p(x,\theta_j)$ щільність імовірності того, що об'єкт x отриманий з j-го компоненту суміші. За формулою умовної імовірності

$$p(x,\theta_j) = p(x)\mathrm{P}(\theta_j \mid x) = w_j p_j(x).$$

Введемо позначення $g_{ij} \equiv \mathrm{P}(\theta_j \mid x)$. Це невідома апостеріорна імовірність того, що навчальний об'єкт x_i отриманий з j-го компонента суміші. Візьмемо ці величини як приховані змінні. Позначимо $G = (g_{ij})_{m\times k} = (g_1, g_2, ..., g_j)$, де g_j – j-й стовпець матриці G. Кожний об'єкт обов'язково належить якомусь компоненту, тому справедлива формула повної імовірності:

$$\sum_{j=1}^{k} g_{ij} = 1 \text{ для всіх } i = 1,2,...,m.$$

Знаючи параметри компонентів w_j, θ_j, легко обчислити g_{ij} за формулою Байєса:

$$g_{ij} = \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} \text{ для всіх } i, j. \qquad (8.5)$$

У цьому і полягає Е-крок алгоритму ЕМ.

М-крок (maximization). Покажемо, що знання значень прихованих змінних g_{ij} і принцип максимуму правдоподібності приводять до оптимізаційної задачі, що допускає ефективний чисельний (або навіть аналітичний) розв'язок. Будемо максимізувати логарифм правдоподібності

$$Q(\Theta) = \ln \prod_{i=1}^{m} p(x_i) = \sum_{i=1}^{m} \ln \sum_{j=1}^{k} w_j p_j(x_i) \to \max_{\Theta} \text{ при обмеженні } \sum_{j=1}^{k} w_j = 1.$$

Запишемо лагранжиан цієї оптимізаційної задачі

$$L(\Theta, X_s) = \sum_{i=1}^{s} \ln\left(\sum_{j=1}^{k} w_j p_j(x_i) \right) - \lambda \left(\sum_{j=1}^{k} w_j - 1 \right)$$

Прирівняємо до нуля похідну лагранжиана по w_j:

$$\frac{\partial L}{\partial w_j} = \sum_{i=1}^{l} \frac{p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} - \lambda = 0, \quad j = 1,2,...,k \tag{8.6}$$

Помножимо ліву й праву частини на w_j, додамо всі k рівностей, і поміняємо місцями знаки додавання по j та по i:

$$\underbrace{\sum_{i=1}^{l} \sum_{j=1}^{k} \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)}}_{=1} = \lambda \underbrace{\sum_{j=1}^{k} w_j}_{=1}$$

звідки випливає $\lambda = m$.

Тепер знову помножимо ліву й праву частини (8.6) на w_j, підставимо $\lambda = m$, і, зауважуючи подібність із формулою (8.5), одержимо вирази для ваг компонентів через приховані змінні:

$$w_j = \frac{1}{l} \sum_{i=1}^{l} \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} = \frac{1}{l} \sum_{i=1}^{l} g_{ij}, \quad j = 1,2,...,k \tag{8.7}$$

Легко перевірити, що обмеження-нерівності $w_j \geq 0$ будуть виконані на кожній ітерації, якщо вони виконані для початкового наближення.

Прирівняємо до нуля похідну лагранжиана по θ_j, пам'ятаючи, що $p_j(x) \equiv \varphi(x;\theta_j)$:

$$\frac{\partial L}{\partial \theta_j} = \sum_{i=1}^{l} \frac{w_j}{\sum_{s=1}^{k} w_s p_s(x_i)} \frac{\partial}{\partial \theta_j} p_j(x_i) = \sum_{i=1}^{l} \frac{w_j p_j(x_i)}{\sum_{s=1}^{k} w_s p_s(x_i)} \ln p_j(x_i) =$$

$$= \sum_{i=1}^{l} g_{ij} \frac{\partial}{\partial \theta_j} \ln p_j(x_i) = \frac{\partial}{\partial \theta_j} \sum_{i=1}^{l} g_{ij} p_j(x_i) = 0, \quad j = 1,2,...,k$$

Отримана умова збігається з необхідною умовою максимуму в задачі максимізації зваженої правдоподібності

$$\theta_j := \arg\max_{\theta} \sum_{i=1}^{l} g_{ij} \ln \varphi(x_i;\theta), \quad j = 1,2,...k \tag{8.8}$$

Таким чином, М-крок зводиться до обчислення ваг компонентів w_j як середніх арифметичних (8.7) і оцінювання параметрів компонент θ_j шляхом розв'язування k незалежних оптимізаційних задач (8.8). Відзначимо, що розділення змінних виявилося можливим завдяки вдалому введенню прихованих змінних.

Алгоритм 8.2. ЕМ-алгоритм з фіксованим числом компонентів

Вхід:

вибірка $X_m = \{x_1, x_2, ..., x_m\}$, k – число компонентів суміші;

$\Theta = \left(w_j, \theta_j\right)_{j=1}^{k}$ – початкове наближення параметрів суміші;

δ – параметр критерію останова;

Алгоритм 8.2. ЕМ-алгоритм з фіксованою кількістю компонентів

Вихід:

$\Theta = \left(w_j, \theta_j\right)_{j=1}^{k}$ – оптимізований вектор параметрів суміші;

1: **ПРОЦЕДУРА** $EM\left(X_m, k, \Theta, \delta\right)$

2: повторювати

3: Е-крок (expectation):

для всіх $i = 1, 2, ..., m, \quad j = 1, 2, ..., k$

$$g_{ij}^0 := g_{ij}; \quad g_{ij} := \frac{w_j \varphi\left(x_i; \theta_j\right)}{\sum_{s=1}^{k} w_s \varphi\left(x_i; \theta_s\right)}$$

4: М-крок (maximization):

для всіх $j = 1, 2, ..., k$

$$\theta_j := \arg\max_{\theta} \sum_{i=1}^{m} g_{ij} \ln \varphi\left(x_i; \theta\right), \quad w_j := \frac{1}{m} \sum_{i=1}^{m} g_{ij}$$

5: поки $\max_{i,j} \left| g_{ij} - g_{ij}^0 \right| > \delta$

6: повернути $\left(w_j, \theta_j\right)_{j=1}^{k}$

Критерій зупинки. Ітерації зупиняються, коли значення функціонала $Q(\Theta)$ або прихованих змінних G перестають істотно змінюватися. Зручніше контролювати приховані змінні, тому що вони мають зміст імовірностей і приймають значення з відрізка [0,1].

Реалізація ітераційного процесу показана в Алгоритмі 8.2. На Е-кроці обчислюється матриця прихованих змінних G за формулою (8.7). На М-кроці вирішується серія з k задач максимізації зваженої правдоподібності (2.20), кожна з них – за повною вибіркою X_m з вектором ваг g_j.

Узагальнений ЕМ-алгоритм. Не обов'язково домагатися високої точності розв'язку оптимізаційної задачі (8.8) на кожному М-кроці алгоритму. Достатньо лише зміститися в напрямку максимуму, зробивши одну або кілька ітерацій, і потім виконати Е-крок. Цей алгоритм також має непогану збіжність

і його називають *узагальненим EM-алгоритмом* (generalized Em-algorithm, GEM).

Проблема вибору початкового наближення. Алгоритм EM сходиться при досить загальних припущеннях до локального оптимуму. Якість цього розв'язку й швидкість збіжності суттєво залежать від початкового наближення. Збіжність погіршується в тих випадках, коли робиться спроба помістити декілька компонентів в один фактичний згусток розподілу, або розмістити компонент посередині між згустками. Стандартна (але далеко не найкраща) евристика полягає в тому, щоб вибрати параметри початкових компонентів випадковим чином. Інша ідея – вибрати як центри компонентів k об'єктів, максимально віддалених один від одного. Можна стартувати ітераційний процес багато разів з різних початкових наближень і потім вибрати найкращий розв'язок.

Алгоритм 8.3. EM-алгоритм з послідовним додаванням компонентів

Вхід:

вибірка $X_m = \{x_1, x_2, ..., x_m\}$;

R – максимальний припустимий діапазон правдоподібності об'єктів;

m_0 – мінімальна довжина вибірки, за якою можна відновити щільність;

δ – параметр критерію останова;

Вихід:

k – число компонентів суміші;

$\Theta = \left(w_j, \theta_j\right)_{j=1}^{k}$ – ваги й параметри компонентів;

1: початкове наближення – один компонент:

$$\theta_1 := \arg\max_{\theta} \sum_{i=1}^{m} g_{ij} \ln \varphi(x_i; \theta), \quad w_1 := 1; \quad k := 1$$

2: **для всіх** $k := 2, 3, ...$

3: виділити об'єкти з низькою правдоподібністю:

$$U := \left\{ x_i \in X_m : p(x_i) < \max_j p(x_i) / R \right\};$$

4: **якщо** $|U| < m_0$ **то**

5: **вихід** з циклу по k;

6: початкове наближення для k-го компоненту:

$$\theta_k := \arg\max_{\theta} \sum_{x_i \in U} g_{ij} \ln \varphi(x_i; \theta), \quad w_k := \frac{1}{m}|U|$$

$$w_j := w_j (1 - w_k), \quad j = 1, 2, ..., k-1$$

7: $EM\left(X_m, k, \Theta, \delta\right)$;

EM-алгоритм з послідовним додаванням компонентів дозволяє розв'язати дві проблеми відразу – вибору початкового наближення й вибору кількості компонент. Ідея полягає в наступному. Маючи деякий набір

компонентів, можна виділити об'єкти x_i, які гірше за все описуються сумішшю – це об'єкти з найменшими значеннями правдоподібності $p(x_i)$. По цих об'єктах будується ще один компонент. Потім він додається до суміші і запускаються ЕМ-ітерації, щоб новий компонент і старий «притерлися один до одного». Так триває доти, поки всі об'єкти не виявляться покритими компонентами.

Реалізація цієї ідеї представлена в Алгоритмі 8.3. На кроці 1 будується перший компонент і покладається k=1. Потім у циклі послідовно додається по одному компоненту. Якщо значення правдоподібності $p(x_i)$ в R разів менше за максимальне значення правдоподібності, то це означає, що об'єкт x_i погано описується сумішшю. Помітимо, що це лише евристика; зовсім не обов'язково порівнювати $p(x_i)$ саме з максимальною правдоподібністю; можна брати середня правдоподібність або фіксоване граничне значення P_0. На кроці 3 формується підвибірка U з об'єктів, які не підходять до жодного з компонентів. Якщо довжина цієї підвибірки менша за поріг m_0, то процес додавання компонентів на цьому закінчується, а об'єкти, що залишилися, вважаються викидами. На кроці 6 знову застосовується метод максимуму правдоподібності для формування нового компонента, але тепер уже не за всією вибіркою, а тільки за підвибіркою U. Ваги компонентів переобчислюються таким чином, щоб їх сума, як і раніше, дорівнювала одиниці. На кроці 7 усі попередні компоненти разом з новим компонентом проходять через цикл ітерацій ЕМ-алгоритму.

Стохастичний ЕМ-алгоритм. Функціонал $Q(\Theta)$, який максимізується, у загальному випадку може мати велику кількість локальних екстремумів. Тому ЕМ-алгоритму властиві звичайні недоліки будь-якого детермінованого процесу багатоекстремальної оптимізації: застрягання в локальних максимумах, залежність розв'язку від початкового наближення, повільна збіжність при невдалому виборі початкового наближення. Зазвичай такого роду недоліки долаються методами адаптивної стохастической оптимізації.

Відмінність одного з варіантів *стохастичного ЕМ-алгоритму* (stochastic EM-algorithm, SEM) від Алгоритму 2.2 у тому, що на М-кроці (крок 4) замість максимізації зваженої правдоподібності

$$\theta_j := \arg\max_{\theta} \sum_{i=1}^{m} g_{ij} \ln \varphi(x_i;\theta)$$

вирішується задача максимізації звичайної, незваженої, правдоподібності

$$\theta_j := \arg\max_{\theta} \sum_{x_i \in X_j} \ln \varphi(x_i;\theta)$$

де вибірки X_j генеруються з X_m шляхом стохастического моделювання. Для кожного об'єкта $x_i \in X_m$ розігрується випадкове значення $j(i)$ з дискретного розподілу імовірностей $\{g_{ij} : j = 1,2,...,k\}$, і об'єкт $x_i \in X_m$ включається тільки у вибірку $X_{j(i)}$.

Інша стратегія – число компонент k послідовно зменшується, починаючи з деякого свідомо надлишкового числа k_{max}. Якщо компонента виявляється занадто нечисленною, $\left|X_j\right| \le m_0$, то вона видаляється. Можливо також сполучення обох стратегій.

Переваги SEM обумовлені тим, що рандомізація «вибиває» оптимізаційний процес із локальних максимумів. SEM працює швидше звичайного детермінованого EM, і його результати менше залежать від початкового наближення. Як правило, SEM знаходить экстремум $Q(\Theta)$, більш близький до глобального.

8.4.2. Суміші багатовимірних нормальних розподілів

Розглянемо розв'язування задачі M-кроку в окремому випадку, коли компоненти мають нормальні (гаусовські) щільності. У цьому випадку функціонал (2.20) є квадратичним і додатно визначеним, тому розв'язок виписується в явному аналітичному виді.

Гаусовські суміші загального виду

Теорема 8.5. *Нехай компоненти суміші мають n-вимірні нормальні розподіли* $\varphi(x;\theta_j) = \aleph(x;\mu_j,\Sigma_j)$ *з параметрами* $\theta_j = (\mu_j,\Sigma_j)$, *де* $\mu_j = \square^n$ – *вектор математичного сподівання,* $\Sigma_j \in \square^{n\times n}$ – *коваріаційна матриця, j = 1,... , k. Тоді стаціонарна точка оптимізаційної задачі (2.20) має вигляд*

$$\hat{\mu}_j = \frac{1}{mw_j}\sum_{i=1}^{m} g_{ij}x_i, j = 1,2,...k \ ;$$

$$\hat{\Sigma}_j = \frac{1}{mw_j}\sum_{i=1}^{m} g_{ij}\left(x_i - \hat{\mu}_j\right)\left(x_i - \hat{\mu}_j\right)^{\mathrm{T}}, j = 1,2,...k$$

Таким чином, M-крок зводиться до обчислення вибіркового середнього й вибіркової коваріаційної матриці для кожного компонента суміші зі своїм розподілом ваг об'єктів. Вага i-го об'єкта для j-ї компоненти дорівнює g_{ij} – оцінці приналежності даного об'єкта даному компоненту, обчисленій на E-кроці.

Суміші багатовимірних нормальних розподілів дозволяють наближати будь-які неперервні щільності імовірності. Вони є універсальними апроксиматорами щільностей, подібно до того, як поліноми є універсальними апроксиматорами неперервних функцій. У практичних задачах це дозволяє відновлювати функції правдоподібності класів навіть у тих випадках, коли на перший погляд для виконання умов Теореми 8.5 немає змістовних підстав.

Недоліком гаусовських сумішей є необхідність обертати коваріаційні матриці. Це трудомістка операція. Крім того, коваріаційні матриці часто виявляються виродженими або погано обумовленими, що призводить до нестійкості вибіркових оцінок щільності й самого класифікатора. Стандартні прийоми (регуляризація, метод головних компонентів) дозволяють упоратися

з цією проблемою. Але є й інший вихід – використовувати для опису компонентів більш прості розподіли, наприклад, сферичні.

Гаусовські суміші з діагональними матрицями коваріації. Трудомісткого обернення матриць можна уникнути, якщо прийняти гіпотезу, що в кожному компоненті суміші ознаки некорельовані. У цьому випадку гаусіани спрощуються, залишаючись, проте, універсальними апроксиматорами щільності.

Теорема 8.6. *Нехай компоненти суміші мають n-вимірні нормальні розподіли з параметрами* $\left(\mu_j, \Sigma_j\right)$, *де* $\mu_j = \left(\mu_{j1}, \mu_{j2}, ..., \mu_{jn}\right)$, $\Sigma_j = diag\left(\sigma_{j1}^2, \sigma_{j2}^2, ..., \sigma_{jn}^2\right)$ *– діагональна матриця, j = 1,2,..., k:*

$$\varphi\left(x;\theta_j\right) = \aleph\left(x;\mu_j,\sigma_j\right) = \prod_{d=1}^{n} \frac{1}{\sigma_{jd}\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\xi_d - \mu_{jd}}{\sigma_{jd}}\right)^2\right), x = \left(\xi_1, \xi_2, ..., \xi_n\right)$$

Тоді стаціонарна точка оптимізаційної задачі (2.20) має вигляд:

$$\hat{\mu}_{jd} = \frac{1}{mw_j}\sum_{i=1}^{m} g_{ij} x_{id}, d = 1,2,...n \ ;$$

$$\hat{\sigma}_{jd}^2 = \frac{1}{mw_j}\sum_{i=1}^{m} g_{ij}\left(x_{id} - \hat{\mu}_{jd}\right)^2, d = 1,2,...n$$

де $x_i = \left(x_{i1}, x_{i2}, ..., x_{in}\right)$ *– об'єкти вибірки* X_m.

Можна було б припустити, що компоненти мають сферичні щільності, $\Sigma_j = \sigma_j^2 I_n$. Однак таке припущення має очевидний недолік: якщо ознаки суттєво різняться по порядку величини, то компоненти будуть мати дуже витягнуту форму, яку доведеться апроксимувати більшою кількістю сферичних гаусіанів. Припущення про нерівні дисперсії ознак приводить до алгоритму класифікації, не чутливому до відмінностей у масштабах виміру ознак.

Радіальні функції. Гаусіан $p_j\left(x\right) = \aleph\left(x;\mu_j,\Sigma_j\right)$ з діагональною матрицею Σ_j можна записати у вигляді $p_j\left(x\right) = \aleph_j \exp\left(-\frac{1}{2}\rho_j^2\left(x,\mu_j\right)\right)$,

де $\aleph_j = \left(2\pi\right)^{-\frac{n}{2}}\left(\sigma_{j1}...\sigma_{jn}\right)^{-1}$ – нормувальний множник, $\rho_j\left(x,x'\right)$ *– зважена евклідова метрика* в *n*-вимірному просторі *X*:

$$\rho_j^2\left(x,x'\right) = \sum_{d=1}^{n} \sigma_{jd}^{-2}\left|\xi_d - \xi_d'\right|^2, x = \left(\xi_1, \xi_2, ..., \xi_n\right), x' = \left(\xi_1', \xi_2', ..., \xi_n'\right)$$

Чим менша відстань $\rho_j\left(x,\mu_j\right)$, тим вищим є значення щільності в точці *x*. Тому щільність $p_j\left(x\right)$ є *функцією близькості* вектора *x* до центру μ_j.

Функції *f (x),* що залежать тільки від відстані між *x* і фіксованою точкою простору *X*, прийнято називати *радіальними*.

8.4.3 Мережа радіальних базисних функцій

Вище ми розглядали задачу поділу суміші розподілів, забувши на деякий час про початкову задачу класифікації.

Нехай тепер $Y=\{1,2,...M\}$, кожний клас $y\in Y$ має свою щільність розподілу $p_y(x)$ і представлений частиною вибірки $X_{my}=\{(x_i,y_i)\in X_m \mid y_i=y\}$

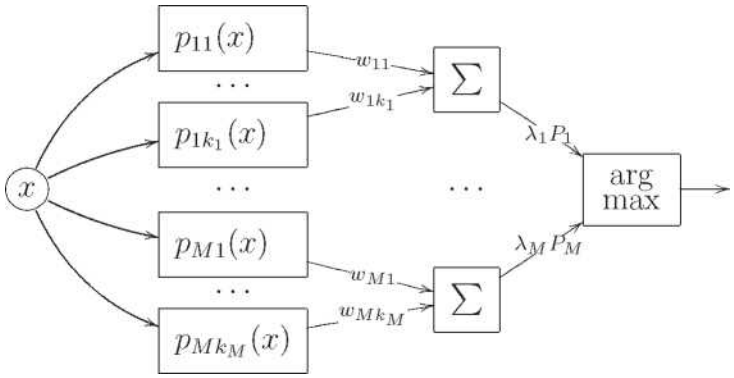


Рис. 8.1. Мережа радіальних базисних функцій є трьохрівневою суперпозицією.

Нехай функції правдоподібності класів $p_y(x)$, $y\in Y$, можуть бути представлені у вигляді сумішей k_y компонентів. Кожний компонент має n-вимірну гаусівську щільність з параметрами $\mu_{yj}=(\mu_{yj1},\mu_{yj2},...,\mu_{yjn})$, $\Sigma_{ji}=diag(\sigma^2_{yj1},\sigma^2_{yj2},...,\sigma^2_{yjn})$, $j=1,2,...,k_y$:

$$p_y(x)=\sum_{j=1}^{k_y} w_{yj}p_{yj}(x),\quad p_{yj}(x)=\aleph(x;\mu_{yj},\Sigma_{yj}),\quad \sum_{j=1}^{k_y} w_{yj}=1,\quad w_{yj}\ge 0$$

Алгоритм класифікації. Запишемо байєсовске вирішальне правило (2.2), виразивши щільність кожного компонента $p_{yj}(x)$ через зважену евклідову відстань від об'єкта x до центру компонента μ_{yj}:

$$a(x)=\arg\max_{y\in Y}\lambda_y P_y\sum_{j=1}^{k_y} w_{yj}\underbrace{\aleph_{yj}\exp\left(-\frac{1}{2}\rho^2_{yj}(x,\mu_{yj})\right)}_{p_{yj}(x)} \qquad (2.21)$$

де $\aleph_{yj}=(2\pi)^{-\frac{n}{2}}(\sigma_{yj1}...\sigma_{yjn})^{-1}$ – нормувальні множники. Алгоритм має вигляд суперпозиції, що складається з трьох рівнів або шарів, Рис 8.1.

Перший шар утворений $k_1+k_2+...+k_M$ гаусіанами $p_{yj}(x)$, $y\in Y$, $j=1,2,...,k_y$. На вході вони приймають опис об'єкта x, на виході видають оцінки близькості об'єкта x до центрів μ_{yj}, які дорівнюють значенням щільностей компонентів у точці x.

Другий шар складається з M суматорів, що обчислюють зважені середні цих оцінок з вагами w_{yj}. На виході другого шару з'являються оцінки приналежності об'єкта x кожному з класів, які дорівнюють значенням щільностей класів $p_{yj}(x)$.

Третій шар утворюється єдиним блоком *argmax*, що приймає остаточне рішення про віднесення об'єкта x до одного з класів.

Таким чином, при класифікації об'єкта x оцінюється його близькість до кожного з центрів μ_{yj} за метрикою $\rho_{yj}(x, \mu_{yj})$, $j = 1,2,...,k_y$. Об'єкт відноситься до того класу, до чиїх центрів він розташовується ближче.

Описаний трьохрівневий алгоритм класифікації називають *мережею з радіальними базисними функціями* або *RBF*-мережею (radial basis function network). Це один з різновидів нейронних мереж.

Навчання Rbf-мережі зводиться до відновлення щільностей класів $p_y(x)$ за допомогою ЕМ-алгоритму. Результатом навчання є центри μ_{yj} і дисперсії Σ_{yj} компонентів $j = 1,2,...,k_y$. Оцінюючи дисперсії, ми фактично підбираємо ваги ознак у метриках $\rho_{yj}(x, \mu_{yj})$ для кожного центру μ_{yj}. При використанні Алгоритму 2.3 для кожного класу визначається оптимальне число компонентів суміші.

Переваги ЕМ-алгоритму. У порівнянні з широко відомими градиентными методами настроювання нейронних мереж, ЕМ-алгоритм більш стійкий до шуму й швидше збігається. Крім того, він описує кожний клас як сукупність компонентів або кластерів, що дозволяє краще розуміти внутрішню структуру даних. Зокрема, центри сферичних гаусовських компонентів μ_{yj} можна інтерпретувати як віртуальні еталонні об'єкти, з якими порівнюються класифіковані об'єкти. У багатьох прикладних задачах віртуальним еталонам вдається підшукати змістовну інтерпретацію. Наприклад, у медичній диференціальній діагностиці це може бути певна $j-$та форма даного ($y-$го) захворювання. Інформація про те, що класифікований об'єкт близький саме до цього еталона, може бути корисною при прийнятті рішень.

Запитання до розділу 8

1. Яка геометрична інтерпретація нормальної щільності розподілу?
2. Коли застосовується відстань Махаланобіса, запишіть формулу відповідної метрики.
3. Сформулюйте суть принципу максимум правдоподібності, що таке оцінка максимуму правдоподібності?
4. Сформулюйте послідовність етапів підстановочного алгоритму та його недоліки.
5. Наведіть умови застосування та імовірність помилки дискримінанта Фішера.

Розділ 9
Логічні алгоритми класифікації

Розглянемо клас алгоритмів класифікації, в основі яких лежить принцип *індуктивного виводу логічних закономірностей* або *індукції правил* (rule induction, rule learning).

Нехай $\varphi : X \to \{0,1\}$ – деякий предикат, визначений на множині об'єктів X. Говорять, що предикат φ *виділяє* або *покриває*(cover) об'єкт x, якщо $\varphi(x) = 1$. Предикат називають *закономірністю,* якщо він виділяє досить багато об'єктів якогось одного класу c, і практично не виділяє об'єкти інших класів (більш строге визначення буде дано нижче).

Особливу цінність представляють закономірності, які описуються простою логічною формулою. Їх називають *правилами* (rules). Процес пошуку правил по вибірці називають *добуванням знань з даних*(knowledge discovery). До знань пред'являється особлива вимога – існування можливості їх інтерпретації, тобто вони мають бути зрозумілими людям. На практиці логічні закономірності часто шукають у вигляді кон'юнкцій невеликого числа елементарних висловлювань. Саме в такій формі люди звикли виражати свій життєвий і професійний досвід.

Приклад 1. (з області медицини). Вирішується питання про доцільність хірургічної операції. Закономірність: якщо вік пацієнта вище 60 років і раніше він переніс інфаркт, то операцію не робити – ризик негативного результату великий.

Приклад 2. (з області банківської діяльності). Вирішується питання про видачу кредиту. Закономірність: якщо позичальник указав в анкеті свій домашній телефон, і його зарплата перевищує $1000 на місяць, і сума кредиту не перевищує $10 000, то кредит можна видати – ризик неповернення малий.

Усяка закономірність класифікує лише деяку частину об'єктів. Об'єднавши певну кількість закономірностей у композицію, можна одержати алгоритм, здатний класифікувати будь-які об'єкти. *Логічними алгоритмами класифікації* будемо називати композиції закономірностей, які легко інтерпретуються. При побудові логічних алгоритмів виникають три основні питання:

- Який критерій інформативності, що дозволяє називати предикати закономірностями?
- Як будувати закономірності? Існують різні методи породження бінарних предикатів з різнотипної вихідної інформації.
- Як будувати алгоритми класифікації на основі закономірностей? Відомі найпоширеніші типи логічних алгоритмів: вирішальні списки, дерева рішень й ліси, голосування правил, алгоритми обчислення оцінок.

Нагадаємо основні позначення. Нехай є простір об'єктів X скінченна множина імен класів $Y = \{1,2,\ldots,M\}$ $Y=\{1,\ldots,M\}$.

Цільова залежність $y^* : X \to Y$ відома тільки на об'єктах навчальної

вибірки $X_m = \left(x_i, y_i\right)_{i=1}^m$, $y_i = y^*\left(x_i\right)$. Потрібно побудувати алгоритм класифікації $a : X \to Y$, що апроксимує y^* на всьому просторі X.

9.1. Поняття інформативності

9.1.1. Евристичне визначення інформативності

Інтуїтивно предикат φ тим більше інформативний, чим більше він виділяє об'єктів «свого» класу $c \in Y$ у порівнянні з об'єктами всіх інших «чужих» класів. Свої об'єкти називають також *позитивними*(positive), а чужі – *негативними* (negative). Введемо наступні позначення:

P_c – число об'єктів класу c вибірці X_m;

$p_c\left(\varphi\right)$ – з них число об'єктів, для яких виконується умова $\varphi\left(x\right) = 1$;

N_c – число об'єктів усіх інших класів $Y \setminus \left\{c\right\}$ у вибірці X_m;

$n_c\left(\varphi\right)$ – з них число об'єктів, для яких виконується умова $\varphi\left(x\right) = 1$.

Для стислості індекс c і аргумент $\left(\varphi\right)$ будемо іноді опускати. Передбачається, що $P \geq 1, N \geq 1, P + N = m$.

Інформативність предиката φ тим вище, чим більше об'єктів він виділяє, і чим менше серед них негативних. Виділення негативного об'єкта є помилкою предиката. Невиділення позитивного об'єкта також можна вважати помилкою, однак менш серйозною, оскільки від закономірностей не потрібно виділяти всі об'єкти. Об'єкт, що не виділений однією закономірністю, може бути виділений іншою.

Отже, задача побудови інформативного предиката φ зводиться до оптимізації по двом критеріям: $p_c\left(\varphi\right) \to \max$ і $n_c\left(\varphi\right) \to \min$. Найменш цікаві ті предикати, які або виділяють занадто мало об'єктів, або виділяють позитивні й негативні об'єкти приблизно в тій же пропорції, у якій вони були представлені у всій вибірці, $n : p \approx N : P$.

Введемо позначення E_c для частки негативних серед усіх об'єктів, що виділяються, і D_c для частки позитивних об'єктів:

$$E_c\left(\varphi, X_m\right) = \frac{n_c\left(\varphi\right)}{p_c\left(\varphi\right) + n_c\left(\varphi\right)}, \quad D_c\left(\varphi, X_m\right) = \frac{p_c\left(\varphi\right)}{m}$$

Визначення 1. Предикат $\varphi\left(x\right)$ будемо називати логічною ε, δ -закономірністю для класу $c \in Y$, якщо $E_c\left(\varphi, X_m\right) \leq \varepsilon$ і $D_c\left(\varphi, X_m\right) \geq \delta$ при заданих досить малому ε і досить великому δ з відрізка $\left[0,1\right]$.

Якщо $n_c(\varphi) = 0$, то закономірність φ називається чистою або несуперечливою. Якщо $n_c(\varphi) > 0$, то закономірність φ називається частковою.

У деяких випадках має сенс обмежуватися пошуком тільки чистих закономірностей. Зокрема, коли довжина вибірки мала або заздалегідь відомо, що дані практично не містять шуму. Такі прикладні задачі нерідко зустрічаються на практиці, наприклад, класифікація родовищ рідких корисних копалин за даними геологорозвідки, де дані коштують дорого, і тому, як правило, ретельно перевіряються.

Але частіше дані виявляються неповними й неточними. Такі багато медичні й економічні задачі (зокрема, задача про видачу кредитів).

Коли деяка частка помилок неминуча, цілком припустимо користуватися частковими закономірностями. Крім того, існують способи побудови коректних алгоритмів на основі часткових закономірностей, наприклад, зважене голосування. У всіх цих випадках доводиться відбирати предикати φ по двом критеріям $p_c(\varphi)$ і $n_c(\varphi)$ одночасно. Що краще: зменшення n на 1 чи збільшення p на 10? Для цілеспрямованого пошуку кращих закономірностей зручно мати критерій інформативності, здатний дати обґрунтовану відповідь на ці питання.

Виявляється, запропонувати адекватну згортку критеріїв n і p не так просто. У літературі наводять десятки різних критеріїв. На перший погляд, закономірності можна порівнювати по різниці $p - n$ або відношенню $\frac{p}{n}$. У таблиці1 зібрані контрприклади, які показують неадекватність «наївних» критеріїв інформативності. Жирним шрифтом виділені пари прикладів, у яких верхня закономірність із очевидністю цінніше нижньої, проте, критерій набуває на них рівні значення, або навіть більше значення на нижній закономірності.

p	n	$p-n$	$p-5n$	$\frac{p}{P}-\frac{n}{N}$	$\frac{p}{n+1}$	$\frac{p}{n+p}$	I_c	IGain_c	$\sqrt{p}-\sqrt{n}$
50	0	**50**	50	0.25	50	1	22.65	23.70	7.07
100	50	**50**	−150	0	1.96	0.67	2.33	1.98	2.93
50	9	41	**5**	0.16	**5**	**0.85**	7.87	7.94	4.07
5	0	5	**5**	0.03	**5**	**1**	2.04	3.04	2.24
100	0	**100**	100	**0.5**	100	1	52.18	53.32	10.0
140	20	**120**	40	**0.5**	6.67	0.88	37.09	37.03	7.36

Таблиця 1. Критерії інформативності предикатів при P=200 і N=100

9.1.2. Статистичне визначення інформативності

Адекватну скалярну характеристику інформативності дає техніка перевірки статистичних гіпотез.

Нехай X – імовірнісний простір, вибірка X_m – *проста,* тобто випадкова,

незалежна, однаково розподілена (independent, identically distributed – i.i.d.), $y^*\left(x\right)$ і $\varphi\left(x\right)$ – випадкові величини. Припустимо, справедлива гіпотеза про незалежність подій $\left\{x : y^*\left(x\right) = c\right\}$ і $\left\{x : \varphi\left(x\right) = 1\right\}$. Тоді ймовірність реалізації пари $\left(p,n\right)$ підкоряється гіпергеометричному розподілу:

$$h_{P,N}\left(p,n\right) = \frac{C_P^p C_N^n}{C_{P+N}^{p+n}},\ 0 \le p \le P,\ 0 \le n \le N \tag{9.1}$$

Де $C_m^k = \frac{m!}{k!\left(m-k\right)!}$ – біноміальні коефіцієнти, $0 \le k \le m$.

Якщо ймовірність (1) мала, і проте пара $\left(p,n\right)$ реалізувалася, то гіпотеза про незалежність повинна бути відкинута. Чим менше значення ймовірності, тим більше значимим є зв'язок між y^* і φ. Можно сказати і так: якщо реализувалась малоймовірна подія, то, скоріше всього, це не випадково, а закономірно.

Визначення 2. Інформативність предиката $\varphi\left(x\right)$ відносно класу $c \in Y$ по вибірці X_m це:

$$I_c\left(\varphi, X_m\right) = -\ln h_{P_c,N_c}\left(p_c\left(\varphi\right), n_c\left(\varphi\right)\right)$$

Предикат $\varphi\left(x\right)$ будемо називати статистичною закономірністю для класу c, якщо $I_c\left(\varphi, X_m\right) \ge I_0$ при заданном досить великому I_0. Поріг інформативності I_0 вибирають так, щоб йому відповідало достатньо мале значення ймовірності, яке називають *рівнем значимості*. Для кожної задачі його необхідно вибирати індивідуально. Описанный критерій застосовується в статистиці для перевірки гіпотез про незалежність подій і називається точним тестом Фішера (Fisher's exact test, FET). Точним – тому що відомі і інші критерії незалежності, але вони є лише асимптотичними наближеннями гіпергеометричного розподілу. Перевага асимптотичних критеріїв в тому, що вони розраховуються за простішими формулами. Але при цьому вони допускають значну помилку на малих вибірках (до декількох десятків об'єктів). Цей недолік може виявитися дуже важливим, коли інформативність предикатів оцінюється на малих вибірках або на частинах вибірки. Зокрема, це відбувається при синтезі списків рішень і дерев рішень.

Ефективне обчислення інформативності. Зауважимо, що обчислення логарифма $h_{P,N}\left(p,n\right)$ зводиться до додавання 9 чисел виду $\ln\left(k!\right)$. Якщо довжина вибірки m фіксована, можна заздалегідь скласти таблицю логарифмів факторіалів $L_k = \ln k!$ за рекурентною формулою $L_1 = 0;\ L_k = L_{k-1} + \ln k$ для всіх $k = 2,\ldots,m$.

Інший спосіб обчислення $\ln\left(k!\right)$ пов'язаний із застосуванням формули Стірлінга:

$$\ln k! \simeq k \cdot \ln k - k + \frac{1}{2}\ln 2k\pi + \frac{1}{12k} - \frac{1}{360k^2} \qquad (9.2)$$

Ця формула дає досить точне наближення. При $k = 2$ відмінність з'являється тільки в п'ятій значущій цифрі, при $k = 10$ – в десятій, і з ростом n точність росте. Більш точне значення дає формула Раманудджана:

$$\log k! \approx k \log k - k + \frac{1}{6}\ln\left(k\left(1 + 4k\left(1 + 2k\right)\right)\right) + \frac{1}{2}\ln \pi$$

Обидві формули можна застосовувати навіть коли P, N, p, n не є цілими числами. Це дозволяє узагальнити поняття інформативності на той випадок, коли об'єкти не рівнозначні.

Зіставлення двох критеріїв інформативності.

Який з критеріїв краще – евристичний або статистичний? При розумних комбінаціях параметрів ε та I_0 евристичний критерій практично завжди виявляється суворіше статистичного, що показано на рис. 9.1

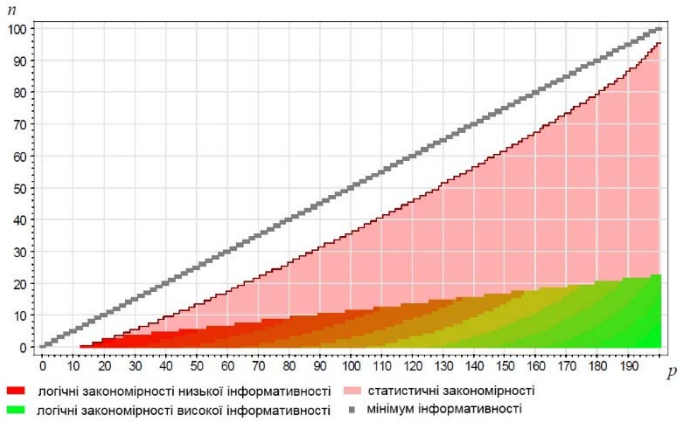


Рис. 9.1. Області логічних $\varepsilon, \delta -$ закономірностей (при $\varepsilon = 0.$) і статистичних закономірностей (при $I_0 = 5$) в координатах $\left(p, n\right)$ при $P = 200, N = 100$

Є досить велика область статистичних закономірностей, для яких імовірність випадкової реалізації вкрай низька, у той же час, вони допускають занадто багато помилок і не є логічними закономірностями в сенсі ε, δ - критерію.

Насправді корисні обидва визначення. Залежно від ситуації виявляється більш доцільним застосовувати або евристичний, або статистичний критерій інформативності.

9.1.3 Ентропійне визначення інформативності

Ще один спосіб визначення інформативності випливає з теорії інформації. Нагадаємо, що якщо є два результати ω_0, ω_1 з імовірністю q_0 і

$q_1 = 1 - q_0$, то *кількість інформації*, яка пов'язана з результатом ω_i , за визначенням дорівнює $-\log_2 q_i$. Це математичне очікування числа біт, необхідних для запису інформації про реалізацію результатів ω_i при використанні оптимального (найбільш економного) кодування. *Ентропія* визначається як математичне очікування кількості інформації:

$$H\left(q_0, q_1\right) = -q_0 \log_2 q_0 - q_1 \log_2 q_1$$

Будемо вважати появу об'єкта класу c результатом ω_0, а появу об'єкта будь-якого іншого класу – результатом $\omega 1$. Тоді, підставляючи замість ймовірностей частоти, можна оцінити ентропію вибірки X_m:

$$\widehat{H}\left(P, N\right) = H\left(\frac{P}{P+N}, \frac{N}{P+N}\right)$$

Припустимо, стало відомо, що предикат φ виділив p об'єктів з P, що належать класу c, і n об'єктів з N, які не належать класу c. Тоді ентропія вибірки $\left\{x \in X_m \middle| \varphi\left(x\right) = 1\right\}$ є $\widehat{H}\left(p, n\right)$. Імовірність появи об'єкта з цієї вибірки оцінюється як $\frac{p+n}{P+N}$. Аналогічно, ентропія вибірки $\left\{x \in X_m \middle| \varphi\left(x\right) = 0\right\}$ є $\widehat{H}\left(P-p, N-n\right)$, а ймовірність появи об'єкта з неї оцінюється як $\frac{P-p+N-n}{P+N}$. Таким чином, ентропія всієї вибірки після отримання інформації φ дорівнює

$$\widehat{H}_{\varphi}\left(P, N, p, n\right) = \frac{p+n}{P+N}\widehat{H}\left(p, n\right) + \frac{P+N-p-n}{P+N}\widehat{H}\left(P-p, N-n\right)$$

В результаті зменшення ентропії становить

$$IGain_c\left(\varphi, X_m\right) = \widehat{H}\left(P, N\right) - \widehat{H}_{\varphi}\left(P, N, p, n\right).$$

Це і є інформаційний виграш (information gain) - кількість інформації про початковий розподілі вибірки на два класи «с» і «не с», яке міститься в предикаті φ. Таким чином, з'являється ще одне, альтернативне, визначення закономірності

Визначення. Предикат φ є закономірністю за ентропійним критерієм інформативності, якщо $IGain_c\left(\varphi, X_m\right) > G_0$ при деякому досить великому G_0.

Теорема. Ентропійний критерій $IGain_c$ асимптотично еквівалентний статистичному I_c.

Незважаючи на асимптотичну еквівалентність, значення I_c і $IGain_c$ можуть помітно відрізнятися при малих *n* або *p*. Згідно таблиці 1, критерій

$IGain_c$ вважає, що «малопотужна» закономірність $(n, p) = (0,5)$ краще, ніж «повна відсутність закономірності» (50, 100), тоді як точний тест I_c показує протилежне. Іншими словами, критерій IGain завищує інформативність малопотужних закономірностей. Ситуації малих *n* або *p* зовсім не екзотичні, вони регулярно виникають при побудові списків і дерев рішень. Точний критерій може давати в цих ситуаціях відчутні переваги. Однак на практиці частіше використовується ентропійний критерій, оскільки він простіше обчислюється.

9.1.4. Багатокласова інформативність

Коли число класів перевищує 3, доводиться оцінювати інформативність не тільки таких предикатів, які відокремлюють один клас від інших, але і таких, які відокремлюють деяку групу класів від інших.

Статистичний критерій інформативності узагальнює статистичне означення на випадок довільної кількості класів $Y = \{1,\ldots,M\}$:

$$I(\beta, U) = -\ln \frac{C_{P_1}^{p_1} \ldots C_{P_M}^{p_M}}{C_m^p},$$

де

P_c – кількість об'єктів класу c у вибірці U з них p_c – кількість об'єктів, які виділяються предикатом β, $p = p_1 + \ldots + p_M$

Ентропійний критерій для випадку великої кількості класів є асимптотичним наближення статистичного критерію:

$$I(\beta, U) = \sum_{c \in Y} h\left(\frac{P_c}{m}\right) - \frac{p}{m} \sum_{c \in Y} h\left(\frac{p_c}{p}\right) - \frac{m-p}{m} \sum_{c \in Y} h\left(\frac{P_c - p_c}{m-p}\right),$$

де введена функція $h(z) \equiv -z \log_2 x$

9.1.5. Зважена інформативність

Ціна помилки на різних об'єктах може бути різною. Наприклад, вона може відрізнятися для різних класів: при видачі кредитів «пропуск цілі» (т. е. ненадійного позичальника) обходиться банку істотно дорожче, ніж «хибна тревога» (відмова хорошому позичальникові). Навчальні об'єкти з класу «погані позичальники» повинні враховуватися з більшою вагою при пошуку логічних закономірностей. Нехай заданий вектор невід'ємних ваг об'єктів $w = (w_i)_{i=1}^m$ з умовою нормування $\sum_{i=1}^m w_i = m$. Визначимо величини, аналогічні $P, N, p(\varphi), n(\varphi)$:

$$P_c^w = \sum_{i=1}^{m} w_i \left[y_i = c \right];\ p_c^w(\varphi) = \sum_{i=1}^{m} w_i \left[y_i = c \right] \left[\varphi\left(x_i = 1 \right) \right];$$
$$N_c^w = \sum_{i=1}^{m} w_i \left[y_i \neq c \right];\ n_c^w(\varphi) = \sum_{i=1}^{m} w_i \left[y_i \neq c \right] \left[\varphi\left(x_i = 1 \right) \right]; \quad (9.3)$$

Визначення логічної ε,δ-закономірності, статистична і ентропійна інформативність легко узагальнюються на цей випадок.

Розумний компроміс між точністю наближення і швидкістю обчислень дає лінійна апроксимація:

9.2. Методи пошуку інформативних закономірностей

Отже, інформативні закономірності служать початковою сировиною для побудови логічних алгоритмів класифікації. Виникає питання: у якій множині предикатів слід шукати інформативні закономірності? Цю множину називають ще *простором пошуку*(search space).

Найбільш простий той випадок, коли всі початкові ознаки є бінарними, $f_j : X \to \{0,1\}, j = 1,2,\ldots,n$. Тоді простір пошуку утворюється самими ознаками й усілякими булевими функціями, які із цих ознак можна побудувати. При цьому досить будувати тільки диз'юнктивні нормальні форми, оскільки будь-яку булеву функцію можна записати у вигляді ДНФ

Більше того, у якості закономірностей можна брати тільки кон'юнкції ознак і їх заперечень, а диз'юнкцію реалізувати як коригувальну операцію, наприклад, як голосування по більшості або старшинству.

Більш складними є ті випадки, коли об'єкти описуються різнотипними ознаками: номінальними, порядковими, кількісними, і т. ін., або коли об'єкти представляють більш складні структури: тексти, зображення або сигнали. Подібного роду ситуації частіше виникають на практиці, ніж «рафінований» бінарний випадок. Тоді простором пошуку стають усілякі бінарні функції від початкових ознак. Процес побудови таких функцій називають *бінаризацією* початкової інформації. Як правило, припустимих способів бінаризації виявляється настільки багато, що виникає нове питання: як скоротити простір пошуку та уникнути оцінювання інформативності для величезного числа свідомо неінформативних або майже однакових предикатів.

9.2.1.Бінаризація кількісних ознак

Довільна ознака $f : X \to D_f$ породжує сімейство предикатів, що перевіряють попадання значення $f(x)$ у певні підмножини множини D_f . Нижче перелічуються найбільш типові конструкції такого виду.

- Якщо f – номінальна ознака:

$$\beta(x) = \left[f(x) = d \right],\ d \in D_f;$$
$$\beta(x) = \left[f(x) \in D' \right], D' \subset D_f.$$

- Якщо f – порядкова або кількісна ознака:

$$\beta\left(x\right)=\left[f\left(x\right)\le d\right],\ d\in D_f;$$
$$\beta\left(x\right)=\left[d\le f\left(x\right)\le d'\right],d,d'\in D_f,\ d<d'.$$

У випадку кількісних ознак $f:X\to R$ має сенс брати тільки такі значення порогів d, які по-різному розділяють вибірку X_m. Якщо виключити тривіальні розбивки, що переводять $\beta\left(x\right)$ в 0 або 1 на всій вибірці, то таких значень виявиться не більше, ніж $m-1$. Наприклад, можна пороги виду:

$$d_i=\frac{f^{(i)}+f^{(i+1)}}{2},\ \ f^{(i)}\ne f^{(i+1)},\ i=1,2,\ldots,m-1 \qquad (9.4)$$

де $f^{(1)}\le\ldots\le f^{(m)}$ – послідовність значень ознаки f на об'єктах вибірки $f\left(x_1\right),\ldots,f\left(x_m\right)$, упорядкована по зростанню (варіаційний ряд), як показано на.рис.3.

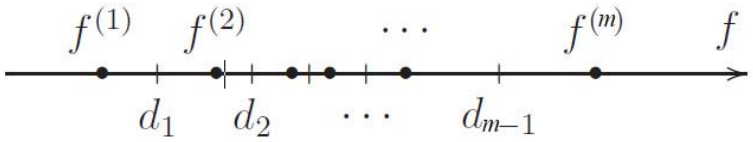


Рис. 9.2. Варіаційний ряд значень ознаки $f\left(x\right)$ та пороги d_i

Описані способи дозволяють одержати величезну кількість предикатів. Якщо надалі вони будуть використовуватися для синтезу кон'юнкцій, то для скорочення перебору має сенс відразу відібрати з них найбільш інформативні. У випадку порядкових і кількісних ознак дана задача вирішується шляхом оптимальної розбивки діапазону значень ознаки на зони.

Розбивка діапазону значень ознаки на інформативні зони

Нехай $f:X\to R$ - числова ознака, $d_1,\ldots,d_r$ – зростаюча послідовність порогів. *Зонами* значень ознаки f будемо називати предикати виду

$$\xi_0\left(x\right)=\left[f\left(x\right)<d_1\right];$$
$$\xi_s\left(x\right)=\left[d_s\le f\left(x\right)<d_{s+1}\right],\ \ s=1,\ldots,r-1;$$
$$\xi_r\left(x\right)=\left[d_r\le f\left(x\right)\right].$$

Алгоритм 1 починає з розбивки на «дрібні зони».

Алгоритм 1. Жадібний алгоритм злиття зон
Вхідні дані:

$f\left(x\right)$ – ознака;

$c\in Y$ – виділений клас;

$X_m = \left\{\left(x_i, y_i\right)\right\}_{i=1}^{m}$ – вибірка, упорядкована за зростанням $f\left(x_i\right)$;

r – бажана кількість зон;

δ_0 – поріг злиття зон (за замовчуванням $\delta_0 = 0$).

Вихідні дані:

$D = \left\{d_1, d_2, \ldots, d_r\right\}$ – строго зростаюча послідовність порогів;

Хід роботи алгоритму:

1: $D := \varnothing$;
2: для всіх $i = 2, \ldots, m$
3: **якщо** $f\left(x_{i-1}\right) \neq f\left(x_i\right)$ і $\left[y_{i-1} = c\right] \neq \left[y_i = c\right]$ **то**
4: додати новий поріг $\frac{1}{2}\left(f\left(x_{i-1}\right) + f\left(x_i\right)\right)$ у кінець послідовності D;
5: повторювати
6: **для всіх** $d_i \in D,\ i = 1, \ldots, \left|D\right| - 1$
7: обчислити виграш від злиття трійки сусідніх зон $\xi_{i-1}, \xi_i, \xi_{i+1}$:

$$\delta I_i := I_c\left(\xi_{i-1} \vee \xi_i \vee \xi_{i+1}\right) - \max\left\{I_c\left(\xi_{i-1}\right), I_c\left(\xi_i\right), I_c\left(\xi_{i+1}\right)\right\};$$

8: знайти трійку зон, для якої злиття найбільш вигідне:
$i := \arg\max_s \delta I_s$;
9: якщо $\delta I_i > \delta_0$ **то**
10: злити зони $\xi_{i-1}, \xi_i, \xi_{i+1}$ видаливши пороги d_i і d_{i+1} з послідовності D;
11: поки $\left|D\right| > r + 1$.

Пороги визначаються за формулою (9.4) і проходять між усіма парами точок, x_{i-1}, x_i рівно одна з яких належить класу c (кроки 2–4). Неважко показати, що розміщення порогів між точками класу c або між точками не класу c призведе тільки до зменшення інформативності зон. Отже, початкова розбивка складається із зон, що чергуються, «тільки c - тільки не c», як показано на рис. 9.3.

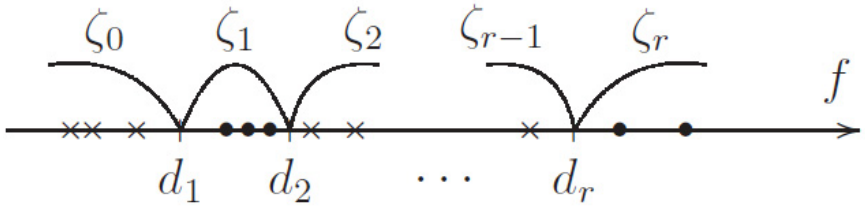


Рис. 9.3. Початкове розбиття на зони позитивних $\left(\times\right)$ і негативних $\left(\bullet\right)$ об'єктів.

Далі зони укрупнюються шляхом злиття трійок сусідніх зон. Саме трійок – злиття пара приводить до порушення чергування «c - не c», у результаті деякі «дрібні зони» можуть так і залишитися незлитими. Зони

зливаються доти, поки інформативність деякої злитої зони $\xi_{i-1} \vee \xi_i \vee \xi_{i+1}$ перевищує інформативність початкових зон $\xi_{i-1}, \xi_i, \xi_{i+1}$, або поки не буде отримана задана кількість зон r. Щораз вибирається та трійка, при злитті якої досягається максимальний виграш інформативності.

Цей алгоритм має асимптотичну складність $O\left(m^2\right)$. Його можна помітно прискорити, якщо на кожній ітерації зливати не одну трійку зон, а τm трійок з досить великим виграшом δI_i, за умови, що вони не перекриваються. Число τ – ще один параметр алгоритму, $0 < \tau < 0.5$. У цьому випадку асимптотична складність становить $O\left(m/\sqrt{\tau}\right)$.

9.2.2. Пошук закономірностей у формі кон'юнкцій

Нехай $B-$ скінченна множина предикатів, які ми будемо називати *елементарними.* Задамо множину кон'юнкцій з обмеженим числом термів з B:

$$K_{Rg}\left[B\right] = \left\{\varphi\left(x\right) = \beta_1\left(x\right) \wedge \ldots \wedge \beta_k\left(x\right) \middle| \beta_1, \beta_2, \ldots, \beta_k \in B,\ k \le Rg\right\}$$

Число термів k в кон'юнкції називається її *рангом.* Кон'юнкції невеликого рангу мають важливу перевагу – вони мають вигляд звичних для людини логічних висловлень і легко піддаються змістовній інтерпретації.

Максимальний ранг кон'юнкцій Rg зазвичай встановлюють від 3 до 7 міркувань інтерпретованості, оскільки майже неможливо відстежити за змістом висловлювань, що містять занадто велику кількість умов.

Пошук найбільш інформативних кон'юнкцій у загальному випадку вимагає повного перебору. Число припустимих кон'юнкцій є $O\left(\left|B\right|^{Rg}\right)$, і може виявитися настільки великим, що повний перебір стане практично нездійсненний. Представимо цілком реалістичну ситуацію: є 100 числових ознак; діапазон значень кожної ознаки розбитий на 10 зон, тобто породжує 10 елементарних предикатів. Тоді число кон'юнкцій рангу Rg дорівнює $C_{100}^{Rg}10^{Rg}$. Уже при $Rg = 5$ ця величина має порядок 10^{13}, що виключає можливість повного перебору на сучасних комп'ютерах.

На практиці використовують різні евристики для скороченого цілеспрямованого пошуку кон'юнкцій, близьких до оптимальних. Ідея всіх цих методів полягає в тому, щоб не перебирати величезна кількість свідомо неінформативних предикатів. Почнемо з найбільш простого методу.

«Градієнтний» алгоритм синтезу кон'юнкцій.

Поставимо кожній кон'юнкції φ у відповідність її *окіл* – множину кон'юнкцій $V\left(\varphi\right), які$ одержують з φ шляхом елементарних модифікацій: додаванням, вилученням або модифікацією одного з термів кон'юнкції.

Починаючи із заданої кон'юнкції φ_0 (наприклад, порожньої), будується послідовність кон'юнкцій $\varphi_0, \varphi_1, \ldots, \varphi_t, \ldots,$ у якій кожну наступну кон'юнкцію φ_t вибирають з околу попередньої $V_t\left(\varphi_{t-1}\right)$ за критерієм максимуму інформативності (крок 3).

Найбільш перспективними з погляду подальших модифікацій вважаються кон'юнкції з максимальною інформативністю. Однак «перспективна» – не означає краща, тому що кон'юнкції з високою інформативністю можуть допускати багато помилок. Тому на кожному кроці t виділяють «найкращу» кон'юнкцію, що задовольняє додатковій умові $E_c\left(\varphi_t^*\right) < \varepsilon$, і в загальному випадку не співпадаюча з найбільш перспективною φ_t.

Оскільки множина кон'юнкцій, що по-різному класифікують вибірку, скінченна, ітераційний процес сходиться за скінченну кількість кроків до деякої кон'юнкції, яку неможливо «локально покращити». Зрозуміло, ні про який градієнт у буквальнім значенні слова не ідеться. Мається на увазі, що даний алгоритм, за аналогією із градієнтним спуском, вибирає на кожній ітерації найкращу з найближчих точок простору пошуку.

Критерій інформативності I і функція околу V, загалом кажучи, є параметрами алгоритму. При формуванні околу V можна застосовувати різні евристики:

— обмежувати максимальний ранг кон'юнкцій Rg ;

— відбирати з околу тільки ε, δ-закономірності;

— дозволяти тільки додавання термів;

— чергувати серії додавання термів із серіями видалень;

— дозволяти модифікацію декількох термів одночасно;

— дозволяти зміну порогу α, але не ознаки f, у термах $\left[f\left(x\right) < a\right]$;

— варіювати пороги a тільки в межах сусідніх зон.

Варіюючи параметри Алгоритму 2, можна одержувати різні процедури пошуку або поліпшення інформативних кон'юнкцій.

Алгоритм 2. «Градієнтний» алгоритм синтезу кон'юнкції
Вхідні дані:

X_m – навчальна вибірка;

φ_0 – початкове наближення;

$c \in Y$ – клас, для якого будують кон'юнкцію;

$t_{\max}$ – максимальне число ітерацій;

d- параметр критерію останову;

ε – поріг частки помилок для відбору кон'юнкцій;

Вихідні дані:

кон'юнкція φ;

Хід роботи алгоритму:

1: $I^* := I_c\left(\varphi_0, X_m\right)$;

2: для всіх $t = 1,2,\ldots,t_{\max}$

3: поточна множина кон'юнкцій, що перебираються: $V_t := V\left(\varphi_{t-1}\right)$;

4: найбільш перспективна кон'юнкція: $\varphi_t := \arg\max_{\varphi \in V_t} I_c\left(\varphi, X_m\right)$;

5: найкраща кон'юнкція на t-й ітерації: $\varphi_t^* := \arg\max_{\varphi \in V_t : E_c(\varphi) < \varepsilon} I_c\left(\varphi, X_m\right)$;

6: **якщо** $I_c\left(\varphi_t^*\right) > I^*$ **то**

запам'ятати, на якій ітерації була отримана найкраща кон'юнкція:

$t^* := t; \varphi^* := \varphi_t^*; I^* := I_c\left(\varphi^*\right)$

7: **якщо** $t - t^* > d$ (кон'юнкція не покращилася за останні d кроків) **то**

8: **вихід**;

9: повернути φ^*;

Розглянемо більш докладно варіанти цього алгоритму.

Жадібний алгоритм синтезу кон'юнкції використовує тільки операцію додавання термів. Початковим наближенням є порожня кон'юнкція (без термів). Недолік жадібної стратегії в тому, що вона може вести убік від глобального максимуму інформативності. Терм, знайдений на k-*му* кроці, перестає бути оптимальним після додавання наступних термів. Проте, у ряді практичних задач ця проста евристика демонструє здатність знаходити непогані закономірності.

Стохастичний локальний пошук (stochastic local search, SLS) також починає з порожньої кон'юнкції, але використовує повний набір можливих модифікацій. Це є перевагою в порівнянні з жадібним алгоритмом, тому що з'являється можливість видаляти й замінювати неоптимальні терми. З іншого боку, потужність околу $\left|V\left(\varphi\right)\right|$ може виявитися настільки великою, що перебір усіх припустимих модифікацій стане нерентабельним. Тому в SLS будують не весь окіл, а тільки деяку його випадкову підмножину. Максимальна припустима потужність цієї підмножини задається як додатковий параметр алгоритму $V_{\max}$.

Для поліпшення кон'юнкції φ, побудованої жадібним нарощуванням або SLS, до неї застосовують методи «фінального шліфування» – стабілізацію й редукцію.

Процедура стабілізації намагається поліпшити кон'юнкцію φ, видаляючи або заміняючи по одному терму. На відміну від SLS,

перебираються всі можливі видалення й заміни. Модифікації проводяться доти, поки зростає інформативність кон'юнкції $I_c\left(\varphi, X_m\right)$. Стабілізація підвищує стійкість алгоритму щодо малих варіацій навчальної вибірки або інших умов навчання (наприклад, генератора псевдовипадкової послідовності в SLS). У результаті стабілізації знайдені кон'юнкції часто сходяться до тих самих локальних максимумів інформативності. Звичайно це позитивно позначається на інтерпретації правил. Експерт більше довіряє правилу, коли бачить, що спроби скорегувати його «вручну» приводять тільки до його погіршення.

Процедура редукції відрізняється тим, що терми тільки видаляються, а інформативність обчислюється по незалежній *контрольній вибірці* X_k, складеній з об'єктів, що не брали участь у побудові кон'юнкції φ. Контрольну вибірку формують до початку навчання, виділяючи з масиву початкових даних близько 30% об'єктів, як правило, випадковим образом. При цьому об'єкти різних класів розподіляються в тій же пропорції, що й у всій вибірці (цей принцип відбору називається *стратифікацією* вибірки). Сенс редукції в тому, щоб перевірити, чи не є знайдена кон'юнкція надлишково складною, і або спростити її, або визнати зовсім невдалою. Спрощення підвищує узагальнення логічного правила, оскільки множина об'єктів, що виділяються за цим правилом розширюється. Недолік редукції в тому, що вона вимагає залишити значну частку даних для контролю, зменшивши представимість навчальної вибірки. Однак при розумному виборі співвідношення $m : k$ пошук правил по X_m з наступною редукцією по X_k може давати кращі результати, ніж пошук по $X_m \cup X_k$ без редукції.

Генетичний (емерджентний) алгоритм синтезу кон'юнкцій

(Genetic algorithm, GA) можна розглядати як подальше вдосконалення SLS на основі ідей дарвінівської еволюції. Головна відмінність GA від SLS в тому, що на кожному кроці відбирається не одна найкраща кон'юнкція, а ціла множина кращих кон'юнкцій, яку називають *популяцією.* З них породжується велика кількість кон'юнкцій – *нащадків* за допомогою двох *генетичних операцій* – схрещування й мутації. *Схрещування* (crossing over) – це створення нової кон'юнкції шляхом обміну термами між двома членами популяції. У ролі *мутацій* виступають уже знайомі операції додавання, заміщення й видалення термів. Таким способом можна одержати величезну кількість нащадків, але на практиці будують лише обмежене число, не більш T_1 нащадків шляхом випадкових схрещувань і мутацій. Потім проводиться *природний добір,* у результаті якого в наступне покоління переходять не більш T_0 найбільш інформативних нащадків. Звичайно беруть $T_0 << T_1$.

Емерджентні алгоритми відрізняються більшою різноманітністю всіляких евристик, запозичених безпосередньо з живої природи. Наприклад, в емерджентний алгоритм легко вмонтувати процедуру селекції або *штучного відбору*, породжуючи нащадків тільки від найкращих кон'юнкцій, або задаючи

розподіл імовірностей на популяції так, щоб імовірність стати батьком збільшувалася з ростом інформативності. Можна організовувати кілька популяцій, які паралельно розвиваються (острівна модель еволюції), щоб збільшити різноманітність кон'юнкцій.

9.2.3. Форми закономірностей

Кон'юнкції над предикатами виду $\beta(x) = \left[d \le f(x) \le d'\right]$ описують області простору X, що мають форму гіперпаралелепіпедів. На практиці застосовуються і інші форми закономірностей, наприклад, кулі або гіперплощини. Вибір форми визначається особливостями конкретної задачі. У загальному випадку до форми пред'являються дві вимоги.

• *Інтерпретованість*. Умова $\varphi(x) = 1$ повинна бути виражене природною мовою у формі, зрозумілій експерту. Для цього, зокрема, закономірність $\varphi(x)$ повинна залежати від невеликої кількості ознак $\omega \subseteq \{1,\ldots,n\}$. Зазвичай $|\omega|$ обмежують зверху числом від 2 до 7. Знайдені поєднання ознак ω можуть або підтверджувати існуючі уявлення про взаємозв'язки між ознаками, або вказувати на існування раніше невідомих взаємозв'язків, і тоді можна говорити про отримання нових знань з даних (knowledge discovery). Деякі зі знайдених поєднань ознак ω можуть виявитися такими, шо не інтерпретуються зі змістовної точки зору; такі закономірності відкидаються експертами як «помилкові»

• *Ефективність пошуку.* Повинні існувати ефективні методи пошуку закономірностей по кінцевій вибірці U в рамках обраного сімейства предикатів Φ: $I(\varphi, U) \to \max\limits_{\varphi \in \Phi}$. Як правило, найбільш трудомістким є пошук інформативних наборів ознак $\omega \subseteq \{1,\ldots,n\}$.

Параметричне сімейство куль:

$$\varphi_c(x) = \left[\rho(x, x_0) \le r_0\right] \tag{9.5}$$

де $\rho(x, x_0)-$ метрика в просторі об'єктів X. Параметрами є центр кулі x_0 , його радіус r_0, і сама функція відстані ρ. Як центри беруть або навчальні об'єкти, або центри локальних згущень, знайдені будь-яким алгоритмом кластеризації. Радіуси куль можна підбирати аналогічно виділенню зон – формується множина $m-1$ порогових значень таких, що по-різному поділяють вибірку, і з них вибирається таке значення радіусу, при якому досягається максимум інформативності предиката (9.5). Функція відстані ρ, як правило, задається у вигляді лінійної комбінації елементарних метрик по деякому набору ознак $\omega \subseteq \{1,\ldots,n\}$:

$$\rho_\omega\left(x,x'\right)=\sum_{j\in\omega}\alpha_j\left|f_j\left(x\right)-f_j\left(x'\right)\right|,$$

де набір ω і коефіцієнти α_j передбачається оптимізувати по вибірці.

Виділення об'єкта x кулястою закономірністю $\varphi\left(x\right)$ легко інтерпретується в термінах схожості: «об'єкт x відноситься до класу c тому, що він близький до еталонного об'єкту x_0, який лежить в класі c, за сукупністю ознак ω». Такого роду пояснення добре сприймаються в тих предметних областях, де поширений прецедентний стиль мислення - в медицині, геології, соціології, юриспруденції, і ін.

Параметричне сімейство гіперплощин:

$$\varphi_c\left(x\right)=\left[\left\langle x,\alpha\right\rangle\le\alpha_0\right] \tag{9.6}$$

де $\left\langle x,\alpha\right\rangle$ – скалярний добуток в просторі об'єктів X. Параметрами є вектор нормалі α і зміщення α_0 розділювальної гіперплощини. Максимізація інформативності зводиться до підбору таких α і α_0 при яких з одного боку гіперплощини лежать об'єкти переважно одного класу. Закономірності виду (6) добре інтерпретуються, коли серед компонент вектора α мало ненульових, і розділяюча гіперплощина проводиться в підпросторі невеликої розмірності:

$$\left\langle x,\alpha\right\rangle=\sum_{j\in\omega}\alpha_j f_j\left(x\right)$$

При цьому бажано, щоб ознаки мали наступну властивість монотонності: «чим вище значення ознаки $f_j\left(x\right)$, тим швидше об'єкт x відноситься до класу c». Тоді коефіцієнти α_j інтерпретуються як ступінь важливості j-ї ознаки.

Параметричне сімейство областей, описуваних ядром:

$$\varphi\left(x\right)=\left[K\left(x,x_0\right)\le K_0\right] \tag{9.7}$$

де $K: X\times X\to R-$ функція ядра (kernel function),
$x_0\in X,\ K_0\in R-$ параметри предиката.

Кулі, гіперплощини і гіперпаралелепіпеди є окремими випадками ядер. Функція K може вільно вибиратися, виходячи з будь-яких апріорних міркувань. Якщо таких в конкретному завданні немає, єдине, що залишається – організувати банк ядер, що містить «потенціально корисні» функції, в тому числі метрики і скалярні добутки. Тоді процедура пошуку інформативних закономірностей повинна включати в себе перебір ядер.

Знову-таки, ядра, що залежать тільки від невеликого числа ознак, простіше піддаються змістовній інтерпретації, але для їх пошуку доводиться організовувати перебір підмножин ознак.

Запитання до розділу 9

1. Сформулюйте евристичне визначення інформативності.

2. Надайте статистичне визначення інформативності.

3. В чому полягає ентропійне визначення інформативності?

4. Опишіть способи бінаризації кількісних ознак.

5. Які ви знаєте алгоритми пошуку закономірностей у формі кон’юнкцій?

Розділ 10
Списки та дерева ухвалення рішень

10.1. Списки ухвалення рішень

Список ухвалення рішень (decision list, DL) – це найбільш простий логічний алгоритм, як по своїй структурі, так і по способу побудови.

Визначення 1. Список ухвалення рішень це алгоритм класифікації $a: X \to Y$, який задається набором закономірностей $\varphi_1(x),\ldots,\varphi_T(x)$, приписаних до класів $c_1,c_2,\ldots,c_T \in Y$ відповідно, і обчислюється згідно Алгоритму 1.

Алгоритм 1. Класифікація об'єкта $x \in X$ списком ухвалення рішень

1: **для всіх** $t = 1,2,\ldots,T$

2: **якщо** $\varphi_t(x) = 1$ **то**

3: **повернути** c_t;

4: **повернути** c_0.

«Особлива відповідь» c_0 означає відмову алгоритму від класифікації об'єкта x. Зазвичай такі об'єкти приписують класу, що має мінімальну ціну помилки. Наприклад, в завданню видачі кредитів відмова алгоритму призведе до більш обережного рішення «не видавати». У задачі розпізнавання спаму більш обережним буде рішення «не спам».

Зауваження. Сусідні правила в списку φ_{t-1},φ_t можна переставляти місцями, якщо тільки вони приписані до одного класу, $c_{t-1} = c_t$. У загальному випадку перестановка правил в списку змінює алгоритм.

Зауваження. Список ухвалення рішень для закономірностей є окремим випадком алгоритмічної композиції з голосуванням по старшинству. Відмінності хіба що термінологічні: тепер базові алгоритми називаються закономірностями або правилами. Розглянемо жадібний алгоритм побудови списку ухвалення рішень.

10.1.1. Жадібний алгоритм побудови списку ухвалення рішень

Алгоритм 2 на кожній ітерації будує рівно одне правило φ_t, що виділяє максимальне число об'єктів деякого класу c_t і мінімальне число об'єктів всіх інших класів.

Алгоритм 2. Жадібний алгоритм побудови списку ухвалення рішень

Вхідні дані:

X_m -навчальна вибірка;

Ф – сімейство предикатів, з якого вибираються закономірності;

$T_{\max}$ – максимальне припустиме число правил у списку;

$I_{\min}$ – мінімальна припустима інформативність правил у списку;

$E_{\max}$ – максимальна припустима частка помилок на навчальній вибірці;

m_0 – максимальне припустиме число відмов;

Вихідні дані:

список ухвалення рішень $\left(\varphi_t, c_t\right)_{t=1}^{T}$;

Хід роботи алгоритму:

1: $U := X_m$;

2: для всіх $t := 1,\ldots,T_{\max}$

3: $c := c_t$ – вибрати клас з Y, для якого буде будуватися правило;

4: знайти найбільш інформативне правило при обмеженні на частку помилок: $\varphi_t := \arg\max\limits_{\varphi\in\Phi'} I_c\left(\varphi,U\right)$, де $\Phi' = \left\{\varphi\in\Phi \middle| E_c\left(\varphi,U\right) \le E_{\max}\right\}$;

5: якщо $I_c\left(\varphi_t,U\right) < I_{\min}$ то вихід;

6: виключити з вибірки об'єкти, виділені правилом φ_t:

$U := \left\{x\in U \middle| \varphi_t\left(x\right) = 0\right\}$;

7: якщо $\left|U\right| \le m_0$ то вихід;

Для цього на кроці 4 проводиться пошук найбільш інформативного правила $\varphi_t \in \Phi$, що допускає відносно мало помилок. Сімейство правил Φ може бути яким завгодно, лише б для нього існувала ефективна процедура пошуку закономірностей. Зокрема, якщо Φ – кон'юнкції, то на кроці 4 можна застосувати «градієнтний Алгоритм 2. Після побудови правила φ

виділені ним об'єкти вилучаються з вибірки і алгоритм переходить до пошуку наступного правила φ_{t+1} для решти об'єктів. В результаті вибірка виявляється покритою множинами виду $\left\{x\middle|\varphi_t\left(x\right)=1\right\}$. Тому список ухвалення рішень називають також *покриваючим набором закономірностей* або *машиною покриваючих множин* (set covering machine, SCM) .

Критерії відбору правил. Чому доводиться використовувати відразу два критерії відбору правил I_c і E_c? Правило з високою інформативністю I_c цілком може допускати значну частку помилок E_c, див. рис. 2 з попередньої лекції. Це небажано, так як в списку кожне правило ухвалює остаточний розв'язок. З іншого боку, правило з невеликою часткою помилок може виділяти занадто мало об'єктів, і із цієї причини не бути закономірністю.

Спільне використання обох критеріїв дозволяє відібрати предикати, що задовольняють умовам як статистичної, так і логічної закономірності.

Критерії останову. В Алгоритмі 2 одночасно працюють три критерії останову:

(1) побудова заданої кількості правил $T_{\max}$;

(2) покриття всієї вибірки, за винятком не більш m_0 об'єктів;

(3) неможливість знайти правило з інформативністю вище, ніж $I_{\min}$ по остачі вибірки.

Оптимізація складності списку ухвалення рішень.

Параметр $E_{\max}$ дозволяє знайти компроміс між точністю класифікації навчального матеріалу й складністю списку. Зменшення $E_{\max}$ приводить до зниження числа помилок на навчанні. З іншого боку, воно посилює відбір правил, сприяє зменшенню числа об'єктів, які виділяють окремі правила, і збільшенню довжини списку T. Правила, що виділяють занадто мало об'єктів, статистично не надійні і можуть допускати багато помилок на незалежних контрольних даних. Іншими словами, збільшення довжини списку при одночасному «здрібнюванні» правил може призводити до ефекту перенавчання. Із загальних міркувань $E_{\max}$ повинно приблизно дорівнювати частці помилок, яку ми очікуємо одержати як на навчальній вибірці, так і поза нею. На практиці параметр $E_{\max}$ підбирають експериментально.

Стратегія вибору класу. В описі кроку 3 Алгоритму 2 нічого не говориться про те, як вибирається клас c_t. Розглянемо два варіанти.

Перший варіант – спочатку будуються всі правила для першого класу, потім для другого, і так далі. Класи беруться в порядку убування важливості або ціни помилки. Перевага даного варіанта в тому, що правила виявляються незалежними – у межах свого класу їх можна переставляти місцями. Це поліпшує інтерпретацію правил.

Другий варіант – сполучити кроки 3 і 4 і вибирати пару $\left(\varphi_t, c_t\right) \in \Phi \times Y$, для якої інформативність $I_{c_t}\left(\varphi_t, U\right)$ максимальна. Тоді правила різних класів можуть випливати упереміж. Доведено, що списки такого типу реалізують більш широку множину функцій. При цьому поліпшується поділяюча здатність списку, але погіршується його інтерпретованість.

На практиці перший варіант часто виявляється більш зручним. У деяких випадках правила будуються тільки для $\left(M-1\right)$ класів, а в останній, найменш важливий, клас c_0 об'єкти заносяться « по залишковому принципу».

Обробка пропусків у даних. Списки ухвалення рішень дозволяють легко обійти проблему пропущених даних. Якщо для обчислення предиката

$\varphi(t)$ не вистачає даних, то вважають, що $\varphi_t(x) = 0$, і обробку об'єкта x беруть на себе наступні правила в списку. Це стосується і до стадії навчання, і до стадії класифікації.

10.1.2. Приклади списків ухвалення рішень

Логіку списку ухвалення рішень мають багато алгоритмів, що пропонувалися в різний час різними авторами під різними назвами. Численні варіанти відрізняються вибором сімейства предикатів Φ, критерієм інформативності й методом пошуку інформативних предикатів.

Приклад. Найпоширенішими є списки кон'юнкцій. Вони майже ідеально відповідають людській логіці ухвалення рішень, заснованій на послідовній перевірці досить простих правил. Тому списки ухвалення рішень часто використовуються для представлення знань, які добуваються безпосередньо з емпіричних даних. Для побудови окремих правил можна використовувати Алгоритм 2, застосовуючи на кроці 4 довільний з методів пошуку інформативних кон'юнкцій, наприклад, градієнтний Алгоритм 2 (минула лекція).

Приклад. Алгоритм 2 з сімейством предикатів Φ виду (5, минула лекція) будує покриття навчальної вибірки кулями (data dependent balls). Списки ухвалення рішень для куль добре працюють, коли метрика $\rho(x, x')$ задовольняє гіпотезі компактності, тобто близькі об'єкти часто виявляються в одному класі. Якщо це не так, то буде побудована занадто велика кількість куль невеликого радіуса. Такі алгоритми мають невисоку узагальнюючу здатність.

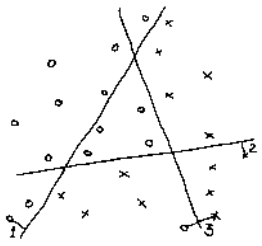


Рис. 10.1. Побудова списку ухвалення рішень з трьох гіперплощин.

Безпосереднє застосування Алгоритму 2 до сімейства предикатів Φ дозволяє побудувати покриття навчальної вибірки гіперплощинами. У цьому випадку список описує кусочно-лінійну поділяючу поверхню між класами, рис. 10.1.

Відома велика кількість евристик для послідовної побудови поділяючих гіперплощин. Це півплощина з максимальною інформативністю.

Переваги списків ухвалення рішень

- Інтерпретованість і простота класифікації. Навчене по вибірці правило класифікації можна записати у вигляді інструкції й виконувати «вручну».
- Гнучкість: залежно від вибору множини Φ можна будувати досить різноманітні алгоритмічні конструкції.
- Можливість обробки різнотипних даних і даних із пропусками.

Недоліки списків рішень.

- Якщо множина правил Φ обрана невдало, список може не побудуватися.

При цьому можливий високий відсоток відмов від класифікації.

- Можлива втрата інтерпретованості, якщо список довгий і правила різних класів випливають упереміж. У цьому випадку правила не можуть бути інтерпретовані окремої, без обліку попередніх правил, і логіка їх взаємодії стає досить заплутаною.

10.2. Дерева ухвалення рішень

Дерево ухвалення рішень (decision tree, DT) – це ще один логічний алгоритм класифікації, заснований на пошуку кон'юнктивних закономірностей. Але, на відміну від списку ухвалення рішень, при синтезі дерева всі кон'юнкції будуються одночасно.

Нагадаємо деякі поняття теорії графів.

Деревом називається кінцевий зв'язний граф із множиною вершин V, що не містить циклів, що й має виділену вершину $v_0 \in V$, у яку не входить жодне ребро. Ця вершина називається *коренем* дерева. Вершину, з якої не виходить жодного ребра, називають *термінальною вершиною* або *листком.* Інші вершини називають *внутрішніми.* Дерево називається *бінарним,* якщо з будь-якої його внутрішньої вершини виходить рівно два ребра. Вихідні ребра зв'язують кожну внутрішню вершину v *лівою дочірньою* вершиною L_v і з *правою дочірньою* вершиною R_v.

Визначення. Бінарне дерево ухвалення рішень – *це* алгоритм класифікації, що задається бінарним деревом, у якому кожній внутрішній вершині $v \in V$ приписаний предикат $\beta_v : X \to \{0,1\}$, кожній термінальній вершині $v \in V$ приписане ім'я класу $c_v \in V$. При класифікації об'єкта $x \in X$ він проходить по дереву шлях від кореня до деякого листка, відповідно до Алгоритму 3.

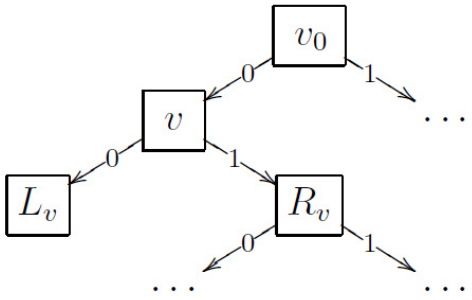


Рис. 10. 2. Приклад дерева ухвалення рішень

Алгоритм 3. Класифікація об'єкта $x \in X$ бінарним деревом ухвалення рішень

1: $v := v_0$;
2: **поки** вершина v внутрішня
3: **якщо** $\beta_v(x) = 1$ **то**
4: $v := R_v$; (перехід вправо)
5: **інакше**

6: $v := L_v$; (перехід уліво)

7: **повернути** c_v.

причому для будь-якого $x \in X$ одно й тільки один доданок у всіх цих сумах дорівнює одиниці. Замість підсумовування можна було б використовувати й диз'юнкцію.

Об'єкт x доходить до вершини v тоді і тільки тоді, коли виконується кон'юнкція $K_v\left(x\right)$, складена з усіх предикатів, які приписані до внутрішніх вершин дерева на шляху від кореня v_0 до вершини v. Нехай T – множина всіх термінальних вершин дерева. Множини об'єктів $\Omega_v = \left\{x \in X \middle| K_v\left(x\right) = 1\right\}$, які виділяються термінальними кон'юнкціями $v \in T$, попарно не перетинаються, а їх об'єднання збігається з усім простором X (це легко доводиться індукцією по числу вершин дерева). Звідси випливає, що вирішальне дерево ніколи не відмовляється від класифікації, на відміну від списку ухвалення рішень. Звідси також випливає, що алгоритм класифікації $a : X \to Y$, який реалізовує бінарне дерево рішень, можна представити у вигляді простого голосування кон'юнкцій:

$$a\left(x\right) = \arg\max_{y \in Y} \sum_{v \in T} \left[c_v = y\right] K_v\left(x\right) \qquad (10.1)$$

причому для будь-якого $x \in X$ один і тільки один доданок у всіх цих сумах дорівнює одиниці. Замість додавання можна було б використовувати і диз'юнкцію.

Природня вимога максимізації інформативності кон'юнкцій $K_v\left(x\right)$ означає, що кожна з них повинна виділяти якнайбільше навчальних об'єктів, допускаючи при цьому якнайменше помилок. Для підвищення узагальнюючої здатності дерева рішень число листків повинно бути якнайменше, і вони повинні покривати підвиборки приблизно однакової потужності $\left|\Omega_v \cap X_m\right|$.

10.2.1. Синтез дерев ухвалення рішень

Задача побудови дерева мінімальної складності, що правильно класифікує задану вибірку, у загальному випадку є NP – повною задачею. На практиці застосовують різні, більш-менш вдалі, евристики, націлені на побудову як можна більш простого дерева, що володіє як можна кращою якістю класифікації. Вибір евристик неоднозначний, як звичайно буває в таких випадках. Придумана величезна кількість різних методів синтезу бінарних вирішальних дерев по навчальній вибірці.

Розглянемо спочатку простий жадібний алгоритм, заснований на принципі «розділяй і пануй».

Алгоритм дерева ID3 (Induction of Decision Tree).
Ідея алгоритму полягає в послідовному дробленні вибірки на дві частини доти, поки в кожній частині не виявляться об'єкти тільки одного класу. Найпростіше

записати цей алгоритм у вигляді рекурсивної процедури LearnID3, яка будує дерево за заданою підвибіркою U.

Для побудови повного дерева вона застосовується до всієї вибірки й повертає вказівник на корінь побудованого дерева: $v_0 := LearnID3\left(X_m\right)$

Алгоритм 4. Рекурсивний алгоритм синтезу бінарного дерева рішень ID3

Вхідні дані:

U – навчальна вибірка;

B – множина елементарних предикатів;

Вихідні дані:

повертає кореневу вершину дерева, побудованого по вибірці U;

Хід роботи алгоритму:

1: **ПРОЦЕДУРА** LearnID3 (U);
2: **якщо** всі об'єкти з U лежать в одному класі $c \in Y$ **то**
3: створити новий листок v;
4: $c_v := c$;
5: **повернути** $\left(v\right)$;
6: знайти предикат з максимальною інформативністю:
$\beta := \arg\max_{\beta \in B} I\left(\beta, U\right)$;
7: розбити вибірку на дві частини $U = U_0 \cup U_1$ по предикату β:
$U_0 := \left\{x \in U \middle| \beta\left(x\right) = 0\right\}$
$U_1 := \left\{x \in U \middle| \beta\left(x\right) = 1\right\}$
8: **якщо** $U_0 = \varnothing$ або $U_1 = \varnothing$ **то**
9: створити новий листок v;
10: $c_v :=$ клас, у якому перебуває більшість об'єктів з U;
11: **інакше**
12: створити нову внутрішню вершину v;
13: $\beta_v := \beta$;
14: $L_v := LearnID3\left(U_0\right)$; (побудувати ліве піддерево)
15: $R_v := LearnID3\left(U_1\right)$; (побудувати праве поддерево)
16: **повернути** $\left(v\right)$

На кроці 6 Алгоритму4 вибирають предикат β з заданого сімейства B, що задає максимально інформативне розгалуження дерева – розбивка вибірки на дві частини $U = U_0 \cup U_1$.

На практиці застосовуються різні *критерії розгалуження.*

1. Критерій, орієнтований на відділення заданого класу $c \in Y$.

$$I\left(\beta,U\right)=\max_{c\notin V} I_c\left(\beta,U\right)$$

2. Більш ефективні (особливо на верхніх рівнях дерева) критерії, орієнтовані на відділення не одного, а відразу декількох класів.

Статистичний критерій інформативності узагальнює статистичне означення на випадок довільної кількості класів $Y=\left\{1,\ldots,M\right\}$:

$$I\left(\beta,U\right)=-\ln\frac{C_{P_1}^{p_1}\ldots C_{P_M}^{p_M}}{C_m^p},$$

де

P_c- кількість об'єктів класу c у вибірці U з них p_c- кількість об'єктів, які виділяються предикатом $\beta,\ p=p_1+\ldots+p_M$

Ентропійний критерій для випадку великої кількості класів є асимптотичним наближення статистичного критерію:

$$I\left(\beta,U\right)=\sum_{c\in Y}h\left(\frac{P_c}{m}\right)-\frac{p}{m}\sum_{c\in Y}h\left(\frac{p_c}{p}\right)-\frac{m-p}{m}\sum_{c\in Y}h\left(\frac{P_c-p_c}{m-p}\right),$$

де введена функція $h\left(z\right)\equiv -z\log_2 x$

3. D-Критерій – число пара об'єктів з різних класів, на яких предикат β ухвалює різні значення. У випадку двох класів він має вигляд:

$$I\left(\beta,U\right)=p\left(\beta\right)\left(N-n\left(\beta\right)\right)+n\left(\beta\right)\left(P-p\left(\beta\right)\right)$$

В Алгоритмі2 множина елементарних предикатів B может бути яким завгодно, аби тільки існував ефективний механізм вибору найбільш інформативного предиката з B на кроці 6. Коли потужність $\left|B\right|$ не велика, ця задача легко вирішується повним перебором. В іншому випадку доводиться застосовувати евристичні процедури спрямованого пошуку.

На практиці в якості елементарних предикатів найчастіше беруть прості порогові умови виду $\beta\left(x\right)=\left[f_j\left(x\right)\le d_j\right]$ або $\beta\left(x\right)=\left[f_j\left(x\right)>d_j\right]$. Кон'юнкції, складені з таких термів, добре інтерпретуються і допускають запис на природній мові. Однак ніхто не забороняє використовувати в вершинах дерева будь-які роздільні правила: кулі, гиперплощини, і, взагалі кажучи, довільні бінарні класифікатори.

10.3. Обробка пропусків

У практичних завданнях, особливо в області медицини або соціології, часто зустрічаються дані з пропусками. Що робити, якщо предикат $\beta\left(x\right)$ не може бути обчислений для даного об'єкта $x\in X$? Найтиповіша ситуація –

коли $\beta(x) = [f_j(x) \le d_j]$ або $\beta(x) = [f_j(x) > d_j]$, і значення ознаки $f_j(x)$ для даного x не зміряне.

1. *Стадія навчання.* Якщо значення $\beta(x_i)$ не визначене для навчального об'єкта $x_i \in X_m$, то при обчисленні інформативності $I(\beta, U)$ цей об'єкт не враховується. Відповідно, довжина вибірки при обчисленні інформативності зменшується. Щоб порівняння інформативності по вибірках різної довжини було «законне», критерій I повинен бути інваріантний щодо збільшення довжини вибірки при пропорційному збільшенні $p(\beta)$ і $n(\beta)$. Евристичний і энтропійний критерії задовольняють цій вимозі, а гіпергеометричний – ні; величину I_c потрібно нормувати на довжину вибірки, тобто на кроці 6 треба порівнювати питомі інформативності $\frac{I_c(\beta, U)}{|U|}$.

2. *Стадія класифікації.* Допустимо, що дерево вже побудоване, і при класифікації об'єкта x значення $\beta_v(x)$ у внутрішній вершині v не визначене. Можливі кілька стратегій обробки цієї ситуації. Найпоширеніша – *пропорційний розподіл*(proportional distribution). На стадії навчання для кожної внутрішньої вершини оцінюється ймовірність лівої гілки $\hat{p}_L = \left|\frac{U_0}{U}\right|$ і ймовірність правої гілки $\hat{p}_R = \left|\frac{U_1}{U}\right|$. На стадії класифікації об'єкт x пропускають через обидві гілки, і результати класифікації зважуються з вагами $\hat{p}_L$ і $\hat{p}_R$. Такі розгалуження можуть відбуватися в піддеревах багатократно. Тому для кожної вершини v оцінюють *апостеріорний розподіл* імовірностей класів, згідно рекуррентной формулі

$$\hat{P}_v(y|x) = \begin{cases} [y = c_v], & v \in T; \\ \hat{p}_L \hat{P}_{L_v}(y|x) \_ \hat{p}_R \hat{P}_{R_v}(y|x), & v \notin T. \end{cases}$$

10.3.1. Оцінювання ймовірностей

У багатьох додатках поряд із класифікацією об'єкта x необхідно одержувати оцінки апостеріорних імовірностей класів $\hat{P}(y|x)$.

Найпростіше оцінити їх як частку навчальних об'єктів кожного із класів $y \in Y$, що потрапили в термінальну вершину v, яка класифікувала об'єкт x. Однак така оцінка може виявитися зміщеною через перенавчання, оскільки предикати $\beta(x)$ у всіх внутрішніх вершинах дерева вибиралися по тій же самій навчальній вибірці.

Вплив перенавчання можна знизити різними способами: будувати кілька різних дерев і використовувати усереднені оцінки; використовувати для оцінювання апостеріорних імовірностей контрольну вибірку; використовувати дерево, редуковане по контрольній вибірці (див. нижче).

10.4. Трудомісткість алгоритму ID3

має порядок $O\left(B_{mid}hm\right)$, де h-глибина дерева, B_{mid}-середнє число предикатів, для яких оцінюється інформативність на кроці 6. Дійсно, найбільше часу займає обчислення інформативності підвибірки U, і цей час прямо пропорційний потужності $|U|$. Сумарна потужність усіх підвибірок, оцінюваних у вершинах одного рівня, точно дорівнює m. Виходить, число операцій, які виконують для побудови одного повного рівня дерева, має порядок $B_{mid}m$. У найгіршому випадку $B_{mid} = \left|B\right|$, однак застосування вдалих евристик для скорочення перебору на кроці 6 дозволяє суттєво зменшити B_{mid}.

Переваги алгоритму ID3.

1. Простота й інтерпретованість класифікації. Алгоритм3 здатний не тільки класифікувати об'єкт, але й видати пояснення класифікації в термінах предметної області. Пояснення будується шляхом виписування послідовності умов, перевірених для даного об'єкта на шляху від кореня дерева до листка v. Ці умови утворюють кон'юнкцію K_v, тобто легко інтерпретоване логічне правило.

2. Трудомісткість Алгоритму 4 лінійна по довжині вибірки.

3. Якщо множина предикатів B настільки багата, що на кроці 6 завжди знаходиться предикат, що розбиває вибірку U на непусті підмножини U_0 і U_1, то алгоритм будує бінарне дерево ухвалення рішень, що безпомилково класифікує вибірку X_m.

4. Не буває відмов від класифікації, на відміну від списків ухваленя рішень.

5. Алгоритм дуже простий для реалізації й легко піддається різним удосконаленням. Можна використовувати різні критерії розгалуження й критерії останову, вводити редукцію, і т.ін.

Недоліки алгоритму ID3.

1. Жадібність. Локально оптимальний вибір предиката β_v не є глобально оптимальним. У випадку невдалого вибору алгоритм не здатний повернутися на рівень нагору й замінити невдалий предикат.

2. Чим далі вершина v розташована від кореня дерева, тем менше довжина підвибірки U, по якій ухвалюється розв'язок про розгалуження у вершині v. Тим менше статистично надійним є вибір предиката β_v. У найгіршому разі все дерево може виявитися складеним з ненадійних закономірностей.

3. Висока чутливість до складу вибірки. Зміна даних в 1–2 об’єктах часто приводить до радикальної зміни структури дерева.

4. Алгоритм ID3 переускладнює структуру дерева, і, як наслідок, схильний до перенавчання. Його узагальнююча здатність (якість класифікації нових об’єктів) відносно невисока.

Основна причина недоліків – неоптимальність жадібної стратегії нарощування дерева. Для їхнього усунення застосовують різні евристичні прийоми: редукцію, елементи глобальної оптимізації, «заглядання вперед» (lookahead), побудова сукупності дерев – лісу ухвалення рішень.

10.5. Редукція дерев ухвалення рішень

Суть редукції полягає у видаленні піддерев, що мають недостатню статистичну надійність. При цьому дерево перестає безпомилково класифікувати навчальну вибірку, зате якість класифікації нових об’єктів (здатність до узагальнення), як правило, поліпшується.

Придумана величезна кількість евристик для проведення редукції, однак жодна з них, загалом кажучи, не гарантує поліпшення якості класифікації. Ми розглянемо лише найбільш прості варіанти редукції.

Передредукція (pre-pruning) або критерій *раннього останову* достроково припиняє подальше розгалуження у вершині дерева, якщо інформативність $I\left(\beta,U\right)$ для всіх предикатів $\beta \in B$ не дотягує до заданого граничного значення I_0. Для цього на кроці 8 умова ($U_0 = \varnothing$ або $U_1 = \varnothing$) заміняється умовою $I\left(\beta,U\right) \le I_0$. Поріг I_0 є керуючим параметром методу.

Передредукция вважається не найефективнішим способом уникнути перенавчання, тому що жадібне розгалуження як і раніше залишається глобально неоптимальним. Більш ефективної вважається стратегія постредукції.

Постредукція (post-pruning) переглядає всі внутрішні вершини дерева й заміняє окремі вершини або однією з дочірніх вершин (при цьому друга дочірня видаляється), або термінальною вершиною. Процес замін триває доти, поки в дереві залишаються вершини, що задовольняють критерію заміни.

Критерієм заміни є скорочення числа помилок на контрольній вибірці, відібраній заздалегідь, якай не брала участь у навчанні дерева. Стандартна рекомендація – залишати в контролі близько 30% об’єктів.

Для реалізації постредукції контрольну вибірку X_k пропускють через побудоване дерево. При цьому в кожній внутрішній вершині v запам’ятовують підмножина $S_v \subseteq X_k$ контрольних об’єктів, які попали у неї. Якщо $S_v = \varnothing$, то вершину v вважають ненадійною і заміняють термінальною за *мажоритарним правилом:* у якості c_v беруть той клас, об’єктів якого найбільше в навчальній підвибірці U, яка прийшла у вершину v.

Потім для кожної внутрішньої вершини v обчислюють кількість помилок, отриманих при класифікації вибірки S_v такими способами:

1) $r\left(v\right)$ – класифікація піддеревом, яке росте з вершини v;

2) $r_L\left(v\right)$ – класифікація піддеревом лівої дочірньої вершини L_v;

3) $r_R\left(v\right)$ – класифікація поддеревом правої дочірньої вершини R_v;

4) $r_c\left(v\right)$ – віднесення всіх об'єктів вибірки S_v док класу $c \in Y$.

Ці величини порівнюють, і, залежно від того, яка з них виявилася мінімальною, ухвалюють, відповідно, одне із чотирьох рішень:

1) зберегти піддерево вершини v;

2) замінити піддерево вершини v піддеревом лівої дочірньої вершини L_v;

3) замінити піддерево вершини v піддеревом правої дочірньої вершини R_v;

4) замінити піддерево v термінальною вершиною класу. $c = \arg\min_{c \in Y} r_c\left(v\right)$

10.6. Перетворення дерева рішень у список рішень

Існує ще одна стратегія редукції дерев рішень, при якій міняється сама структура класифікатора.

Згідно з формулою (10.1) усяке дерево ухвалення рішень еквівалентне списку рішень, складеному з термінальних кон'юнкцій $K_v\left(x\right), v \in T$. Порядок правил у списку не має значення, тому що області, $\Omega_v, v \in T$ не перетинаються. Крім того, такий список ніколи не відмовляється від класифікації, тому що об'єднання цих областей збігається з усіє множиною X.

Отриманий список можна спростити, застосувавши процедуру редукції до всіх кон'юнкцій K_v по черзі, як це розглядалося в попередній лекції в розділі «Пошук закономірностей у формі кон'юнкцій».

Редукція приводить до розширення множин Ω_v об'єктів, які виділяються кон'юнкціями K_v, і вони починають перекриватися. Виникає питання: у якому порядку розташувати правила K_v у списку. Здається розумним додавати правила до списку у порядку убування інформативності, і кожне додане правило відразу редукувати. Очевидно, скорочений список також ніколи не відмовляється від класифікації. На відміну від редукції, стабілізація може привести до утворення непокритих областей простору X, отже, до відмов алгоритму на деяких об'єктах.

Заглядання вперед

У багатьох задачах жадібне розгалуження приводить до побудови дерев, що суттєво відрізняються від оптимальних. Як приклад приведемо знамениту задачу «**виключне АБО**» (XOR).

Приклад. Нехай класів два, вибірка двовимірна, цільова залежність має вигляд $y^*\left(\xi_1,\xi_2\right)=\left[\xi_1\xi_2>0\right]$. «Ідеальне» дерево для цього випадку показане на Рис. 10.3.

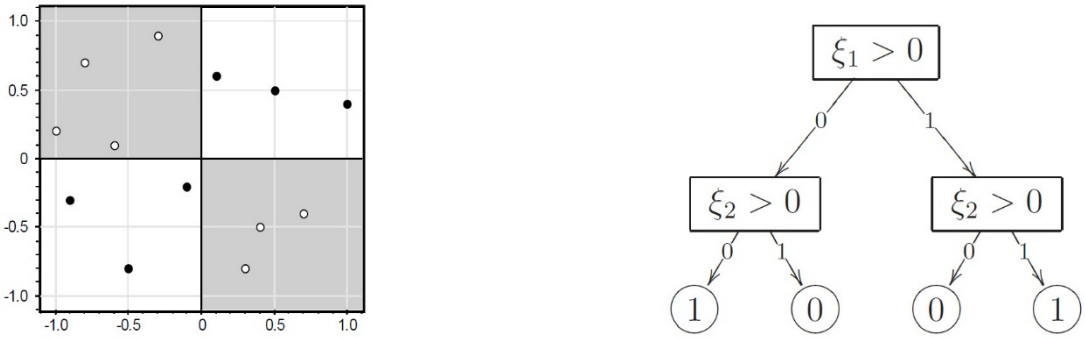


Рис. 10.3. Вибірка типу XOR та ідеальне дерево для неї

Жадібний алгоритм не зможе побудувати таке дерево, тому що правильне розгалуження в кореневій вершині не збільшує інформативність, отже, алгоритм ID3 ніколи не вибере його, і весь подальший процес побудови дерева піде неоптимальним образом. Результат показаний на Рис. 10.4.

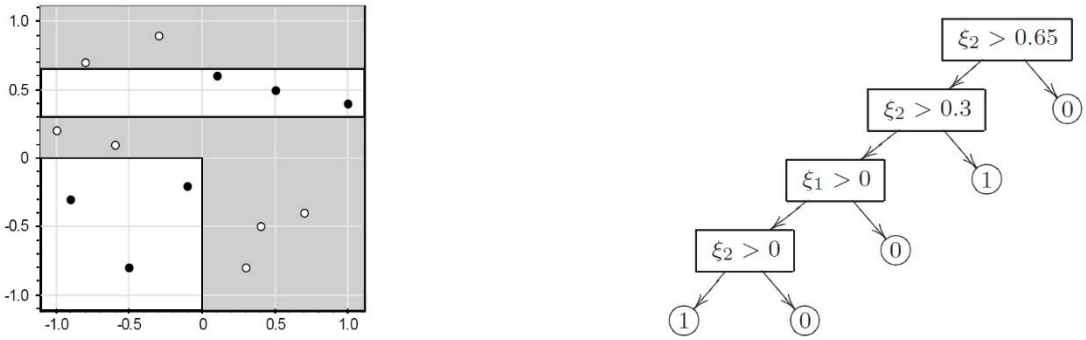


Рис. 10.4. Та ж вибірка й дерево, але побудоване жадібним алгоритмом

Ідея *заглядання вперед* (look ahead) полягає в тому, щоб на кроці 6, замість обчислення інформативності для кожного $\beta \in B$ побудувати піддерево невеликої глибини h. У внутрішню вершину v розміщують той предикат β, при якому піддерево допускає найменшу кількість помилок. Цей алгоритм працює помітно довше, але будує більш надійні й прості дерева.

При невеликих фксованих значеннях h дана стратегія практично не дає виграшу, але може бути побудований *необмежений за часом алгоритм*(anytime algorithm), який у фоновому режимі постійно поліпшує

дерево, виконуючи усе більш і більш глибоке заглядання вперед. Ця робота може бути в будь-який момент припинена для одержання готового дерева, і потім відновлена знову. Така технологія зручна, і навіть краща, у тих практичних ситуаціях, коли вибірки великі і є можливість задіяти обчислювальний ресурс, який простоює.

Запитання до розділу 10

1. Опишіть принцип роботи алгоритму ухвалення рішень.
2. Опишіть основні етапи жадібного алгоритму побудови списку ухвалення рішень.
3. В чому полягає оптимізація складності списку ухвалення рішень?
4. Ідея та основні принципи створення алгоритму ID3.
5. Опишіть процес перетворення дерева рішень у список рішень.

Розділ 11
Зважене голосування правил

Припустимо, є консиліум експертів, кожний член якого може припуститися помилки. Процедура голосування – це спосіб підвищення якості прийнятих розв'язків, при якому помилки окремих експертів компенсують один одного.

Розглянемо композиції, що полягають із логічних закономірностей.

11.1. Принцип голосування

Нехай для кожного класу $c \in Y$ побудована множина логічних закономірностей (правил), що спеціалізуються на розрізнянні об'єктів даного класу:

$$R_c = \left\{ \varphi_c^t : X \to \{0,1\} \middle| t - 1,\ldots,T_c \right\}$$

Вважають, що якщо $\varphi_c^t(x) = 1$, те правило φ_c^t відносить об'єкт $x \in X$ до класу c. Якщо ж $\varphi_c^t(x) = 0$, то правило φ_c^t утримується від класифікації об'єкта x.

Алгоритм **простого голосування** (simple voting) підраховує частку правил у наборах R_c, що відносять об'єкт x до кожного із класів:

$$\Gamma_c(x) = \frac{1}{T_c} \sum_{t=1}^{T_c} \varphi_c^t(x),\ c \in Y,$$

і відносить об'єкт *x* до того класу, за який подана найбільша частка голосів:

$$a(x) = \arg\max_{c \in Y} \Gamma_c(x) \tag{11.1}$$

Якщо максимум досягається одночасно на декількох класах, вибирається той, для якого ціна помилки менше.

Нормуючий множник $\frac{1}{T_c}$ вводиться для того, щоб набори з більшим числом правил не перетягали об'єкти у свій клас.

Алгоритм зваженого голосування (weighted voting, WV) діє більш тонко, враховуючи, що правила можуть мати різну цінність. Кожному правилу φ_c^t приписується вага $\alpha_c^t \geq 0$, і при голосуванні береться зважена сума голосів:

$$\Gamma_c(x) = \sum_{t=1}^{T_c} \alpha_c^t \varphi_c^t(x),\ \alpha_c^t \geq 0, \tag{11.2}$$

Ваги прийнято нормувати на одиницю: $\sum_{t=1}^{T_c} \alpha_c^T = 1$, для всіх $c \in Y$. Тому функцію $\Gamma_c(x)$ називають також *опуклою комбінацією* правил $\varphi_c^1,\ldots,\varphi_c^{T_c}$. Очевидно, просте голосування є окремим випадком зваженого, коли ваги однакові й рівні $\frac{1}{T_c}$.

На перший погляд, вагу правила необхідно визначати за його інформативністю. Однак, важливо ще, наскільки дане правило унікальне. Якщо є 10 гарних, але однакових (або майже однакових) правил, то їх загальна вага повинна бути зрівнянною з вагою настільки ж гарного правила, не схожого на всі інші. Таким чином, ваги повинні враховувати не тільки цінність правил, але і їх різноманітність.

Простий загальний підхід до налаштування ваг полягає в тому, щоб спочатку знайти набір правил $\{\varphi_c^t(x)\}$, потім прийняти їх за нові (бінарні) ознаки й побудувати в цьому новому ознаковому просторі лінійну розділяючу поверхню (кусочно-лінійну, якщо $|Y| > 2$). Для цього можна використовувати логістичну регресію, одношаровий персептрон або метод опорних векторів. Існують і інші підходи. Наприклад, бустинга, у якому правила настроюються послідовно, і для кожного правила відразу обчислюється його вага.

11.1.1. Проблема диверсифікованості правил

Голосуючі правила повинні бути суттєво різні, інакше вони будуть марні для класифікації. Продовжуючи аналогію з консиліумом, помітимо, що немає ніякого сенсу тримати в консиліумі експерта A, якщо він регулярно підглядає розв'язки в експерта B.

Приведемо просте теоретико-імовірнісне обґрунтування *принципу диверсифікації,* або підвищення відмінності (diversity) правил. Нехай X – імовірнісний простір, множина відповідей Y скінченна. Введемо випадкову величину $M(x)$, яка дорівнює перевазі голосів на користь правильного класу; її називають також *відступом* (margin) об'єкта x від границі класів:

$$M(x) = \Gamma_c(x) - \Gamma_{\bar{c}}(x),\ \Gamma_{\bar{c}}(x) = \max_{y \in Y \setminus \{c\}} \Gamma_y(x),\ \ c = y^*(c)$$

Якщо відступ позитивний, $M(x) > 0$, то алгоритм голосування правильно класифікує об'єкт x. Припустимо, що в середньому наш алгоритм класифікує хоча б небагато краще, чим навмання: $\mathrm{E}M > 0$. Тоді можна оцінити ймовірність помилки за нерівністю Чебишева:

$$\mathrm{P}\left\{M < 0\right\} \le \mathrm{P}\left\{\left|\mathrm{E}M - M\right| > \mathrm{E}M\right\} \le \frac{DM}{\left(\mathrm{E}M\right)^2}$$

Звідси висновок: для зменшення ймовірності помилки необхідно максимізувати очікування переваги голосів $\mathrm{E}M$ і мінімізувати його дисперсію DM. Для виконання цих умов кожний об'єкт повинен виділятися приблизно однаковим числом правил. Звичайно жодне із правил не виділяє клас цілком, тому правила повинні бути суттєво різні, тобто виділяти суттєво різні підмножини об'єктів.

Непогана евристика, що підсилює відмінності між правилами, що й дозволяє рівномірніше виділяти об'єкти навчання, використовується в алгоритмі CORAL. Спочатку для фіксованого класу $c \in Y$ будують покриваючий набір правил точно так, як це робилося для вирішальних списків рішень. Потім будується другий покриваючий набір, але при цьому забороняється використовувати ознаки, що часто входили в закономірності першого набору. Тому другий набір неминуче виявиться відмінним від першого. Потім забороняються ознаки, що часто входили в обидва набори, і будується третій набір. І так далі, для кожного класу $c \in Y$.

Інша стратегія використовується в алгоритмі бустинга. Після побудови кожної закономірності ваги виділених нею об'єктів зменшуються, заохочуючи виділення інших об'єктів наступними закономірностями.

11.1.2. Відмови від класифікації

Можливі ситуації, коли жодне із правил не виділяє об'єкт x, який підлягає класифікації. Тоді алгоритм повинен або відмовлятися від класифікації, або відносити об'єкт до класу, що має найменшу ціну помилки. Відмова алгоритму означає, що даний об'єкт є нетиповим, який не підпадає ні під одну з раніше виявлених закономірностей. Взагалі, *виявлення нетиповості*(novelty detection) прийнято вважати окремим видом задач навчання по прецедентах, поряд із класифікацією й кластеризацією. Здатність алгоритмів відмовлятися від класифікації нетипових об'єктів у багатьох додатках є скоріше перевагою, чому недоліком. У той же час, кількість відмов не повинна бути занадто великою.

Отже, при побудові алгоритмів зваженого голосування правил виникає чотири основні питання:

- Як побудувати багато правил по одній і тій же вибірці?
- Як уникнути повторів і побудови майже однакових правил?
- Як уникнути появи непокритих об'єктів і забезпечити рівномірне покриття всієї вибірки правилами?
- Як визначати ваги правил при зваженім голосуванні?

Розглянемо, як ці проблеми вирішуються у відомих алгоритмах.

11.2. Алгоритм КОРА

Алгоритм комбінаторного розпізнавання КОРА, запропонований М. М. Бонгардом в 1961-м році, будує набір кон'юнктивних закономірностей. Цей алгоритм неодноразово довів свою ефективність при розв'язку прикладних задач.

В основі алгоритму лежать наступні евристичні припущення.

Множину елементарних предикатів B підібрано настільки вдало, що серед кон'юнкцій рангу 2 або 3 уже перебуває достатня кількість інформативних закономірностей.

Оскільки ранг кон'юнкцій обмежено зверху числом 3, для пошуку закономірностей можна застосувати повний перебір.

Найбільший інтерес представляють несуперечливі закономірності (у початковому варіанті алгоритму тільки вони й будувалися).

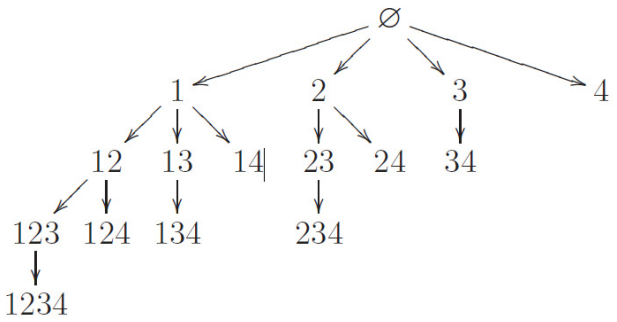


Рис. 11.1. Дерево повного перебору кон'юнкцій в алгоритмі КОРА при |B| = 4. Для стислості кон'юнкції позначені номерами предикатів, з яких вони складаються.

Алгоритм1 будує для кожного класу $c \in Y$ свій список кон'юнкцій R_c.

Алгоритм 1. Побудова списку інформативних кон'юнкцій методом пошуку в ширину (алгоритм КОРА)

Вхідні дані:

X_m – навчальна вибірка;

B – сімейство елементарних предикатів;

K – максимальний ранг кон'юнкцій;

$E_{\max}$ – максимальна частка помилок $E_c(\varphi)$ для кон'юнкцій $\varphi \in R_c$;

$D_{\min}$ – мінімальна частка позитивних об'єктів $D_c(\varphi)$ для кон'юнкцій $\varphi \in R_c$;

$T_{\min}, T_{\max}$ – обмеження на число кон'юнкцій T_c;

Вихідні дані:

списки кон'юнкцій $R_c = \left(\varphi_c^t(x) \middle| t = 1,\ldots,T_c\right)$, для всіх $c \in Y$;

Хід роботи алгоритму:

1: ініціалізувати списки: $R_c = \varnothing$ для всіх $c \in Y$;
2: **повторювати**
3: Наростити $(\varnothing)$;
4: $T := \min_{c \in Y} |R_c|$;
5: **якщо** $T < T_{\min}$ **то**
6: зменшити $D_{\min}$ і/або збільшити $E_{\max}$;
7: **якщо** $T \geq T_{\max}$ **або** час пошуку занадто великий **то**
8: збільшити $D_{\min}$ и/або зменшити $E_{\max}$;
9: **поки** $T \notin [T_{\min}, T_{\max})$

Процедура рекурсивного нарощування кон'юнкції $\varphi = \beta_{j_1} \wedge \ldots \wedge \beta_{j_s}$ шляхом додавання термів всіма можливими способами.

10: ПРОЦЕДУРА Наростити (φ);
11: **якщо** $\varphi = \varnothing$ **то** $j_s := 0$;
12: **для всіх** $j \in \{j_s + 1, \ldots, |B|\}$
13: додати терм β_j до початкової кон'юнкції:
$\varphi' := \varphi \wedge \beta_j$;
14: **якщо** $|\varphi'| \leq K$ **і** $\exists c \in Y : (D_c(\varphi') \geq D_{\min})$ **і** $E_c(\varphi') \leq E_{\max}$ **то**
15: Додати_в_список $(R_c, \varphi', T_{\max})$;
16: **інакше якщо** $|\varphi'| < K$ **і** $\exists c \in Y : (D_c(\varphi') \geq D_{\min})$ **то**
17: **Наростити** (φ') **.**

Кожна кон'юнкція містить не більше K термів, які обирають з множини предикатів B. Ядром алгоритму є рекурсивна процедура Наростити (φ), яка додає терми в конъюнкцию $\varphi(x)$ всіма можливими способами й заносить в список закономірностей R_c тільки найкращі кон'юнкції – такі, що

задовільняють критеріям відбору $D_c(\varphi) \geq D_{\min}$ і $E_c(\varphi) \leq E_{\max}$.

Перебір кон'юнкцій здійснюється методом пошуку в глибину. На Рис. 11.1 показане дерево перебору всіх кон'юнкцій при |B| = 4. У процесі перебору необхідно уникати повторного перегляду кон'юнкцій, що відрізняються тільки порядком запису термів. Для цього фіксується деяка початкова нумерація предикатів $B = \{\beta_1, \ldots, \beta_{|B|}\}$, і кон'юнкції $\varphi = \beta_{j_1} \wedge \ldots \wedge \beta_{j_s}$ нарощуються так, щоб номери предикатів строго зростали: $1 \leq j_1 < \ldots < j_s \leq |B|$.

Процедура нарощування використовує два прийоми для скорочення перебору.

По-перше, кон'юнкція φ перестає нарощуватися, якщо вона виділяє занадто мало об'єктів свого класу, $D_c(\varphi) < D_{\min}$, оскільки зі збільшенням числа термів кількість виділюваних об'єктів може тільки зменшуватися.

По-друге, кон'юнкція, що задовольняє критеріям відбору й додана в список R_c, більше не нарощується. Тим самим перевага віддається більш коротким кон'юнкціям.

Параметри $D_{\min}$ и $E_{\max}$ вірішальним чином впливають на кількість одержуваних кон'юнкцій. Якщо критерії відбору задані занадто жорстко, алгоритм може взагалі не знайти ні однієї кон'юнкції. Якщо ж критерії занадто слабкі, алгоритм буде витрачати час на перебір і оцінювання великої кількості малоінформативних кон'юнкцій. Якби не ця проблема, алгоритм зводився б до однократного застосування процедури Наростити до порожньої кон'юнкції (крок 3). Більш складний зовнішній цикл алгоритму (кроки 2-9) необхідний для того, щоб скорегувати параметри $D_{\min}$ і $E_{\max}$ в залежності від кількості одержуваних кон'юнкцій T і часу, витраченого на пошук. На практиці ці параметри найчастіше підбирають експериментальним шляхом, фактично, виконуючи зовнішній цикл вручну.

Процедура Додати_в_список заносить кон'юнкцію φ у список R_c так, щоб у підсумку в ньому залишилося близько $T_{\max}$ найкращих кон'юнкцій. Ця процедура ще знадобиться нам надалі, тому вона винесена в окремий Алгоритм 2. Включення кон'юнкції φ в список R_c, що містить не менше, ніж T найінформативніших кон'юнкцій

1: ПРОЦЕДУРА Додати_в_список R_c, φ, T ;

2: вставити кон'юнкцію φ в список R_c у порядку убування

інформативності $I_c(\varphi)$;

3: найгірша інформативність кон'юнкцій у списку R_c: $J := \min_{\psi \in R_c} I_c(\psi)$;

4: кіфлькість кон'юнкцій з найгіршою інформативністю:

$\Delta := \#\left\{\psi \in R_c \middle| I_c(\psi) = J\right\}$;

5: **якщо** $|R_c| - \Delta \geq T$ **то**

6: вилучити зі списку R_c всі кон'юнкції з найгіршою інформативністю J ;

Зауваження про деталі реалізації

• Разом з кожною кон'юнкцією має сенс зберігати бінарний вектор її значень на всіх об'єктах з X_m. Тоді після нарощування кон'юнкції (крок 13) не доведеться заново обчислювати кон'юнкцію всіх попередніх термів.

• Список R_c підтримується відсортованим за спаданням інформативності I_c. Це дозволяє ефективно реалізувати вставку кон'юнкції на кроці 2 за $O\left(\log_2 |R_c|\right)$ операцій, а виділення множини найгірших кон'юнкція на кроках 3-4 за $O(1)$ операцій.

• Чому в алгоритмі 2 не можна зробити простіше - після кроку 2 видаляти останню в списку найгіршу кон'юнкцію? Проблема в тому, що інформативність $I_c(\varphi)$, як функція дискретних величин $p_c(\varphi)$ і $n_c(\varphi)$ набуває скінченне число значень. Число кон'юнкцій φ, для яких викликається процедурами Додати_в_список, як правило, значно більше. Кон'юнкції, складені з предикатів з меншими порядковими номерами, будуть потрапляти в список раніше, але не будуть витіснятися іншими кон'юнкції з такою ж інформативністю. Це призведе до того, що предикати з меншими номерами будуть частіше входити в відібрані кон'юнкції, хоча ніяких об'єктивних підстав для такого переваги немає.

Переваги алгоритму КОРА.

• Короткі кон'юнкції легко інтерпретуються в термінах предметної області. Алгоритм здатний не тільки класифікувати об'єкти, але і пояснювати свої рішення на мові, зрозумілій фахівцям.

• При малих K, $K \leq 3$, алгоритм дуже ефективний.

• Якщо короткі інформативні кон'юнкції існують, вони обов'язково будуть знайдені, так як алгоритм здійснює повний перебір.

Недоліки алгоритму КОРА.

• При невдалому виборі множини предикатів B коротких інформативних кон'юнкцій може просто не існувати. У той же час, збільшення числа K призводить до експоненціального падіння ефективності, так як число операцій, що виконуються алгоритмом, становить $O\left(\left|B\right|^K m\right)$.

• Алгоритм не прагне диверсифікувати кон'юнкції і забезпечувати рівномірність покриття об'єктів. Це негативно позначається на узагальнюючій здатності (ймовірності помилки) алгоритму.

• Немає налаштування коефіцієнтів α_c^t; передбачається просте голосування.

11.3. Алгоритм ТЕМП

Повний перебір усіх кон'юнкцій рангу не більше K потребує експоненціального по K числа операцій. У реальних задачах об'єм обчислень стає величезним уже при $K > 3$, і від ідеї повного перебору доводиться відмовитися.

Існує дві стандартні стратегії перебору кон'юнкцій: *пошук у глибину* (depth-first search) і *пошук в ширину* (breadth-first search). Перша застосовується в алгоритмі КОРА, друга – в алгоритмі ТЕМП, запропонованим. Пошук в ширишну працює небагато швидше, і в нього легше вбудовувати різні евристики, що скорочують перебір.

У початковому варіанті алгоритм ТЕМП виконував повний перебір усіх кон'юнкцій рангу не більше K. Нижче описаний злегка модифікований варіант, що дозволяє обмежити перебір і збільшити максимальний ранг кон'юнкцій K.

Алгоритм3 починає процес пошуку закономірностей з побудови кон'юнкцій рангу 1. Для цього відбираються не більше, ніж T_1 найінформативніших предикатів з базової множини B. Потім до кожного з відібраних предикатів додають по одному терму з B всіма можливими способами. Виходить не більш $T_1\left|B\right|$ кон'юнкцій рангу 2, з яких знову відбираються T_1 самих інформативних. І так далі. На кожному кроці процесу робиться спроба додати один терм до кожної з наявних кон'юнкцій. Нарощування кон'юнкцій припиняється або при досягненні максимального рангу K, або коли ні одну з кон'юнкцій не вдається поліпшити шляхом додавання терма.

Кращі кон'юнкції, зібрані із усіх кроків, заносяться в списки R_c. Таким чином, списки R_c можуть містити кон'юнкції різного рангу.

Параметр T_1 дозволяє знайти компроміс між якістю й швидкістю роботи алгоритму. При $T_1 = 1$ алгоритм ТЕМП працює винятково швидко й будує єдину кон'юнкцію, додаючи терми по черзі.

При збільшенні T_1 простір пошуку розширюється, алгоритм починає працювати повільніше, але знаходить більше інформативних кон'юнкцій. На практиці вибирають максимальне значення параметра T_1, при якім пошук забирає прийнятний час. Однак стратегія пошуку однаково залишається жадібною – терми оптимізуються окремо, і при доборі кожного терма враховуються тільки попередні, але не наступні терми.

Алгоритм 3. Побудова списку інформативних кон'юнкцій методом пошуку в ширину (алгоритм ТЕМП)

Вхідні дані:

X_m – навчальна вибірка;

B– сімейство елементарних предикатів;

$c \in Y$ – клас, для якого будується список кон'юнкцій;

K- максимальний ранг кон'юнкцій;

T_1 – число кращих кон'юнкцій, що відбираються на кожному кроці;

T_0 – число кращих кон'юнкцій, що відбираються на останньому кроці, $T_0 \le T_1$;

$I_{\min}$ – поріг інформативності;

$E_{\max}$ – поріг припустимої частки помилок;

X_k – контрольна вибірка для проведення редукції;

Вихідні дані:

список кон'юнкцій $R_c = \left\{ \varphi_c^t(x) \middle| t = 1,\ldots,T_c \right\}$;

Хід роботи алгоритму:

1: $R_c := \varnothing$;

2: **для всіх** $\beta \in B$

3: Додати_в_список $\left(R_c, \beta, T_1\right)$;

4: **для всіх** $k = 2,\ldots,K$

5: **для всіх** кон'юнкцій $\varphi \in R_c$ рангу $\left(k-1\right)$

6: **для всіх** предикатів $\beta \in B$, яких ще немає в кон'юнкції φ

7: додати терм β до кон'юнкції φ:

$\varphi' := \varphi \wedge \beta$;

8: **якщо** $I_c\left(\varphi'\right) \ge I_{\min}$ і $E_c\left(\varphi'\right) \le E_{\max}$ і кон'юнкції φ' немає в R_c, **то**

9: Додати_в_список $\left(R_c, \varphi', T_1\right)$;

10: **для всіх** кон'юнкцій $\varphi \in R_c$

11: Стабілізація φ;

12: Редукція $\left(\varphi, X_k\right)$;

13: вилучити зі списку R_c дублюючі кон'юнкції;

14: залишити в списку R_c не більше, ніж T_0 кращих кон'юнкцій;

Для поліпшення кон'юнкцій до них застосовують евристичні методи «фінального шліфування» – стабілізацію й редукцію.

В результаті стабілізаці кон'юнкції стють такими, що локально не покращуються. Алгоритм у цілому стає більш стійким – при незначних змінах у складі навчальної вибірки він частіше знаходить ті самі закономірності, тобто поліпшується його здатність узагальнювати емпіричні факти.

У результаті стабілізації деякі кон'юнкції можуть співпадати, і в списку з'являться дублікати. Їхнє видалення передбачено на кроці 13. Якщо список R_c підтримується відсортованим по інформативності, то видалення дублікатів є недорогою операцією, тому що досить перевіряти на збіг тільки сусідні кон'юнкції з однаковою інформативністю.

Якщо задати $T_1 = \infty$, то алгоритм виконає повний перебір, як у початковому варіанті ТЕМП. «Фінальне шліфування» у цьому випадку не потрібне.

І ще одна відмінність описаного алгоритму від початкового ТЕМП: тут кон'юнкції відбираються по інформативності, тоді як в початковому варіанті використовувався евристичний критерій $p_c\left(\varphi\right) \geq p_{\min}$, $n_c\left(\varphi\right) \leq n_{\max}$.

Переваги алгоритму ТЕМП.

• ТЕМП істотно більш ефективний, ніж КОРА, особливо при пошуку кон'юнкцій рангу більше 3. Він вирішує поставлене завдання за $O\left(KT_1\left|B\right|m\right)$ операцій, тоді як КОРА має трудомісткість $O\left(\left|B\right|^K m\right)$.

• Параметр T_1 дозволяє управляти жадібністю алгоритму і знаходити компроміс між якістю кон'юнкцій і швидкістю роботи алгоритму.

• Завдяки простоті й ефективності алгоритм ТЕМП можна використовувати в складі інших алгоритмів як генератор кон'юнкцій, досить близьких до оптимальних.

Недоліки алгоритму ТЕМП.

• Немає гарантії, що будуть знайдені найкращі кон'юнкції, особливо при малих значеннях параметра T_1.

• Алгоритм не прагне збільшувати різноманітність кон'юнкцій, щоб домогтися рівномірного покриття об'єктів вибірки. Стабілізація і редукція лише почасти компенсує цей недолік.

• Немає налаштування коефіцієнтів α_c^t; передбачається просте голосування.

11.4. Алгоритм бустинга

Алгоритми КОРА і ТЕМП мають загальний недолік – вони не прагнуть збільшувати різноманітність кон'юнкцій. Ця проблема вирішується в алгоритмі бустинга.

Бустинг(boosting) запропонували американські вчені Фройнд і Шапир як універсальний метод побудови опуклої комбінації класифікаторів. Ми вже

У бустингу закономірності будуються послідовно, і після побудови чергової закономірності ваги виділених нею об'єктів змінюються – зменшуються в позитивних і збільшуються в негативних об'єктів. Оновлений вектор ваг w використовується для пошуку наступної закономірності φ по критерію максимуму *зваженої інформативності*. У результаті кожна наступна закономірність прагне виділити «найменш покриті» об'єкти, які виявилися «найбільш важкими» для попередніх закономірностей. Це сприяє підвищенню різноманітності закономірностей, більш рівномірному покриттю об'єктів і підвищенню узагальнюючої здатності опуклої комбінації закономірностей.

Описана стратегія нагадує алгоритм побудови вирішального списку. Різниця в тому, що там було досить покрити об'єкт один раз, після чого він виключався з розгляду. Тут же кожне покриття тільки змінює вагу об'єкта.

Для реалізації цієї ідеї залишається зрозуміти, як саме повинні обчислюватися ваги об'єктів і ваги закономірностей на кожному кроці алгоритму.

Розглянемо задачу класифікації із двома класами, $Y = \{-1, +1\}$ і алгоритм зваженого голосування (1),(2), який складається з $T = T_{-1} + T_{+1}$ закономірностей:

$$a_T(x) = sign\left(\underbrace{\sum_{t=1}^{T_{+1}} \alpha_{+1}^t \varphi_{+1}^t(x)}_{\Gamma_{+1}(x)} - \underbrace{\sum_{t=1}^{T_{-1}} \alpha_{-1}^t \varphi_{-1}^t(x)}_{\Gamma_{-1}(x)} \right), \; \alpha_c^t > 0, \; c \in Y$$

Експонентна апроксимація граничної функції втрат. Нехай уже побудовано T закономірностей*, які* разом становлять алгоритм класифікації $a_T(x)$. При додаванні ще однієї закономірності $\varphi_c(x)$ у список R_c зважена сума голосів за клас $c \in \{-1, +1\}$ матиме вигляд:

$$\Gamma_c'(x) = \Gamma_c(x) + \alpha \varphi_c(x)$$

Завдання полягає в тому, щоб знайти закономірність φ_c і її вагу α , при яких алгоритм $a_{T_{+1}}\left(x\right)$ допускає мінімальне число помилок на навчальній вибірці X_m

Число помилок алгоритму $a_T\left(x\right)$ перед додаванням закономірності φ_c:

$$Q_T = \sum_{i=1}^{m}\left[\Gamma_{y_i}\left(x_i\right) - \Gamma_{-y_i}\left(x_i\right) < 0\right]$$

Число помилок алгоритму $a_{T_{+1}}\left(x\right)$ після додавання закономірності φ_c:

$$Q_{T_{+1}}\left(\varphi_c,\alpha\right) = \sum_{i=1}^{m}\left[y_i = c\right]\left[\Gamma_{y_i}\left(x_i\right) - \Gamma_{-y_i}\left(x_i\right) + \alpha\varphi_c\left(x_i < 0\right)\right] +$$

$$+\sum_{i=1}^{m}\left[y_i \neq c\right]\left[\Gamma_{y_i}\left(x_i\right) - \Gamma_{-y_i}\left(x_i\right) - \alpha\varphi_c\left(x_i < 0\right)\right]$$

Виписаний функціонал містить параметр α усередині порогової функції виду $\left[z\left(\alpha\right) < 0\right]$, отже, є розривною функцією від α . Мінімізація такого функціонала є нетривіальною задачею комбінаторної оптимізації. Спробуємо розв'язати її приблизно. Замінимо порогову функцію неперервно диференційованою оцінкою зверху. Тоді мінімізацію по α можна буде виконати аналітично.

Вибір конкретної апроксимуючої функції є евристикою. Найбільш прості викладки виходять, якщо скористатися оцінкою $\left[z < 0\right] \leq e^{-z}$.

Запишемо верхню оцінку $\tilde{Q}_T$ функціонала Q_T :

$$Q_T \leq \tilde{Q}_T \equiv \sum_{i=1}^{m}\underbrace{\exp\left(\Gamma_{-y_i}\left(x_i\right) - \Gamma_{y_i}\left(x_i\right)\right)}_{w_i} = \sum_{i=1}^{m} w_i$$

Якщо алгоритм $a_T\left(x\right)$ правильно класифікує об'єкт x_i, то $w_i < 1$. Якщо помиляється, то $w_i > 1$. Чим більше перевага голосів на користь помилкового класу, тим більше вага w_i. Таким чином, більша вагу одержують найбільше «складні» об'єкти.

Введемо вектор $w = \left(\tilde{w}_i\right)_{i=1}^{m}$ з компонентами $\tilde{w}_i = w_i m \big/ \tilde{Q}_T$. Тоді буде виконана умова нормування $\sum_{i=1}^{m}\tilde{w}_i = m$. Наступна теорема показує, що вибір $\tilde{w}_i$ у якості ваг об'єктів оптимальний для побудови T_{+1}-й закономірності.

Теорема. Мінімум функціонала $\tilde{Q}_{T_{+1}}\left(\varphi_c,\alpha\right)$ досягається при

$$\varphi_c^* = \arg\max_{\varphi} J_c^w\left(\varphi, X_m\right),\ J_c^w\left(\varphi, X_m\right) = \sqrt{p_c^w\left(\varphi\right)} - \sqrt{n_c^w\left(\varphi\right)}; \qquad (11.3)$$

$$\alpha^* = \frac{1}{2}\ln\frac{p_c^w\left(\varphi_c^*\right)}{n_c^w\left(\varphi_c^*\right)},\ \text{при}\ n_c^w\left(\varphi_c\right) \neq 0$$

де функції $p_c^w\left(\varphi\right)$ і $n_c^w\left(\varphi\right)$ визначаються по формулах

$$P_c^w = \sum_{i=1}^{m} w_i\left[y_i = c\right];\ p_c^w\left(\varphi\right) = \sum_{i=1}^{m} w_i\left[y_i = c\right]\left[\varphi\left(x_i = 1\right)\right];$$

$$N_c^w = \sum_{i=1}^{m} w_i\left[y_i \neq c\right];\ n_c^w\left(\varphi\right) = \sum_{i=1}^{m} w_i\left[y_i \neq c\right]\left[\varphi\left(x_i = 1\right)\right];$$

Зауваження. Теорема не дозволяє визначити вагу несуперечливої закономірності φ_c^*, тому що знаменник (1.11) звертається в нуль. Щоб запобігти цьому небажаному ефекту, уводять додатковий параметр $\lambda \in \left(0,1\right)$, злегка модифікуючи формулу розрахунків ваги:

$$\alpha^* = \frac{1}{2}\ln\frac{p_c^w\left(\varphi_c^*\right)}{\max\left\{n_c^w\left(\varphi_c^*\right), \lambda\right\}}$$

Це однаково, що ввести у функціонал $\tilde{Q}_{T_{+1}}$ додатковий штрафний доданок з коефіцієнтом λ:

$$\tilde{Q}_{T_{+1}} \leq \approx Q_{T_{+1}} = \frac{\tilde{Q}_T}{m}\left(m - p - n + e^{-\alpha}p + e^{-\alpha}ne^{-\alpha}p + e^{\alpha}\lambda\left[n = 0\right]\right)$$

Чим менше λ, тем більша вагу одержують несуперечливі закономірності. Таким чином, управляючи параметром λ, можна змусити алгоритм шукати переважно несуперечливі закономірності.

Алгоритм 4. Побудова опуклої комбінації закономірностей при класифікації на два класи, $Y = \left\{-1, +1\right\}$ (алгоритм бустинга)

Вхідні дані:

X_m – навчальна вибірка;

Φ – сімейство базових предикатів;

T – загальне число закономірностей усіх класів;

λ – коефіцієнт заохочення несуперечливих закономірностей;

Вихідні дані:

списки закономірностей і їх ваг $\left\{\varphi_c^t\left(x\right), \alpha_c^t \middle| t = 1,\ldots,T_c\right\}$ для всіх $c \in Y$;

Хід роботи алгоритму:

1: ініціалізувати ваги: $w_i := 1$ для всіх $i = 1,\ldots,m$;

2: **для всіх** $t = 1,\ldots,T$

3: $c := c_t$ –вибрати клас для якого будемо будувати закономірність;

4: $$\varphi_c^t := \arg\max_{\varphi\in\Phi} \sqrt{p_c^w\left(\varphi\right)} - \sqrt{n_c^w\left(\varphi\right)};$$

5: $$\alpha_c^t := \frac{1}{2}\ln\frac{p_c^w\left(\varphi\right)}{\max\left\{n_c^w\left(\varphi\right), \lambda\right\}};$$

6: **для всіх** $i = 1,\ldots,m$ перерахувати вагу w_i

7: $$w_i := \begin{cases} w_i & \varphi_c\left(x\right) = 0 \\ w_i \exp\left(-\alpha_c^t\right) & \varphi_c\left(x\right) = 1 \ i \ y_i = c \\ w_i \exp\left(\alpha_c^t\right) & \varphi_c\left(x\right) = 1 \ i \ y_i \neq c \end{cases}$$

8: нормувати ваги: $Z := \frac{1}{m}\sum_{i=1}^{m} w_i;\ w_i := \frac{w_i}{Z}$ для всіх $i = 1,\ldots,m$

Зауваження. Після того, як закономірність $\varphi_c\left(x\right)$ знайдена й обчислений відповідний їй коефіцієнт α, легко перерахувати ваги об'єктів w_i' (для наступного кроку алгоритму:

$$w_i' = w_i\left(\left[y_i = c\right]e^{-\alpha\varphi_c\left(x_i\right)} + \left[y_i \neq c\right]e^{\alpha\varphi_c\left(x_i\right)}\right) = \begin{cases} w_i & \varphi_c\left(x\right) = 0 \\ w_i \exp\left(-\alpha_c\right) & \varphi_c\left(x\right) = 1 \ i \ y_i = c \\ w_i \exp\left(\alpha_c\right) & \varphi_c\left(x\right) = 1 \ i \ y_i \neq c \end{cases}$$

Таким чином, при виділенні позитивного об'єкта його вага зменшується в e^{α} раз, а при виділенні негативного – збільшується в стільки ж раз. У результаті багаторазового застосування цього правила бустинг прагне виділяти позитивні об'єкти «рівномірно часто», а негативні – «рівномірно рідко».

Зауваження. Фактично, теорема 2 вводить ще один функціонал інформативності предикатів $J_c^w\left(\varphi, X_m\right) = \sqrt{p_c^w\left(\varphi\right)} - \sqrt{n_c^w\left(\varphi\right)}$. Як видне з таблиці1, він достатньо адекватно оцінює якість закономірностей, а обчислюється суттєво простіше, ніж I_c або $IGain_c$. Його можна застосовувати не тільки в алгоритмі бустинга, але й окремо, оскільки теорема 2 залишається

вірна для функціонала $\tilde{Q}_1$, якщо покласти всі ваги w_i равными одиниці.

Зауваження. У процесі роботи алгоритму має сенс проаналізувати розподіл ваг об'єктів. Об'єкти з найбільшими вагами w_i являются найбільше «важкими» для всіх побудованих закономірностей. Можливо, це «шумові викиди» – об'єкти, в описі яких допущені грубі помилки. Виключення таких об'єктів, називане *цензуруванням вибірки*, як правило, підвищує якість класифікації. Після виключення викидів побудову закономірностей краще почати заново, оскільки викиди заважали адекватно оцінювати інформативність закономірностей.

Зауваження. В Алгоритмі 4 основна робота виконується на кроці 4, коли шукають закономірність, що доставляє максимальне значення функціоналу інформативності $J_c^w(\varphi)$. Для розв'язку даної задачі можна скористатися будь-яким доступним алгоритмом синтезу закономірностей – градієнтним, емерджентним, ТЕМПом з параметром $T_0 = 1$, і т.ін. Єдина модифікація, яка для цього буде потрібна – замінити критерій інформативності I_c на J_c^w. Відзначимо, що симбіоз «бустинг + ТЕМП із редукцією правил» практично збігається з алгоритмом SLIPPER (simple learner with iterative pruning to produce error reduction), який уважається одним із кращих логічних алгоритмів класифікації.

Переваги бустинга.

- *Висока узагальнююча здатність* досягається за рахунок більш рівномірного покриття об'єктів закономірностями.
- *Коректність на навчальній вибірці* гарантується при досить слабких додаткових обмеженнях.
- *Універсальність.* Можна використовувати будь-яке сімейство базових предикатів Ф, не обов'язково кон'юнкції. Передбачається тільки, що на кроці 4 застосовується досить ефективний алгоритм перебору по множині Ф.
- *Ефективність* бустинга цілком визначається цим зовнішнім алгоритмом. Власні накладні витрати бустинга невеликі.
- *Интерпретованість.* Лінійна комбінація правил легко інтерпретується, якщо правил небагато й вони мають вигляд кон'юнкцій. Ваги правил завжди позитивні й показують ступінь їх важливості для класифікації.
- *Фільтрація шуму.* Можливість виділення шумових об'єктів.

Недоліки бустинга.

- Ефект перенавчання може все-таки спостерігатися, якщо на кроці 4 не вдається знаходити досить гарні закономірності.

- У цьому випадку різко зростає число правил T, необхідних для забезпечення коректності й лінійна комбінація втрачає властивість интерпретованості. Алгоритм голосування стає «чорним ящиком» з неочевидною внутрішньою логікою, коли число правил перевищує кілька десятків.

Запитання до розділу 11

1. Опишіть алгоритми простого та зваженого голосування.
2. Опишіть основні етапи та суть алгоритму КОРА.
3. Опишіть основні етапи та суть алгоритму ТЕМП.
4. На яких принципах базується алгоритм бустинга.
5. Основні задачі експонентної апроксимації граничної функції втрат.

Розділ 12
Штучні нейронні мережі

Людині і вищим тваринам буквально на кожному кроці доводиться розпізнавати, приймати рішення і вчитися. Нейромережевий підхід виник з прагнення зрозуміти, яким чином мозок вирішує такі складні завдання, і реалізувати ці принципи в автоматичних пристроях. Поки штучні нейронні мережі (artificial neural networks, ANN) є лише гранично спрощеними аналогами природних нейронних мереж. Нервові системи тварин і людини набагато складніше тих пристроїв, які можна створити за допомогою сучасних технологій. Однак для успішного вирішення багатьох практичних завдань виявилося цілком достатньо «підглянути» лише загальні принципи функціонування нервової системи. Деякі різновиди ANN представляють собою математичні моделі, що мають лише віддалене подібність з нейрофізіології, що аж ніяк не перешкоджає їх практичного застосування.

Нейронні мережі (Neural Networks) – це моделі біологічних нейронних мереж мозку, в яких нейрони імітуються відносно простими, часто однотипними, елементами (штучними нейронами).

Нейронна мережу можна представити орієнтованим графом з зваженими зв’язками, в якому штучні нейрони є вузлами, а синаптичні зв’язки - дугами.

Нейронні мережі широко використовуються для вирішення різноманітних завдань.

Серед областей застосування нейронних мереж - автоматизація процесів розпізнавання образів, прогнозування, адаптивне управління, створення експертних систем, організація асоціативної пам’яті, обробка аналогових і цифрових сигналів, синтез і ідентифікація електронних ланцюгів і систем.

За допомогою нейронних мереж можна, наприклад, передбачати обсяги продажів виробів, показники біржового ринку, виконувати розпізнавання сигналів, конструювати системи, які сомі навчаються

Моделі нейронних мереж можуть бути програмного і апаратного виконання. Ми будемо розглядати мережі першого типу.

Якщо говорити простою мовою, шарувата нейронна мережа являє собою сукупність нейронів, які становлять шари. У кожному шарі нейрони між собою ніяк не пов’язані, але пов’язані з нейронами попереднього і наступного шарів. Інформація надходить з першого на другий шар, з другого - на третій і т. ін.

Серед завдань машинного навчання, що вирішуються за допомогою нейронних мереж, будемо розглядати такі:

Класифікація (навчання з учителем). Приклади завдань класифікації: розпізнавання тексту, розпізнавання мови, ідентифікація особистості.

Прогнозування. Для нейронної мережі задача прогнозування може бути поставлена таким чином: знайти найкраще наближення функції, заданої кінцевим набором вхідних значень (навчальних прикладів). Наприклад, нейронні мережі дозволяють вирішувати завдання відновлення пропущених значень.

Кластеризація (навчання без вчителя). Прикладом завдання кластеризації може бути задача стиснення інформації шляхом зменшення розмірності даних. Завдання кластеризації вирішуються, наприклад, до самоорганізації картами Кохонена. Цим мережам буде присвячена окрема лекція.

Елементи нейронних мереж

Штучний нейрон (формальний нейрон) – елемент штучних нейронних мереж, що моделює деякі функції біологічного нейрона.

Головна функція штучного нейрона – формувати вихідний сигнал в залежності від сигналів, що надходять на його входи.

В найпоширенішою конфігурації вхідні сигнали обробляються адаптивним суматором, потім вихідний сигнал суматора надходить в нелінійний перетворювач, де перетворюється функцією активації, і результат подається на вихід (в точку розгалуження).

Загальний вигляд штучного нейрона наведено на рис.1

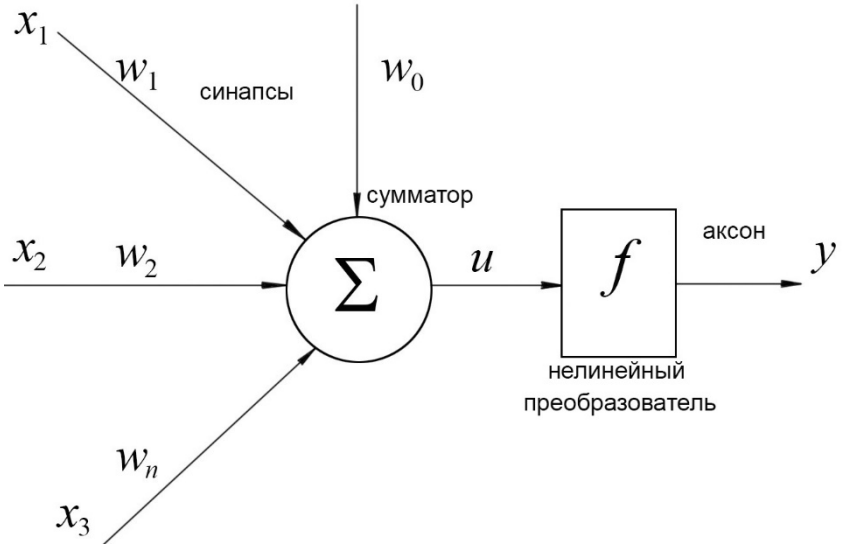


Рис. 12.1. Штучний нейрон

Нейрон характеризується поточним станом і має груп синапсів – односпрямованих вхідних зв'язків, з'єднаних з виходами інших нейронів.

Нейрон має аксон – вихідну зв'язок даного нейрона, з якої сигнал (збудження або гальмування) надходить на синапси наступних нейронів.

Кожен синапс характеризується величиною синаптичного зв'язку (її вагою w_i).

Поточний стан нейрона визначається як зважена сума його входів:

$$u = \sum_{i=1}^{n} x_i w_i$$

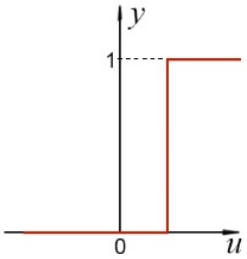
Вихід нейрона є функція активації: $y = f\left(u\right)$

Функція активації (активаційна функція, функція збудження) – функція, що обчислює вихідний сигнал штучного нейрона. Як аргумент

приймає сигнал u , отриманий на виході вхідного суматора Σ. Найбільш часто використовуються наступні функції активації.

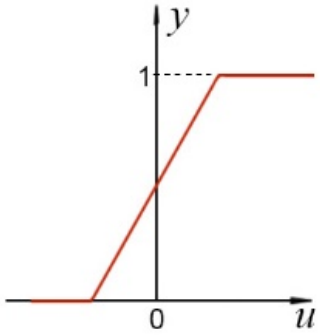
1. Одиничний стрибок або жорстка порогова функція

Проста кусочно-лінійна функція. Якщо вхідне значення менше порогового, то значення функції активації дорівнює мінімальному допустимому, інакше - максимально допустимому.



2. Лінійний поріг або гістерезис

Нескладна кусочно-лінійна функція. Має два лінійних ділянки, де функція активації тотожно дорівнює мінімально допустимому і максимально допустимого значення і є ділянка, на якому функція строго монотонно зростає.



3. сигмоїдальна функція або сигмоид

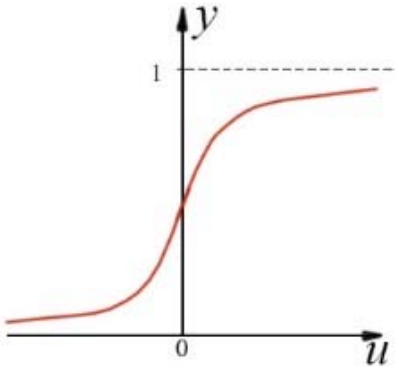
Монотонно зростаюча всюди диференційована s-подібна нелінійна функція з насиченням. Сигмоїд дозволяє підсилювати слабкі сигнали і не наситилися з сильних сигналів. Визначено, що подібна нелінійна функція активації вирішує поставлену їм дилему шумового насичення.

Слабкі сигнали мають потребу у великій мережевому посилення, щоб дати придатний до використання вихідний сигнал. Однак підсилювальні каскади з великими коефіцієнтами посилення можуть призвести до насичення виходу шумами підсилювачів, які присутні в будь-який фізично реалізованої мережі. Сильні вхідні сигнали в свою чергу також будуть приводити до насичення підсилюючих каскадів, виключаючи можливість корисного використання виходу. Яким чином одна і та ж мережа може обробляти як слабкі, так і сильні сигнали?

Прикладом сигмоїдальної функції активації може служити логістична функція, що задається наступним виразом:

$$y = \frac{1}{1 + \exp\left(-\alpha u\right)}$$

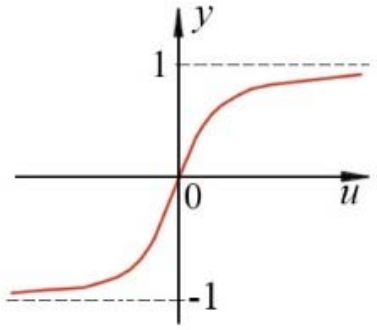
де α– параметр нахилу сигмоидальной функції активації. Змінюючи цей параметр, можна побудувати функції з різною крутизною.



Ще одним прикладом сигмоїдальної функції активації є гіперболічний тангенс, що задається наступним виразом:

$$y = th\left(\frac{u}{\alpha}\right)$$

де α – це також параметр, що впливає на нахил сигмоїдальної функції.



Функції активації типу одиничного стрибка та лінійного порогу зустрічаються дуже рідко і, як правило, використовуються на навчальних прикладах. У практичних завдань майже завжди застосовується сигмоїдальна функція активації.

Нелінійний перетворювач – це елемент штучного нейрона, що перетворює поточний стан нейрона (вихідний сигнал адаптивного суматора) в вихідний сигнал нейрона по деякому нелінійному закону (активаційної функції).

Точка розгалуження (вихід) – це елемент формального нейрона, який посилає його вихідний сигнал за кількома адресами і має один вхід і кілька виходів.

На вхід точки розгалуження зазвичай подається вихідний сигнал нелінійного перетворювача, який потім надсилається на входи інших нейронів.

Архітектура нейронних мереж

Сьогодні відома велика кількість нейронних структур і їх модифікацій, орієнтованих на рішення конкретного типу завдань. Найбільш відомі типи таких структур показані на рис. 12.2.

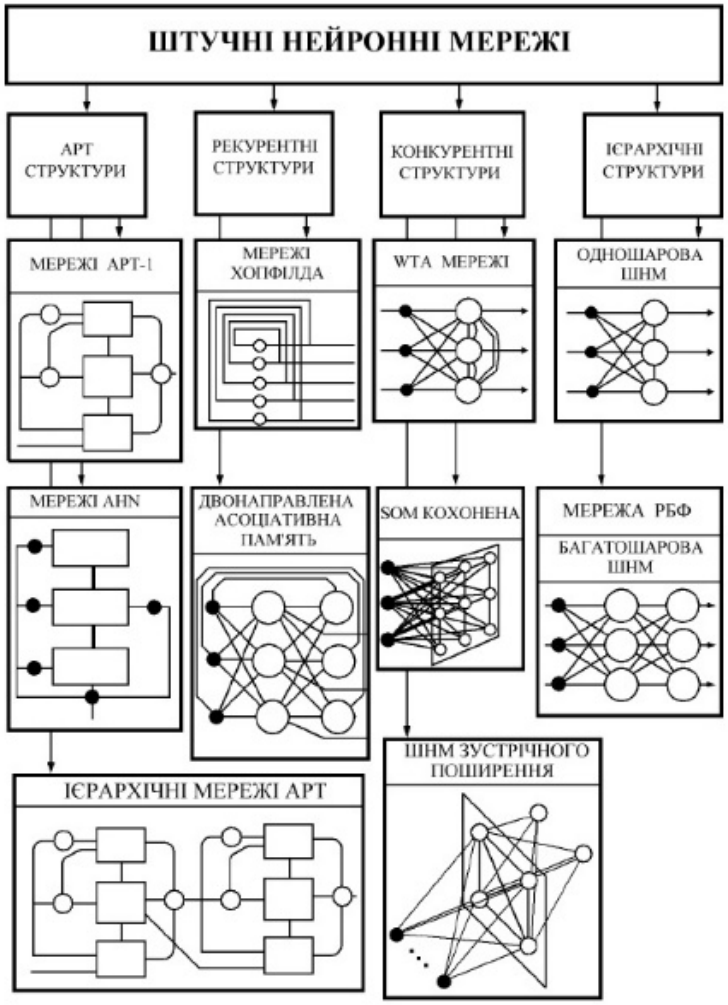


Рис. 12.2. Архітектури нейронних мереж

Глибокі нейронні мережі – всі зв’язки направлені строго від вхідних нейронів до вихідних. До таких мереж відносяться, наприклад: найпростіший персептрон (розроблений Розенблатта) і багатошаровий персептрон.

Реккурентное нейронні мережі – сигнал з вихідних нейронів або нейронів прихованого шару частково передається назад на входи нейронів вхідного шару.

Радіально базисні функції – вид нейронної мережі, що має прихований шар з радіальних елементів і вихідний шар з лінійних елементів. Мережі цього типу досить компактні і швидко навчаються. Запропоновано в роботах Broomhead and Lowe (1988) і Moody and Darkin (1989). Радіально базисна мережа володіє наступними особливостями: один прихований шар, тільки нейрони прихованого шару мають нелінійну активаційну функцію і синаптичні ваги вхідного і прихованого шарів дорівнюють одиниці.

Самоорганізовані карти або Мережі Кохонена – такий клас мереж, як правило, навчається без учителя і успішно застосовується в задачах розпізнавання. Мережі такого класу здатні виявляти новизну у вхідних даних: якщо після навчання 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 має всього два прошарки: вхідний і вихідний, складений з радіальних елементів.

Нейронні мережі можуть бути синхронні і асинхронні.

У синхронних нейронних мережах в кожен момент часу свій стан змінює лише один нейрон.

В асинхронних – стан змінюється відразу у цілої групи нейронів, як правило, у всього шару.

Можна виділити дві базові архітектури – шаруваті і повнозв'язні мережі.

Ключовим в шаруватих мережах є поняття шару.

Шар – один або кілька нейронів, на входи яких подається один і той же загальний сигнал.

Шаруваті нейронні мережі – нейронні мережі, в яких нейрони розбиті на окремі групи (верстви) так, що обробка інформації здійснюється пошарово.

У шаруватих мережах нейрони і-го шару отримують вхідні сигнали, перетворюють їх і через точки розгалуження передають нейронам (i + 1) шару. І так до k-го шару, який видає вихідні сигнали для інтерпретатора і користувача. Число нейронів в кожному шарі не пов'язане з кількістю нейронів в інших шарах, може бути довільним.

Шаруваті мережі, в свою чергу, можуть бути одношаровими і багатошаровими.

Одношарова мережа - мережа, що складається з одного шару.

Багатошарова мережа – мережа, що має кілька шарів.

У багатошаровій мережі перший шар називається вхідним, наступні – внутрішніми або прихованими, останній шар – вихідним. Таким чином, проміжні шари – це всі верстви в багатошарової нейронної мережі, крім 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 організовано вихідний шар мережі, видає результати роботи нейронної мережі**.**

У повнозв'язних мережах кожен нейрон передає свій вихідний сигнал іншим нейронам, включаючи самого себе. Вихідними сигналами мережі можуть бути всі або деякі вихідні сигнали нейронів після кількох тактів функціонування мережі. Всі вхідні сигнали подаються всім нейронам.

Навчання нейронних мереж

Найважливішим властивістю нейронних мереж є їх здатність навчатися на основі даних навколишнього середовища і в результаті навчання підвищувати свою продуктивність. Підвищення продуктивності відбувається з часом відповідно до певних правил. Навчання нейронної мережі відбувається

за допомогою інтерактивного процесу коригування синаптичних ваг і порогів. В ідеальному випадку нейронна мережа отримує знання про навколишнє середовище на кожній ітерації процесу навчання.

Навчання – це процес, в якому вільні параметри нейронної мережі настроюються за допомогою моделювання середовища, в яку ця мережа вбудована. Тип навчання визначається способом підстроювання цих параметрів.

Це визначення процесу навчання нейронної мережі передбачає наступну послідовність подій:

1. У нейронну мережу надходять стимули із зовнішнього середовища.

2. В результаті першого пункту змінюються вільні параметри нейронної мережі.

3. Після зміни внутрішньої структури нейронна мережа відповідає на порушення вже іншим чином.

Вищевказаний список чітких правил вирішення проблеми навчання нейронної мережі називається алгоритмом навчання. Нескладно здогадатися, що не існує універсального алгоритму навчання, відповідного для всіх архітектур нейронних мереж. Існує лише набір засобів, представлений безліччю алгоритмів навчання, кожен з яких має свої переваги. Алгоритми навчання відрізняються один від одного способом налаштування синаптичних ваг нейронів. Ще однією відмінною характеристикою є спосіб зв’язку навченою нейронної мережі з зовнішнім світом. У цьому контексті говорять про парадигму навчання, пов’язаної з моделлю навколишнього середовища, в якому функціонує дана нейронна мережа.

Існують два концептуальних підходи до навчання нейронних мереж: навчання з учителем і навчання без учителя.

Навчання нейронної мережі з учителем передбачає, що для кожного вхідного вектора з навчальної множини існує необхідне значення вихідного вектора, званого цільовим. Ці вектора утворюють навчальну пару. Ваги мережі змінюють доти, поки для кожного вхідного вектора не буде отриманий прийнятний рівень відхилення вихідного вектора від цільового.

Навчання нейронної мережі без вчителя є набагато більш правдоподібною моделлю навчання з точки зору біологічних коренів штучних нейронних мереж. Навчальна множина складається лише з вхідних векторів. Алгоритм навчання нейронної мережі підлаштовує ваги мережі так, щоб виходили узгоджені вихідні вектори, тобто щоб пред’явлення досить близьких вхідних векторів давало однакові виходи.

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 контрольному безлічі.

Процес навчання здійснюється на навчальній вибірці.

Існують два концептуальних підходи до навчання нейронних мереж: навчання з учителем і навчання без учителя.

Навчання нейронної мережі з учителем передбачає, що для кожного вхідного вектора з навчальної множини існує необхідне значення вихідного вектора, званого цільовим. Ці вектора утворюють навчальну пару. Ваги мережі змінюють доти, поки для кожного вхідного вектора не буде отриманий прийнятний рівень відхилення вихідного вектора від цільового.

Навчання нейронної мережі без вчителя є набагато більш правдоподібною моделлю навчання з точки зору біологічних коренів штучних нейронних мереж. Навчальна множина складається лише з вхідних векторів. Алгоритм навчання нейронної мережі підлаштовує ваги мережі так, Навчальна вибірка включає вхідні значення і відповідні їм вихідні значення набору даних. В ході навчання нейронна мережа знаходить якісь залежності вихідних полів від вхідних.

Таким чином, перед нами постає питання - які вхідні поля (ознаки) нам необхідно використовувати. Спочатку вибір здійснюється евристичний, далі кількість входів може бути змінено.

Складність може викликати питання про кількість спостережень в наборі даних. І хоча існують якісь правила, що описують зв’язок між необхідною кількістю спостережень і розміром мережі, їх вірність не доведена.

Кількість необхідних спостережень залежить від складності розв’язуваної задачі. При збільшенні кількості ознак кількість спостережень зростає нелінійно, ця проблема носить назву "прокляття розмірності". При недостатній кількості даних рекомендується використовувати лінійну модель.

Аналітик повинен визначити кількість шарів у мережі і кількість нейронів в кожному шарі.

Далі необхідно призначити такі значення ваг і зміщень, які зможуть мінімізувати помилку рішення. Ваги і зміщення автоматично налаштовуються таким чином, щоб мінімізувати різницю між бажаним і отриманим 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 нейронної мережі залежить її здатність вирішувати поставлені перед нею завдання.

Перенавчання нейронної мережі

При навчанні нейронних мереж часто виникає серйозна трудність, яка називається проблемою перенавчання (overfitting).

Перенавчання, або надмірно близька підгонка - зайве точну відповідність нейронної мережі конкретному набору навчальних прикладів, при якому мережу втрачає здатність до узагальнення.

Перенавчання виникає в разі занадто довгого навчання, недостатню кількість навчальних прикладів або переускладненою структури нейронної мережі.

Перенавчання пов’язано з тим, що вибір навчального (тренувального) безлічі є випадковим. З перших кроків навчання відбувається зменшення помилки. На наступних кроках з метою зменшення помилки (цільової функції) параметри підлаштовуються під особливості навчальної множини. Однак при цьому відбувається "підстроювання" не під загальні закономірності ряду, а під особливості його частини - навчає підмножини. При цьому точність прогнозу зменшується.

Один з варіантів боротьби з перенавчанням мережі - поділ навчальної вибірки на дві множини (навчальне і тестове).

На навчальній множині відбувається навчання нейронної мережі. На тестовому безлічі здійснюється перевірка побудованої моделі. Ці множини не повинні перетинатися.

З кожним кроком параметри моделі змінюються, однак постійне зменшення значення цільової функції відбувається саме на навчальній множині. При розбитті множини на два ми можемо спостерігати зміну помилки прогнозу на тестовому безлічі паралельно зі спостереженнями над навчальним безліччю. Якась кількість кроків помилки прогнозу зменшується на обох множинах. Однак на певному етапі помилка на тестовому безлічі починає зростати, при цьому помилка на навчальній множині продовжує зменшуватися. Цей момент вважається кінцем реального або справжнього навчання, з нього і починається перенавчання. Описаний процес проілюстровано на рис. 12.3

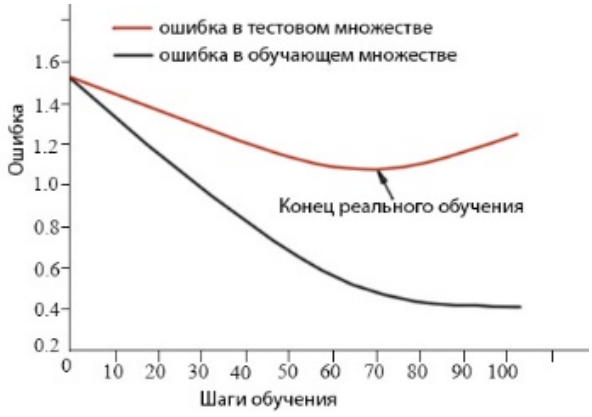


Рис. 12.3. Процес навчань мережі. Явище перенавчання

На першому кроці помилки прогнозу для навчального і тестового безлічі однакові. На наступних кроках значення обох помилок зменшуються, проте з семидесятого кроку помилка на тестовому безлічі починає зростати, тобто починається процес перенавчання мережі.

Прогноз на тестовому безлічі є перевіркою працездатності побудованої моделі. Помилка на тестовому безлічі може бути помилкою прогнозу, якщо тестове безліч максимально наближене до поточного моменту.

Моделі нейронних мереж

Розглянемо найбільш прості моделі нейронних мереж: одношаровий і багатошаровий персептрон.

Персептрон

Велика кількість моделей персептрона розглянуто в основній роботі Розенблатта. Найпростіша модель нейронної мережі – одношаровий персептрон.

Одношаровий персептрон (персептрон Розенблатта) – одношарова нейронна мережа, всі нейрони якої мають жорстку порогову функцію активації.

Одношаровий персептрон має простий алгоритм навчання і здатний вирішувати лише найпростіші завдання. Ця модель викликала до себе великий інтерес на початку 1960-х років і стала поштовхом до розвитку штучних нейронних мереж.

Персептрони складаються з одного шару (тобто кількість шарів нейронів між входом і виходом одно одному) штучних нейронів, з'єднаних за допомогою вагових коефіцієнтів з безліччю входів (див. Рис. 12.4).

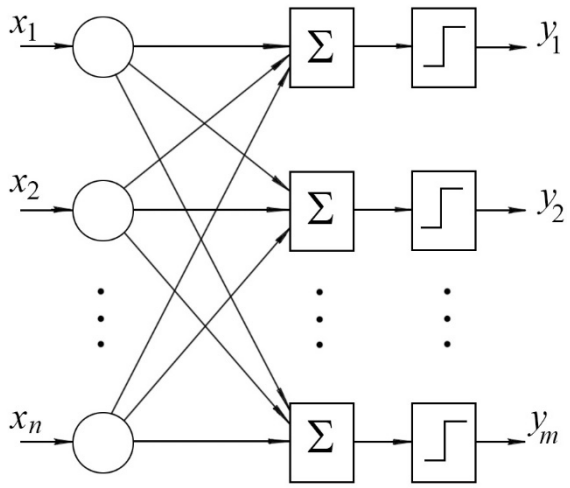


Рис. 12.4. Одношаровий персептрон

Вершини-кола в лівій частині малюнка служать лише для розподілу вхідних сигналів. Вони не виконують будь-яких обчислень, і тому не вважаються шаром. З цієї причини вони позначені у вигляді кола, щоб відрізняти їх від обчислює нейронів (суматорів), позначених квадратами.

У 60-ті роки персептрони викликали великий інтерес і оптимізм. Розенблатт довів чудову теорему про навчання персептронів. Уїдроу дав ряд переконливих демонстрацій систем персептронного типу, і дослідники в усьому світі прагнули вивчити можливості цих систем. Початкова ейфорія змінилася розчаруванням, коли виявилося, що персептрони не здатні навчитися вирішенню низки простих завдань. Мінський строго проаналізував цю проблему і показав, що є жорсткі обмеження на те, що можуть виконувати одношарові персептрони, і, отже, на те, чого вони можуть навчатися. Так як в той час методи навчання багатошарових мереж не були відомі, дослідники

перейшли в більш багатообіцяючі області, і дослідження одношарових персептронів прийшли в занепад. Нещодавнє відкриття методів навчання багатошарових мереж більшою мірою, ніж будь-який інший фактор, вплинуло на відродження інтересу і дослідницьких зусиль.

Робота Мінського, можливо, і охолодила запал ентузіастів персептрона, але забезпечила час для необхідної консолідації та розвитку теорії, яка лежить в його основі. Важливо відзначити, що аналіз Мінського не спростувують. Він залишається важливим дослідженням і має вивчатися, щоб помилки 60-х років не повторилися.

Теорія персептронів є основою для багатьох інших типів штучних нейронних мереж, а самі персептрони є логічною початковою точкою для вивчення штучних нейронних мереж.

Мережа, зображена на малюнку, має входи, на які надходять сигнали, що йдуть по синапсах до нейронів. Ці нейрони утворюють єдиний шар даної мережі і видають вихідні сигнали.

12.1. Представимість персептрона. Проблема XOR

Доведення теореми навчання персептрона показало, що персептрон здатний навчитися всьому тому, що він здатний представляти. Важливо при цьому вміти розрізняти представимість і здатність до навчання. Поняття представимості відноситься до здатності персептрона (або іншої мережі) моделювати певну функцію.

Для ілюстрації проблеми представимості допустимо, що є безліч карт, помічених цифрами від 0 до 9. Припустимо також, що існує гіпотетична машина, здатна відрізняти карти з непарним номером від карт з парним номером і вона запалює індикатор на своїй панелі при пред'явленні карти з непарним номером. Чи представима така машина персептроном? Тобто, чи може бути сконструйований персептрон і налаштовані його ваги (неважливо яким чином) так, щоб він мав таку ж роздільну здатністю? Якщо це так, то кажуть, що персептрон здатний представляти бажану машину. Можливості представлення одношаровими персептронами дуже обмежені. Є багато простих машин, які не можуть бути представлені персептроном незалежно від того, як налаштовуються його ваги. Для прикладу розглянемо проблему XOR.

Проблема функції *XOR*

Один з найбільш песимістичних результатів Мінського показує, що одношаровий персептрон не може відтворити таку просту функцію, як XOR. Це функція від двох аргументів, кожен з яких може бути нулем або одиницею. Вона приймає значення 1, коли один з аргументів дорівнює одиниці, але не обидва, інакше 0. Проблему можна проілюструвати за допомогою одношарової однонейронной системи з двома входами, показаної на рисунку нижче.

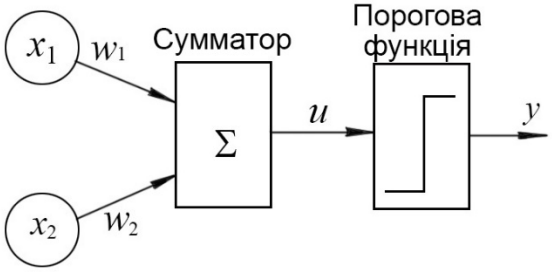


Рис. 12.5. Однонейронний персептрон

Позначимо один вхід через x_1, а інший через x_2, тоді всі їхні можливі комбінації будуть складатися з чотирьох точок на площині, як показано на рисунку 12.6.

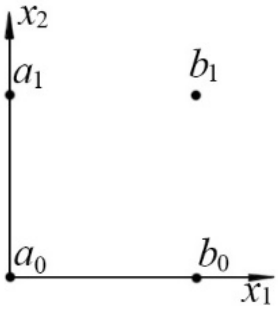


Рис. 12.6. Графік можливих станів

Таблиця нижче показує необхідну зв'язок між входами і виходом, де вхідні комбінації, які повинні давати нульовий вихід, помічені a_0 і a_1, одиничний вихід – b_0 і b_1.

Точки	Значення x_1	Значення x_2	Необхідний вихід
a_0	0	0	0
b_0	1	0	1
b_1	0	1	1
a_1	1	1	0

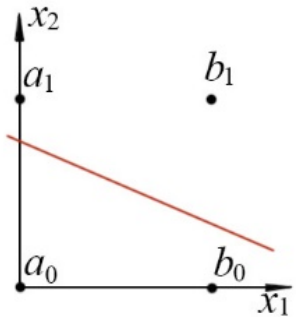
У моделі мережі, наведеної на рисунку вище, в якості активаційної функції застосовується звичайна порогова функція: y приймає значення нуль, коли u менше за 0.5, і одиниця в разі, коли u більше або дорівнює 0.5. Нейрон виконує наступне обчислення:

$$x_1 w_1 + x_2 w_2 = u$$

Ніяка комбінація значень двох ваг не може дати співвідношення між входом і виходом, заданого в таблиці вище. Щоб зрозуміти це обмеження, зафіксуємо u на величині порога 0.5. Мережа в цьому випадку описується наступним рівнянням:

$$x_1 w_1 + x_2 w_2 = 0.5$$

Це рівняння лінійне по x_1 і x_2 , тобто, всі значення по x_1 і x_2, що задовольняють цьому рівнянню, будуть лежати на деякій прямій у площині. Будь-які вхідні значення для x_1 і x_2 на цій лінії будуть давати порогове значення 0.5. Пряма розділяє площину на дві півплощини. Рівні введення в одній півплощині забезпечать значення u більше порога, отже $y = 1$. Рівні введення в другій напівплощині забезпечать значення u менше порогового значення, роблячи $y = 0$. Зміни значення w_1, w_2, і порога мінятимуть нахил і розміщення прямої. Для того, щоб мережа реалізувала функцію – виключає XOR, задану таблицею вище, потрібно розташувати пряму так, щоб точки a були з одного боку прямої, а точки b – з іншого. Спробувавши намалювати таку пряму на малюнку нижче, переконуємося, що це неможливо. Це означає, що за будь-яких значень ваг і порога, мережа не здатна відтворити співвідношення між входом і виходом, необхідне для представлення функції.



Навчання персептрона. Дельта-правило

Здатність штучних нейронних мереж навчатися є їх найбільш інтригуючою властивістю. Подібно біологічним системам, які вони моделюють, ці нейронні мережі самі моделюють себе в результаті спроб досягти кращої моделі поведінки.

Навчання може бути з учителем або без нього. Для навчання з учителем потрібен «зовнішній» учитель, який оцінював би поведінку системи і керував її подальшими модифікаціями. При навчанні без учителя мережу шляхом самоорганізації робить необхідні зміни. Навчання персептрона є навчанням з учителем.

Персептрон навчають, подаючи велику кількість образів по одному на його вхід і підлаштовуючи ваги до тих пір, поки для всіх образів не буде досягнутий необхідний вихід.

Розглянемо модель персептрона (рис. 12.5).

Алгоритм навчання персептрона наступний:

1. Присвоїти синаптичним вагам деякі початкові значення. Наприклад, нулі.

2. Подати вхідний образ і обчислити. Якщо відповідь правильна, то переходять до кроку 4. Інакше до кроку 3.

3. Застосовуючи дельта-правило (див. нижче) обчислити нові значення синаптичних ваг.

4. Повторити кроки 2–4 даного алгоритму навчання персептрона поки мережа не стане видавати очікуваний вихід на векторах з навчальної вибірки або поки відхилення не стане нижче деякого порогу.

Таким чином, логіка навчання персептрона наступна: якщо сигнал персептрона при деякому образі вірний, то нічого коригувати не треба, якщо ж ні – проводиться коригування ваги. Правила коригування ваг таке:

1. Якщо y є невірним і дорівнює нулю, то необхідно збільшити ваги тих входів, на які була подана одиниця.
2. Якщо y є невірним і дорівнює одиниці, то необхідно зменшити ваги тих входів, на які була подана одиниця.

Пояснимо ці правила. Припустимо, що на вхід був поданий деякий навчальний двійковий вектор. Цьому вектору відповідає вихід дорівнює одиниці. І цей вихід неправильний. Тоді ваги, приєднані до одиничних входів, повинні бути зменшені, так як вони прагнуть дати невірний результат. Аналогічно, якщо деякій іншій навчального вектора відповідає неправильний вихід рівний нулю, то ваги, приєднані до одиничних входів, повинні бути вже зменшені.

Дельта-правило

Дельта-правило є математичною моделлю правил коригування ваг. Введемо величину, яка дорівнює різниці між необхідним і реальним виходом:

$$\delta = y^* - y$$

Тоді, ваги персептрона після корекції дорівнюватимуть:

$$w_j\left(i+1\right) = w_j\left(i\right) + \eta\delta x_j$$

де:

- i – номер поточної ітерації навчання персептрона;
- η – коефіцієнт швидкості навчання, дозволяє управляти середньою величиною зміни ваг;
- x_j – величина входу відповідна синаптичній вазі w_j. величина. Додавання величини в твір дозволяє уникнути зміна тих ваг, яким на вході відповідав нуль.
- Існує доведення збіжності цього алгоритму навчання персептрона за кінцеве число кроків. Модель персептрона зараз представляє більше історичну цінність, ніж практичну. Але саме на його прикладі вдається більш наочно показати деякі важливі принципи функціонування, в тому числі і навчання нейронних мереж.

12.2. Глибокі нейронні мережі

Ймовірно, архітектура глибоких нейронних мереж використовується зараз найбільш часто. Вона була запропонована ще в роботах Розенблатта і детально обговорюється майже у всіх підручниках з нейронних мереж. Зазвичай мережа складається з множини сенсорних елементів (вхідних вузлів), які утворюють вхідний шар; одного або декількох прихованих шарів обчислювальних нейронів і одного вихідного шару нейронів.

У літературі немає єдиного підходу щодо того, як рахувати кількість шарів в багатошарових нейронних мережах. Одні пропонують вважати число шарів, включаючи вхідний шар, інші – враховують, тільки шари, що виконують обчислення. Ми пропонуємо використовувати останнє визначення. Згідно з цим визначенням, глибока нейронна мережа на малюнку нижче розглядається як двошарова. Вхід розподільного шару вважається нульовим шаром.

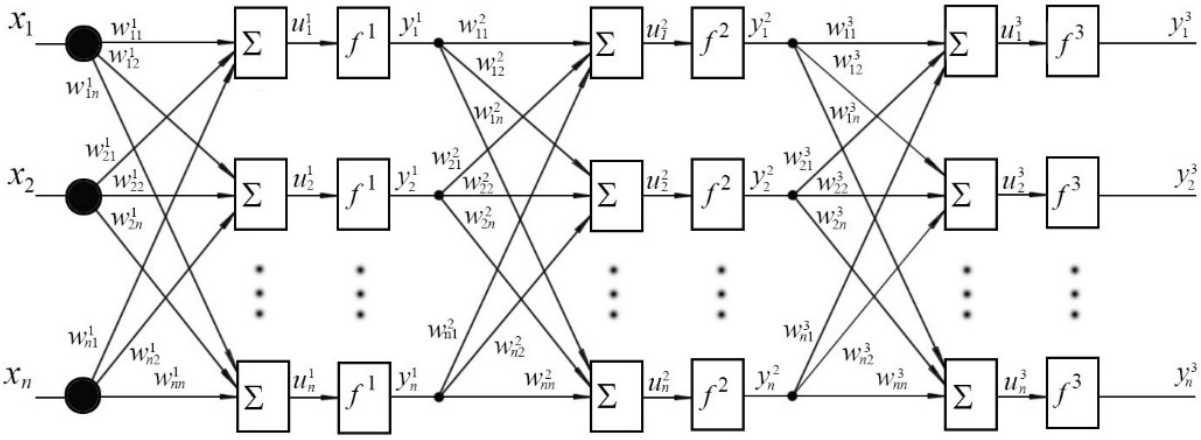


Рис. 12.7. Двошарова нейронна мережа

Глибока нейронна мережа може моделювати функцію практично будь-якого ступеня складності, причому число шарів і число елементів в кожному шарі визначають складність функції. Визначення числа проміжних шарів і числа елементів в них є важливим питанням при конструюванні.

Серед глибоких нейронних мереж можна виділити чотири найбільш значущих і важливих класи нейронних мереж:

мережі прямого поширення – всі зв’язки направлені строго від вхідних нейронів до вихідних. Такі мережі ще називають багатошаровим персептроном, за аналогією зі звичайним персептроном Розенблатта, в якому тільки один прихований шар;

реккурентні нейронні мережі або мережі зворотного поширення – сигнал в таких мережах з вихідних нейронів або нейронів прихованого шару частково передається назад на входи нейронів вхідного шару;

радіально базисні функції – вид багатошарової нейронної мережі, що має прихований шар з радіальних елементів і вихідний шар з лінійних елементів. Мережі цього типу досить компактні і швидко навчаються. Радіально базисна мережа володіє наступними особливостями: один прихований шар, тільки нейрони прихованого шару мають нелінійну

активаційну функцію і синаптичні ваги вхідного і прихованого шарів дорівнюють одиниці;

самоорганізовані карти або мережі Кохонена. Такий клас багатошарових нейронних мереж, як правило, навчається без учителя і успішно застосовується в задачах розпізнавання. Мережі такого класу здатні виявляти новизну у вхідних даних: якщо після навчання 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 має всього два шари: вхідний і вихідний, складений з радіальних елементів.

Багатошаровий персептрон

Багатошаровими персептронами називають глибокі нейронні мережі. Вхідний сигнал в таких мережах поширюється в прямому напрямку, від шару до шару. Багатошаровий персептрон в загальному уявленні складається з наступних елементів:

- множини вхідних вузлів, які утворюють вхідний шар;
- одного або декількох прихованих шарів обчислювальних нейронів;
- одного вихідного шару нейронів.

Багатошаровий персептрон представляє собою узагальнення одношарового персептрона Розенблатта. Прикладом багатошарового персептрона є наступна модель нейронної мережі:

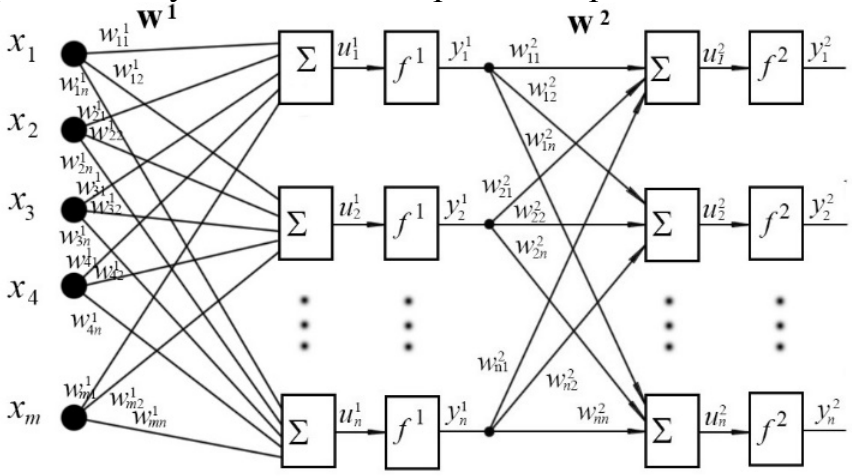


Рис. 12.8. Двошаровий персептрон

Мережа, зображена на рисунку, має *m* входів. На них надходять сигнали, що йдуть далі по синапсах на *n* нейронів, які утворюють перший шар. Вихідні сигнали першого шару передаються n нейронам другого шару. Останні, в свою чергу, видають два вихідних сигнали.

Кількість вхідних і вихідних елементів в багатошаровому персептрона визначається умовами завдання. Сумніви можуть виникнути щодо того, які вхідні значення використовувати, а які ні. Питання про те, скільки використовувати проміжних шарів і елементів в них, поки зовсім неясний. В

якості початкового наближення можна взяти один проміжний шар, а число елементів в ньому покласти рівним напівсумі числа вхідних і вихідних елементів.

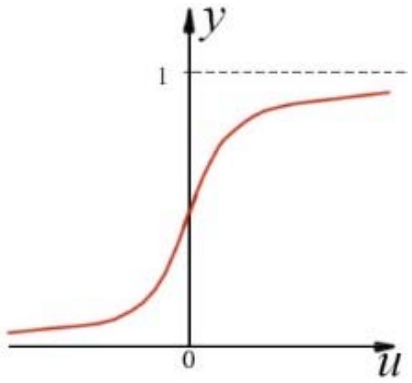
Багатошарові персептрони успішно застосовуються для вирішення різноманітних складних завдань і мають наступних три відмітних ознаки.

Властивість 1. Кожен нейрон мережі має нелінійну функцію активації

Важливо підкреслити, що така нелінійна функція повинна бути гладкою (тобто всюди диференціюється), на відміну від жорсткої порогової функції, використовуваної в персептрони Розенблатта. Найпопулярнішою формою функції, що задовольняє цій вимозі, є сигмоїдальна. Прикладом сигмоїдальної функції може служити логістична функція, що задається наступним виразом:

$$y = \frac{1}{1 + \exp\left(-\alpha u\right)}$$

де α – параметр нахилу сигмоїдальної функції. Змінюючи цей параметр, можна побудувати функції з різною крутизною.



Наявність нелінійності грає дуже важливу роль, так як в протилежному випадку відображення «вхід-вихід» мережі можна звести до звичайного одношарового персептрона.

Властивість 2. Кілька прихованих шарів

Багатошаровий персептрон містить один або кілька шарів прихованих нейронів, які не є частиною входу або виходу мережі. Ці нейрони дозволяють мережі навчатися вирішення складних завдань, послідовно отримуючи найбільш важливі ознаки з вхідного образу.

Властивість 3. Висока зв’язність

Багатошаровий персептрон має високу степінь зв’язності, що реалізовується за допомогою синаптичних з’єднань. Зміна рівня зв’язності мережі вимагає зміни великої кількості синаптичних з’єднань або їх вагових коефіцієнтів.

Комбінація всіх цих властивостей поряд зі здатністю до навчання на власному досвіді забезпечує обчислювальну потужність багатошарового персептрона. Однак ці ж якості є причиною неповноті сучасних знань про поведінку такого роду мереж: розподілена форма нелінійності і висока зв’язність мережі істотно ускладнюють теоретичний аналіз багатошарового персептрона.

12.3. Алгоритм зворотного поширення помилки

Алгоритм зворотного поширення помилки є одним з методів навчання багатошарових нейронних мереж прямого поширення, які називають також багатошаровими персептронами або глибокими нейронними мережами. Багатошарові персептрони успішно застосовуються для вирішення багатьох складних завдань.

Навчання алгоритмом зворотного поширення помилки передбачає два проходи по всім шарах мережі: прямого і зворотного. При прямому проході вхідний вектор подається на вхідний прошарок нейронної мережі, після чого поширюється по мережі від шару до шару. В результаті генерується набір вихідних сигналів, який і є фактичною реакцією мережі на даний вхідний образ. Під час прямого проходу всі синаптичні ваги мережі фіксовані. Під час зворотного проходу всі синаптичні ваги налаштовуються відповідно до правила корекції помилок, а саме: фактичний вихід мережі віднімається з бажаного, в результаті чого формується сигнал помилки. Цей сигнал згодом поширюється по мережі в напрямку, протилежному напрямку синаптичних зв'язків. Звідси і назва - алгоритм зворотного поширення помилки. Синаптичні ваги налаштовуються з метою максимального наближення вихідного сигналу мережі до бажаного.

Розглянемо роботу алгоритму детальніше. Нехай необхідно навчити нейронну мереду, застосувавши алгоритм зворотного поширення помилки. На наведеному малюнку використані наступні умовні позначення:

- кожен шар нейронної мережі пронумерований від 0 до 2;
- всі нейрони кожного шару пронумеровані арабськими цифрами;
- - w_{12}^{1} - синаптична вага між нейрономі 1 шару 0 і нейроном 2 на шарі 1;
- - y_{1}^{2} - вихід нейрона 1 шару 2.

Як активаційна функція в багатошарових персептронах, як правило, використовується раніше розглянута сигмоїдальна активаційна функція. Змінюючи цей параметр α, можна побудувати функції з різною крутизною. Обмовимося, що для всіх наступних міркувань буде використовуватися саме логістична функція активації, представлена формулою вище.

Сигмоїд звужує діапазон зміни так, що значення лежить між нулем і одиницею. Глибокі нейронні мережі мають більшу потужність представлення, ніж одношарові, тільки в разі присутності нелінійності. Стискаюча функція забезпечує необхідну нелінійність. Насправді є безліч функцій, які могли б бути використані. Для алгоритму зворотного поширення помилки потрібно лише, щоб функція була всюди дифференційована. Сигмоїд задовольняє цій вимозі. Його додаткова перевага полягає в автоматичному контролі підсилення. Для слабких сигналів (тобто коли близько до нуля) крива вхід-вихід має сильний нахил, що дає велике підсилення. Коли величина сигналу стає більше, підсилення падає. Таким чином, великі сигнали сприймаються

мережею без насичення, а слабкі сигнали проходять по мережі без надмірного ослаблення.

Метою навчання мережі алгоритмом зворотного поширення помилки є таке налаштування її ваг, щоб додаток деякої множини входів призводив до необхідної множини виходів. Для стислості ці множини входів і виходів будуть називатися векторами. При навчанні передбачається, що для кожного вхідного вектора існує парний йому цільової вектор, що задає необхідний вихід. Разом вони називаються навчальною парою. Мережа навчається на багатьох парах.

Алгоритм зворотного поширення помилки наступний**:**

1. Ініціалізувати синаптичні ваги маленькими випадковими значеннями.
2. Вибрати чергову навчальну пару з навчальної множини; подати вхідний вектор на вхід мережі.
3. Обчислити вихід мережі.
4. Обчислити різницю між виходом мережі і необхідним виходом (цільовим вектором навчальної пари).
5. Відкоригувати ваги мережі для мінімізації помилки (див. нижче).
6. Повторювати кроки з 2 по 5 для кожного вектора навчальної множини до тих пір, поки помилка на всій множині не досягне прийнятного рівня.

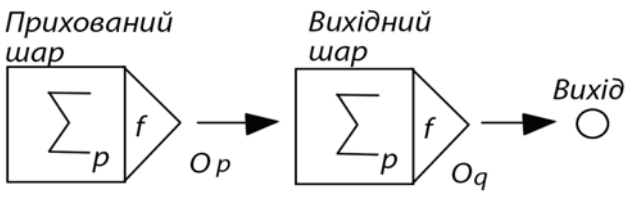
Операції, що виконуються кроками 2 і 3, подібно до тих, які виконуються при функціонуванні вже навченої мережі, тобто подається вхідний вектор і обчислюється одержаний вихід. Обчислення виконуються пошарово. На рис. 12.8 спочатку обчислюються виходи нейронів шару 1 (шар 0 вхідний, а значить ніяких обчислень в ньому не відбувається), потім вони використовуються в якості входів шару 2, обчислюються виходи нейронів шару 2, які і утворюють вихідний вектор мережі. Кроки 2 і 3 утворюють так званий «прохід вперед», так як сигнал поширюється по мережі від входу до виходу.

Кроки 4 і 5 складають «зворотний прохід», тут вираховується сигнал помилки поширюється назад по мережі і використовується для налаштування ваг.

Розглянемо більш детально 5 крок – коригування ваги мережі. Тут слід виділити два важливих випадки.

Випадок 1. Коригування синаптичних ваг вихідного шару

Наприклад, для моделі нейронної мережі на рис. 12.8, ваги мають такі позначення: w_{11}^2 і w_{21}^2. Визначимося, що індексом p будемо позначати нейрон, з якого виходить синаптична вага, а q – нейрон, в який входить:



Введемо величину δ , яка дорівнює різниці між необхідним T_q і реальним O_q виходами, помноженої на похідну логістичної функції активації :

$$\delta_q = O_q\left(1-O_q\right)\left(T_q - O_q\right)$$

Тоді, ваги вихідного шару після корекції дорівнюватимуть:

$$w_{p-q}\left(i+1\right) = w_{p-q}\left(i\right) + \eta\delta_q O_p$$

де:

- i – номер поточної ітерації навчання;
- w_{p-q} – величина синаптического ваги, що з'єднує нейрон p з нейроном q;
- η – коефіцієнт «швидкості навчання», дозволяє управляти середньою величиною зміни ваг;
- O_p – вихід нейрона p .

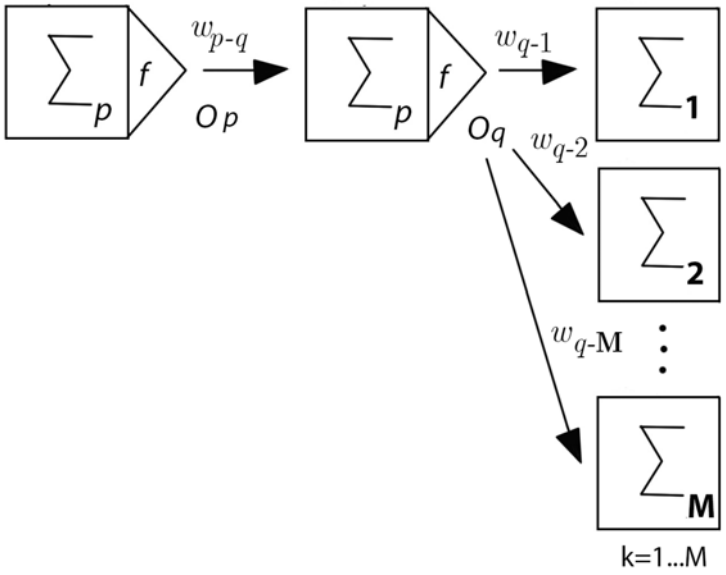
Наведемо приклад обчислень для синаптичного вагового коефіцієнта $w_{B_1-C_1}$:

$$\delta_{C_1} = O_{C_1}\left(1-O_{C_1}\right)\left(T_{C_1} - O_{C_1}\right)$$

$$w_{B_1-C_1}\left(i+1\right) = w_{B_1-C_1}\left(i\right) + \eta\delta_{C_1}O_{B_1}$$

Випадок 2. Коригування синаптичних ваг прихованого шару

Для моделі нейронної мережі на рис. 12.8, це будуть ваги відповідні шарам A і B. Визначимося, що індексом p будемо позначати нейрон з якого виходить синаптична вага, а q – нейрон в який входить (зверніть увагу на появу нової змінної k):



Введемо величину δ_q , яка дорівнює:

$$\delta_q = O_q\left(1-O_q\right)\sum_{k=1}^{M}\delta_k w_{q-k}$$

Тоді, ваги прихованих шарів після корекції дорівнюватимуть:

$$w_{p-q}\left(i+1\right) = w_{p-q}\left(i\right) + \eta\delta_q O_p$$

Наведемо приклад обчислень для синаптичної ваги $w_{A_1-B_1}$:

$$\delta_{B_1} = O_{B_1}\left(1-O_{B_1}\right)$$

$$w_{A_1-B_1}\left(i+1\right) = w_{A_1-B_1}\left(i\right) + \eta\delta_{B_1} O_{A_1}$$

Для кожного нейрона в прихованому шарі має бути обчислено δ і налаштовані всі ваги, асоційовані з цим шаром. Цей процес повторюється шар за шаром у напрямку до входу, поки всі ваги не будуть скориговані.

Поява алгоритму зворотного поширення помилки стало знаковою подією в області розвитку нейронних мереж, так як він реалізує обчислювально ефективний метод навчання багатошарового персептрона. Буде неправильно стверджувати, що алгоритм зворотного поширення помилки пропонує дійсно оптимальне рішення всіх потенційно вирішуваним проблем, проте він розвіяв песимізм щодо навчання багатошарових машин, що панував в результаті публікації Мінського в 1969 році.

Метод зворотного поширення помилки (Back propagation) – алгоритм навчання багатошарових персептронів, заснований на обчисленні градієнта функції помилок. В процесі навчання ваги нейронів кожного шару нейромережі коригуються з урахуванням сигналів, що надійшли з попереднього шару, і невязки кожного шару, яка обчислюється рекурсивно в зворотному напрямку від останнього шару до першого.

Запитання до розділу 12

1. Основна модель штучного нейрона та нейронних мереж. Одношарова та багатошарова нейронні мережі.
2. Основні умови перенавчання штучних нейронних мереж.
3. Проблема представимості персептрона. Суть проблеми реалізації функції XOR.
4. Принципи роботи та базові структури глибоких нейронних мереж.
5. Опишіть основні етапи алгоритму зворотного поширення помилки.

Розділ 13
Кластеризація й візуалізація

У багатьох прикладних задачах вимірювати ступінь подібності об'єктів суттєво простіше, ніж формувати описи ознак. Наприклад, два рядки з описом молекул ДНК або протеїнів набагато легше порівняти безпосередньо один з одним, ніж перетворювати кожен з них у вектор ознак, і потім порівнювати ці вектори. Те ж саме можна сказати про тексти, часові ряди або растрові зображення.

Задачу класифікації об'єктів на основі їх подібності один одному, коли приналежність навчальних об'єктів яким-небудь класам не задана, називають *задачею кластеризації*. В 1 розглянемо статистичні, ієрархічні й графові алгоритми кластеризації. В 3 – методи багатовимірного шкалування, які дозволяють відновлювати описи ознак об'єктів по матриці попарних відстаней між ними.

13.1. Алгоритми кластеризації

Задача кластеризації (або навчання без учителя) полягає в наступному. Є навчальна вибірка $X_m = \{x_1, x_2, ..., x_m\} \subset X$ та функція відстані між об'єктами $\rho(x, x')$. Потрібно розбити вибірку на підмножини, що не перетинаються, названі кластерами, так, щоб кожний кластер складався з об'єктів, близьких за метрикою ρ, а об'єкти різних кластерів суттєво відрізнялися. При цьому кожному об'єкту $x_i \in X_m$ приписується мітка (номер) кластера y_i.

Алгоритм кластеризації – це функція $a: X \to Y$, яка будь-якому об'єкту $x \in X$ ставить у відповідність мітку кластера $y \in Y$. Множина міток Y у деяких випадках відома заздалегідь, однак частіше ставиться задача визначити оптимальну кількість кластерів, з огляду на той або інший критерій якості кластеризації.

Розв'язок задачі кластеризації принципово неоднозначний з декількох 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 здійснюють достатньо розумну кластеризацію «за побудовою». Усі вони можуть давати різні результати. По-друге, кількість кластерів, як правило, невідома заздалегідь і встановлюється відповідно до деякого суб'єктивного критерію. По-третє, результат кластеризації суттєво залежить від метрики ρ, вибір якої, як правило, також є суб'єктивним і визначається експертом.

Кластеризація (навчання без учителя) відрізняється від класифікації (навчання з учителем) тим, що мітки вхідних об'єктів y_i спочатку не задані, і навіть може бути невідомою сама множина Y. У цьому сенсі задача кластеризації ще більшою мірою некоректно поставлена, ніж задача класифікації.

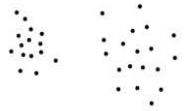
Цілі кластеризації можуть бути різними залежно від особливостей конкретної прикладної задачі:

1. Спростити подальшу обробку даних, розбити множину X_m на групи схожих об'єктів, щоб працювати з кожною групою окремо (задачі класифікації, регресії, прогнозування).
2. Скоротити обсяг збережених даних, залишивши по одному представнику від кожного кластера (задачі стискання даних).
3. Виділити нетипові об'єкти, які не підходять до жодного з кластерів (задачі однокласової класифікації).
4. Побудувати ієрархію множини об'єктів (задачі таксономії).

У першому випадку кількість кластерів намагаються зробити найменшою. У другому випадку більш важливо забезпечити високий ступінь подібності об'єктів усередині кожного кластера, а кластерів може бути скільки завгодно. У третьому випадку найбільший інтерес представляють окремі об'єкти, що не вписуються в жоден з кластерів.

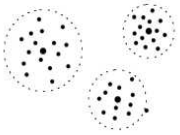
У всіх цих випадках може застосовуватися ієрархічна кластеризація, коли великі кластери дроблять на більш дрібні, ті у свою чергу дроблять на ще дрібніші, і т. д. Такі задачі називають *задачами таксономії* (taxonomy). Результатом таксономії є не проста розбивка множини об'єктів на кластери, а деревоподібна ієрархічна структура. Замість номера кластера об'єкт характеризують перерахуванням усіх кластерів, яким він належить, від великого до найменшого. Класичним прикладом таксономії на основі подібності є систематизація живих істот, запропонована Карлом Ліннеєм у середині XVIII століття. У сучасному вигляді біологічна ієрархія має близько 30 рівнів, 7 з них вважаються основними: царство, тип, клас, ряд, родина, рід, вид. Таксономії будують в багатьох областях знання, щоб упорядкувати інформацію про велику кількість об'єктів. Ми будемо розглядати алгоритми *ієрархічної кластеризації*, які дозволяють автоматизувати процес побудови таксономій.

Типи кластерних структур. Спробуємо скласти реєстр різних типів кластерних структур, які можуть виникати в практичних задачах.

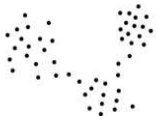


Згущення: внутрішньокластерні відстані, як правило, менші за міжкластерні.

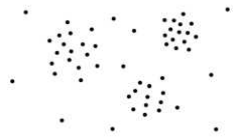
Стрічки: для будь-якого об'єкта існує близький до нього об'єкт того ж кластера, у той же час існують об'єкти одного кластера, які не є близькими.



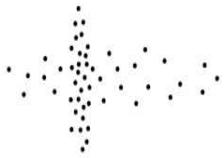
Кластери з центром: у кожному кластері існує об'єкт, такий, що майже всі об'єкти кластера лежать усередині кулі з центром у цьому об'єкті.



Кластери можуть з'єднуватися перемичками, що утрудняє роботу багатьох алгоритмів кластеризації.



Кластери можуть накладатися на розріджене тло з невеликою кількістю нетипових об’єктів.



Кластери можуть перекриватися.

Кластери можуть утворюватися не за принципом подібності, а за будь-якими іншими, заздалегідь невідомими, властивостями об’єктів. Стандартні методи кластеризації тут незастосовні.

Кластери можуть взагалі бути відсутніми. У цьому випадку потрібно застосовувати не кластеризацію, а інші методи аналізу даних.

Різні алгоритми кластеризації можуть бути більш-менш успішними в цих ситуаціях. Прості алгоритми, як правило, вузько спеціалізовані й дають адекватні результати тільки в одній-двох ситуаціях. Більш складні алгоритми, такі як FOREL або агломеративна процедура Ланса-Вільямса, справляються з декількома типами ситуацій. Однак створення алгоритму, що успішно працює у всіх ситуаціях без винятку, є важкою та навряд чи розв’язною задачею.

13.1.1 Евристичні графові алгоритми

Великий клас алгоритмів кластеризації заснований на представленні вибірки у вигляді графа. Вершинам графа відповідають об’єкти вибірки, а ребрам – попарні відстані між об’єктами $\rho_{ij} = \rho\left(x_i, x_j\right)$.

Перевагою графових алгоритмів кластеризації є наочність, відносна простота реалізації, можливість вносити різні вдосконалення, спираючись на прості геометричні міркування.

Алгоритм виділення зв’язних компонентів. Задають параметр R і в графі видаляють всі ребра (i, j), для яких $\rho_{ij} > R$. З’єднаними залишають тільки найбільш близькі пари об’єктів. Ідея алгоритму полягає в тому, щоб підібрати таке значення $R \in \left[\min \rho_{ij}, \max \rho_{ij}\right]$, за якого граф розвалиться на кілька зв’язних компонентів. Знайдені зв’язні компоненти і є кластери.

Зв’язним компонентом графа називають підмножину його вершин, у якій будь-які дві вершини можна з’єднати шляхом, що цілком лежить у цій підмножині. Для пошуку зв’язних компонент можна використовувати стандартні алгоритми пошуку в ширину (алгоритм Дейкстри) або пошуку в глибину.

Алгоритм 1. Алгоритм найкоротшого незамкнутого шляху (КНП)
1: Знайти пару точок (i, j) з найменшим ρ_{ij} і з'єднати їх ребром;
2: **поки** у вибірці залишаються ізольовані точки
3: знайти ізольовану точку, найближчу до деякої неізольованої;
4: з'єднати ці дві точки ребром;
5: вилучити $K-1$ найдовших ребер;

Для підбору параметра R зазвичай рекомендують побудувати гістограму розподілу попарних відстаней ρ_{ij}. У задачах з вираженою кластерною структурою ця гістограма має два чітких піка: зона невеликих внутрішньокласових відстаней і зона великих міжкласових відстаней. Параметр R задають як відстань, яка відповідає точці мінімуму між цими піками.

Відмітимо два недоліки цього алгоритму.

1. Обмежена застосовність. Алгоритм виділення зв'язних компонент найбільше підходить для виділення кластерів типу згущення або стрічок. Наявність розрідженого фону або «вузьких перемичок» між кластерами призводить до неадекватної кластеризації.

2. Погана керованість числом кластерів. Для багатьох додатків зручніше задавати не параметр R, а кількість кластерів або деякий поріг «чіткості кластерізації». Управляти числом кластерів за допомогою параметра R досить важко. Доводиться багаторазово розв'язувати задачу при різних R, що негативно позначається на часових витратах.

Алгоритм найкоротшого незамкнутого шляху будує граф з $m-1$ ребер так, щоб вони з'єднували всі m точок і мали мінімальну сумарну довжину. Такий граф називають *найкоротшим незамкнутим шляхом* (КНП), мінімальним деревом покриття або каркасом. Доведено, що цей граф можна побудувати за допомогою нескладної процедури, яка відповідає крокам 1–4 Алгоритму 1. На кроці 5 видаляються $K-1$ найдовших ребер, і зв'язний граф розпадається на K кластерів.

На відміну від попереднього алгоритму, число кластерів K задають як вхідний параметр. Його можна також визначати графічно, якщо упорядкувати всі відстані, що утворюють каркас, в порядку спадання, і відкласти їх на графіку. Різкий стрибок вниз десь на початковій (лівій) ділянці графіка покаже кількість кластерів, які виділяються найбільш чітко.

Цей алгоритм, як і попередній, дуже простий і також має обмежену застосовність. Наявність розрідженого фону або перемичок призводить до неадекватної кластеризації. Іншим недоліком КНП є висока трудомісткість – для побудови найкоротшого незамкнутого шляху потрібно $O(m^3)$ операцій.

Алгоритм FOREL (формальні елементи) запропонований Загоруйко та Йолкіною в 1967 році при розв'язуванні однієї прикладної задачі в області

палеонтології. Алгоритм має чисельні варіації, в основі яких лежить наступна базова процедура.

Нехай задана деяка точка $x_0 \in X$ і параметр R. Виділяють всі точки вибірки $x_i \in X_m$, що потрапляють усередину сфери $\rho(x_i, x_0) \le R$, і точку x_0 переносять в центр ваги виділених точок. Ця процедура повторюється доти, поки склад виділених точок, а отже, і положення центру, не перестане змінюватись. Доведено, що ця процедура сходиться за скінченне число кроків. При цьому сфера переміщається в місце локального згущення крапок. Центр сфери x_0 у загальному випадку не є об'єктом вибірки, тому його називають *формальним елементом*.

Для обчислення центру необхідно, щоб множина об'єктів X була не тільки метричним, але й лінійним векторним простором. Ця вимога природно виконується, коли об'єкти описуються числовими ознаками. Однак існують задачі, у яких початково задана тільки метрика, а додавання й множення на число не визначені на X. Тоді як центр сфери можна взяти той об'єкт навчальної вибірки, для якого середня відстань до інших об'єктів кластера є мінімальною. Відповідно, крок 6 заміняється на $x_0 := \arg\min_{x \in K_0} \sum_{x' \in K_0} \rho(x, x')$.

При цьому помітно збільшується трудомісткість алгоритму. Якщо в лінійному просторі для обчислення центру потрібно $O(\eta)$ операцій, то в метричному – $O(\eta^2)$, де η – кількість точок у кластері. Алгоритм можна дещо прискорити, якщо помітити, що переобчислення центру при додаванні або видаленні окремої точки кластера вимагає лише $O(\eta)$ операцій, а в лінійному просторі – $O(1)$.

Різні варіанти алгоритму FOREL відрізняються способами об'єднання сфер у кластери, способами варіювання параметра R, способами вибору початкового наближення для точок x_0. В Алгоритмі 2 представлено один з варіантів, у якому сфери будуються послідовно. На кроці 9 до центрів цих сфер застосовується алгоритм КНП. З одного боку, це вирішує проблему низької ефективності КНП, тому що сфер набагато менше, ніж початкових об'єктів. З іншого боку, ми одержуємо більш тонку, дворівневу, структуру кластерів: кожний кластер верхнього рівня розпадається на більш дрібні підкластери нижнього рівня.

Інша перевага цього алгоритму – можливість описувати кластери довільної геометричної форми. Варіюючи параметр R, можна одержувати кластеризації різного ступеня деталізації. Якщо кластери близькі за формою до куль, можна зробити R істотно більшим. Для опису кластерів більш складної форми слід зменшувати R.

Алгоритм 2 досить чутливий до вибору початкового положення точки x_0 для кожного нового кластера. Для усунення цього недоліку потрібно генерувати декілька (10…20) кластеризацій. Оскільки початкове положення центрів вибирається випадковим чином, ці кластеризації будуть досить сильно

відрізнятися. Остаточно вибирається та кластеризація, яка надає найкраще значення заданому функціоналу якості.

Різні види функціоналів якості розглянемо далі.

Алгоритм 2. Алгоритм FOREL
1: Ініціалізувати множину некластеризованих точок:
$U := X_m$;
2: **поки** у вибірці є некластеризовані точки, U = 0:
3: взяти довільну точку $x_0 \in U$ випадковим чином;
4: **повторювати**
5: утворювати кластер – сферу з центром у x_0 й радіусом R:
$K_0 := \{x_i \in U \mid \rho(x_i, x_0) \le R\}$;
6: помістити центр сфери в центр мас кластера:
$x_0 := \frac{1}{|K_0|} \sum_{x_i \in K_0} x_i$;
7: **поки** центр x_0 не стабілізується;
8: позначити всі точки K_0 як кластеризовані:
$U := U \setminus K_0$;
9: застосувати алгоритм КНП до множини центрів усіх знайдених кластерів;
10: кожний об'єкт $x_i \in X_m$ приписати кластеру з найближчим центром.

13.1.2. Функціонали якості кластеризації

Задачу кластеризації можна ставити як задачу дискретної оптимізації: необхідно так приписати номери кластерів y_i об'єктам x_i, щоб значення обраного функціонала якості набуло найкращого значення. Існує багато різновидів функціоналів якості кластеризації, але немає «самого правильного» функціонала. Кожний метод кластеризації можна розглядати як точний або наближений алгоритм пошуку оптимуму деякого функціонала.

Середня внутрішньокластерна відстань повинна бути найменшою:

$$F_0 = \frac{\sum_{i<j} [y_i = y_j] \rho(x_i, x_j)}{\sum_{i<j} [y_i = y_j]} \to \min$$

Середня міжкластерна відстань повинна бути якнайбільшою:

$$F_1 = \frac{\sum_{i<j} [y_i \neq y_j] \rho(x_i, x_j)}{\sum_{i<j} [y_i \neq y_j]} \to \max$$

Якщо алгоритм кластеризації обчислює центри кластерів $\mu_y, y \in Y$, то можна визначити функціонали, обчислювально більш ефективні.

Сума середніх внутрішньокластерних відстаней повинна бути якнайменша:

$$\Phi_0 = \sum_{y \in Y} \frac{1}{|K_y|} \sum_{i: y_i = y} \rho^2 \left(x_i, \mu_y \right) \to \min$$

де $K_y = \left\{ x_i \in X^m \mid y_i = y \right\}$ – кластер з номером y . У цій формулі можна було б брати не квадрати відстаней, а самі відстані. Однак, якщо ρ – евклідова метрика, те внутрішня сума у Φ_0 набуває фізичного сенсу моменту інерції кластера K_y відносно його центру мас, якщо розглядати кластер як матеріальне тіло, що складається з $|K_y|$ точок однакової маси.

Сума міжкластерних відстаней повинна бути якнайбільша:

$\Phi_1 = \sum_{y \in Y} \rho^2 \left(\mu_y, \mu \right) \to \max$, де μ – центр мас усієї вибірки.

На практиці обчислюють відношення пари функціоналів, щоб урахувати як міжкластерні, так і внутрішньокластерні відстані:

$F_0 / F_1 \to \min$, або $\Phi_0 / \Phi_1 \to \min$.

Алгоритм 3. Кластеризація за допомогою ЕМ-алгоритму

1: початкове наближення для всіх кластерів $y \in Y$;

$\omega_y := 1/|Y|$

$\mu_y :=$ випадковий об’єкт вибірки;

$$\sigma_{yj}^2 := \frac{1}{m|Y|} \sum_{i=1}^{m} \left(f_j \left(x_i \right) - \mu_{yj} \right)^2, j = 1,2,...,n$$

2: **повторювати**

3: E – крок (expectation)

$$g_{iy} := \frac{\omega_y p_y \left(x_i \right)}{\sum_{z \in Y} \omega_z p_z \left(x_i \right)}, y \in Y, i = 1,2,...,m;$$

4: M – крок (maximization)

$$\omega_y := \frac{1}{m} \sum_{i=1}^{m} g_{ij}, y \in Y;$$

$$\mu_{yj} := \frac{1}{m\omega_y} \sum_{i=1}^{m} g_{iy} f_j \left(x_i \right), y \in Y, j = 1,2,...,n$$

$$\sigma_{yj}^2 := \frac{1}{m\omega_y} \sum_{i=1}^{m} g_{iy} \left(f_j \left(x_i \right) - \mu_{yj} \right)^2, y \in Y, j = 1,2,...,n$$

5: Віднести об’єкти до кластерів за байєсовським правилом обчислень:

$$y_i := \arg\max_{y \in Y} g_{iy}, i = 1,2,...,m;$$

6: **поки** y_i не перестануть змінюватись.

13.1.3. Статистичні алгоритми

Статистичні алгоритми засновані на припущенні, що кластери непогано описуються деяким сімейством імовірнісних розподілів. Тоді задача кластеризації зводиться до поділу суміші розподілів по кінцевій вибірці.

Знову ЕМ-алгоритм. Нагадаємо основні гіпотези байєсовського підходу до поділу сумішей імовірнісних розподілів.

Гіпотеза 1 (про імовірнісну природу даних). *Об'єкти вибірки* X_m *з'являються випадково й незалежно згідно з імовірнісним розподілом, що представляє собою суміш розподілів*

$$p(x)=\sum_{y\in Y}\omega_y p_y(x),\sum_{y\in Y}\omega_y=1,$$

де $p_y(x)$ *– функція щільності розподілу кластера* y, ω_y *– невідома апріорна імовірність появи об'єктів із кластера* y.

Конкретизуючи вид розподілів $p_y(x)$, найчастіше беруть сферичні гаусівські щільності. Будемо вважати, що кластери схожі скоріше не на кулі, а на еліпсоїди, осі яких спрямовані вздовж осей координат. Перевага еліптичних гаусианів у тому, що вони обходять проблему вибору нормування ознак. Нормувати можна дещо по-різному, причому результат кластеризації суттєво залежить від нормування. Поки не зроблена кластеризація, важко зрозуміти, яке нормування краще. При використанні еліптичних гаусианів оптимальне нормування підбирається самим алгоритмом кластеризації, індивідуально для кожного кластера.

Гіпотеза 2 (про простір об'єктів і форму кластерів). Кожний об'єкт x з $X=\mathbb{R}^n$ описується n числовими ознаками: $x\equiv(f_1(x),f_2(x),...,f_n(x))$. Кожний кластер в $y\in Y$ описується n-вимірною гаусівською щільністю $p_y(x)$ з центром $\mu_y=(\mu_{y1},\mu_{y2},...,\mu_{yn})$ і діагональною матрицею коваріацій $\Sigma_y=diag(\sigma_{y1}^2,\sigma_{y2}^2,...,\sigma_{yn}^2)$:

$$p_y(x)=(2\pi)^{-\frac{n}{2}}(\sigma_{y1}\cdots\sigma_{yn})^{-1}\exp\left(-\frac{1}{2}\rho_y^2(x,\mu_y)\right),$$

де $\rho_y^2(x,x')=\sum_{j-1}^{n}\sigma_{yj}^{-2}\,|f_j(x)-f_j(x')|^2$ – зважена евклідова відстань з вагами σ_{yj}^{-2}.

За цих припущень задача кластеризації збігається з задачею поділу суміші імовірнісних розподілів, і для її розв'язку можна застосувати ЕМ-алгоритм. Для оцінювання параметрів кластерів скористаємося формулами, отриманими у відповідній теоремі саме для випадку еліптичних гаусіанів. Реалізація цієї ідеї представлена в Алгоритмі 3.

Нагадаємо, що ЕМ-алгоритм полягає в ітераційному повторенні двох кроків. На Е-кроці за формулою Байеса обчислюються приховані змінні g_{iy}. Значення g_{iy} дорівнює апостеріорній імовірності того, що об'єкт $x_i \in X_m$ належить кластеру $y \in Y$. На М-кроці уточнюються параметри кожного кластера (μ_y, Σ_y), при цьому суттєво використовуються приховані змінні g_{iy}.

В Алгоритмі 3 для простоти передбачається, що число кластерів відомо заздалегідь. Однак у більшості практичних випадків його краще визначати автоматично.

Метод *k*-середніх, представлений в Алгоритмі 4, є спрощенням ЕМ--алгоритму. Головна відмінність у тому, що в ЕМ-алгоритмі кожний об'єкт x_i розподіляється по всіх кластерах з імовірностями $g_{iy} = P\{y_i = y\}$. В алгоритмі k-середніх (k-means) кожний об'єкт жорстко приписується тільки до одного кластеру.

Друга відмінність у тому, що в k-means форма кластерів не настроюється. Однак ця відмінність не настільки принципова. Можна запропонувати спрощений варіант ЕМ, у якому форма кластерів також не буде настроюватися – для цього достатньо брати сферичні гаусіани з коваріаційними матрицями $\Sigma_y = \sigma_y I_n$. З іншого боку, можливий і узагальнений варіант k-means, у якому будуть визначатися розміри кластерів уздовж координатних осей. Для цього в Алгоритмі 3 достатньо забрати крок 5 і замінити Е-крок твердим приписуванням об'єктів кластерам:

$$y_i := \arg\min_{y \in Y} \rho_i\left(x_i, \mu_y\right), j = 1,2,...n$$

$$g_{iy} := \left[y_i = y\right], j = 1,2,...,n, y \in Y$$

Таким чином, ЕМ і k-means досить плавно «перетікають» один в інший, дозволяючи будувати різні «проміжні» варіанти цих двох алгоритмів.

Помітимо, що k-means схожий також на пошук центру кластера в алгоритмі FOREL. Відмінність у тому, що в FOREL кластер – це куля заданого радіуса R, тоді як в k-means об'єкти відносять до кластерів за принципом найближчого сусіда.

Існує два «канонічні» варіанти алгоритму k-means. Варіант Болла- Холу представлено в Алгоритмі 4. Варіант Мак Кіна відрізняється тим, що щоразу, коли деякий об'єкт x_i переходить із одного кластера в інший, центри обох кластерів перераховуються. Для цього крок 4 треба перенести усередину циклу по i, виконуваного на кроці 3. Мак Кін показав в 1967 році, що цей варіант алгоритму приводить до локального мінімуму функціонала Φ_0.

Алгоритм k-means вкрай чутливий до вибору початкових наближень центрів. Випадкова ініціалізація центрів на кроці 1 може приводити до поганих кластеризацій. Для формування початкового наближення можна виділити k найбільш віддалених точок вибірки: перші дві точки виділяються по максимуму всіх попарних відстаней; кожна наступна точка вибирається так, щоб відстань від неї до найближчої вже виділеної була максимальною.

Алгоритм 4. Кластеризація за допомогою алгоритму k -середніх

1: сформувати початкове наближення центрів всіх кластерів $y \in Y$:
μ_y – найбільш віддалені один від одного об'єкти вибірки;
2: **повторювати**
3: віднести кожен об'єкт до найближчого центру (аналог Е-кроку):
$y_i := \arg\min_{y \in Y} \rho(x_i, \mu_y), i = 1,2,...,m$
4: обчислити нове положення центрів (аналог М-кроку):
$$\mu_{yj} := \frac{\sum_{i=1}^{m} [y_i = y] f_j(x_i)}{\sum_{i=1}^{m} [y_i = y]}, y \in Y, j = 1,2,...,n$$
5: **поки** y_i не припинять змінюватись.

Інша рекомендація – виконати кластеризацію кілька раз, з різних випадкових початкових наближень і вибрати кластеризацію з найкращим значенням заданого функціонала якості.

Кластеризація може виявитися неадекватною й у тому випадку, якщо число кластерів буде початково невірно вгадане. Стандартна рекомендація – провести кластеризацію при різних значеннях k й вибрати те, при якому досягається різке поліпшення якості кластеризації за заданим функціоналом.

Кластеризація з частковим навчанням. Алгоритми EM і k -means легко пристосувати для розв'язування задач кластеризації з частковим навчанням (semi-supervised learning), коли для деяких об'єктів x_i відомі правильні класифікації $y^*(x_i)$. Позначимо через U підмножину таких об'єктів, $U \subset X_m$.

Прикладом такої задачі є рубрикація текстових документів, зокрема, сторінок в Інтернеті. Типова ситуація, коли є відносно невелика множина документів, вручну класифікованих за тематикою. Потрібно визначити тематику великої кількості некласифікованих документів. Подібність документів $\rho(x, x')$ може оцінюватися по-різному залежно від цілей рубрикації й специфіки самих документів: за частотою «зустрічальності» ключових слів, за частотою відвідуваності заданою множиною користувачів, за кількістю взаємних гіпертекстових посилань, або у інший спосіб.

Модифікація обох алгоритмів досить проста: на Е-кроці (крок 3) для всіх $x_i \in U$ вважаємо $g_{iy} := [y = y^*(x_i)]$, для всіх інших $x_i \in X_m \setminus U$ приховані змінні g_{iy} обчислюються як раніше. На практиці часткова класифікація навіть невеликої кількості об'єктів суттєво поліпшує якість кластеризації.

13.1.4. Ієрархічна кластеризація

Ієрархічні алгоритми кластеризації, називані також алгоритмами *таксономії*, будують не одне розбиття вибірки на класи, що не перетинаються, а систему вкладених розбивок. Результат таксономії звичайно представляється у вигляді таксономічного дерева – дендрограми. Класичним прикладом такого дерева є ієрархічна класифікація тварин і рослин.

Серед алгоритмів ієрархічної кластеризації різняться два основні типи. *Дивизимні* або спадні алгоритми розбивають вибірку на усе більш і більш дрібні кластери. Більш поширені *агломеративні* або висхідні алгоритми, у яких об'єкти поєднуються в усе більші і більші кластери. Реалізація цієї ідеї представлена в Алгоритмі 5.

Спочатку кожний об'єкт вважають окремим кластером. Для одноелементних кластерів природно визначається функція відстані

$$R\left(\{x\},\{x'\}\right)=\rho\left(x,x'\right)$$

Потім запускається процес злиття. На кожній ітерації замість пари найближчих кластерів U і V утворюється новий кластер $W=U\cup V$. Відстань від нового кластера W до будь-якого іншого кластера S обчислюється за відстанями $R\left(U,V\right), R\left(U,S\right)$ та $R\left(V,S\right)$, які до цього моменту вже повинні бути відомі:

$$R\left(U\cup V,S\right)=\alpha_U R\left(U,S\right)+\alpha_V R\left(V,S\right)+\beta R\left(U,V\right)+\gamma\left|R\left(U,S\right)-R\left(V,S\right)\right|,$$

де α_U, α_V, β,γ – числові параметри. Ця універсальна формула узагальнює практично всі розумні способи визначити відстань між кластерами. Вона була запропонована Лансом і Уильямсом в 1967 році.

На практиці використовуються наступні способи обчислення відстаней *R(W, S)* між кластерами *W* і *S*. Для кожного з них доведена відповідність формулі Ланса-Вільямса за певних комбінацій параметрів:

Відстань близького сусіда: $\alpha_U=\alpha_V=\frac{1}{2},\beta=0,\gamma=-\frac{1}{2}$;

$$R^{б}\left(W,S\right)=\min_{w\in W,s\in S}\rho\left(w,s\right);$$

Відстань далекого сусіда: $\alpha_U=\alpha_V=\frac{1}{2},\beta=0,\gamma=\frac{1}{2}$

$$R^{д}\left(W,S\right)=\max_{w\in W,s\in S}\rho\left(w,s\right)$$

Середня відстань: $\alpha_U=\frac{|U|}{|W|},\alpha_V=\frac{|V|}{|W|},\beta=\gamma=0$

$$R^{с}\left(W,S\right)=\frac{1}{|W||S|}\sum_{w\in W}\sum_{s\in S}\rho\left(w,s\right)$$

Відстань між центрами: $\alpha_U=\frac{|U|}{|W|},\alpha_V=\frac{|V|}{|W|},\beta=-\alpha_U\alpha_V,\gamma=0$

$$R^{ц}\left(W,S\right)=\rho^2\left(\sum_{w\in W}\frac{w}{|W|},\sum_{s\in S}\frac{s}{|S|}\right)$$

Відстань Уорда: $\alpha_U = \frac{|S|+|U|}{|S|+|W|}, \alpha_V = \frac{|S|+|V|}{|S|+|W|}, \beta = \frac{-|S|}{|S|+|W|}, \gamma = 0$

$$R^y(W,S) = \frac{|S||W|}{|S|+|W|}\rho^2\left(\sum_{w\in W}\frac{w}{|W|}, \sum_{s\in S}\frac{s}{|S|}\right)$$

Алгоритм 5. Агломеративна кластеризація Ланса-Вільямса
1: ініціалізувати множину кластерів C_1:
$t := 1; C_t = \{\{x_1\},\{x_2\},...,\{x_m\}\}$;
2: **для всіх** $t = 2,3,...,m$ (t – номер ітерації):
3: знайти в C_{t-1} два найближчі кластери:
$(U,V) := \arg\min_{U\neq V} R(U,V)$;
$R_t := R(U,V)$;
4: вилучити кластери U і V, додати злитий кластер $W = U \cup V$:
$C_t := C_{t-1} \cup \{W\} \setminus \{U,V\}$;
5: **для всіх S** G Ct
6: обчислити відстань $R(W,S)$ за формулою Ланса-Вільямса;

Можливих варіантів занадто багато, і на перший погляд усі вони видаються досить розумними. Виникає питання: якому з них надати перевагу? Розглянемо кілька додаткових властивостей, що характеризують якість кластеризації.

Властивість монотонності. Позначимо через R_t відстань між найближчими кластерами, вибраними на *t*-му кроці для злиття. Кажуть, що функція відстані *R* має властивість *монотонності*, якщо при кожному злитті відстань між кластерами, що об'єднуються, тільки збільшується: $R_2 \le R_3 \le ... \le R_m$.

Властивість монотонності дозволяє зобразити процес кластеризації у вигляді спеціального графіка, названого *дендрограма*. По вертикальній осі відкладаються об'єкти, по горизонтальній – відстані R_t. Неважко довести, що якщо кластеризація має властивість монотонності, то дендрограму можна побудувати так, щоб вона не мала самоперетинів. При цьому будь-який кластер з множини C_t представляється суцільною послідовністю точок на вертикальній осі. Якщо ж процес кластеризації йде не монотонно, то замість дендрограми отримаємо заплутаний клубок ліній, на якому важко що-небудь розібрати.

Дендрограма дозволяє уявити кластерну структуру у вигляді плоского графіка незалежно від того, яка розмірність початкового простору. Існують і інші способи двовимірної візуалізації багатовимірних даних, такі як багатовимірне шкалування або карти Кохонена, але вони привносять в картину штучні спотворення, вплив яких досить важко оцінити.

Виявляється, що не будь-яке поєднання коефіцієнтів у формулі Ланса-Вільямса приводить до монотонної кластеризації.

Теорема 1 (Мілліган, 1979). *Якщо виконуються наступні три умови, то кластеризація є монотонною:*

1) $\alpha_U \geq 0, \alpha_V \geq 0$;
2) $\alpha_U + \alpha_V + \beta \geq 1$
3) $\min\{\alpha_U, \alpha_V\} + \gamma \geq 0$

З перерахованих вище відстаней тільки $R^{ц}$ не є монотонною. Відстань Уорда відрізняється від неї мультиплікативною поправкою, яка і робить її монотонною.

13.1.5. Властивості розтягування і стискання

Деякі відстані мають *властивість розтягування*. У міру того, як кластер укрупнюється, відстані від нього до інших кластерів збільшуються, як ніби простір навколо кластера розтягується.

На дендрограмі розтягувальні відстані характеризуються підвищенням щільності ліній ліворуч, в області найменших значень R_t. Властивість розтягання вважається бажаною, тому що вона сприяє більш чіткому відділенню кластерів. З іншого боку, при занадто сильному розтягуванні можливо знайти кластери там, де їх на початку не було. Розтягувальними є відстані $R^{д}$ і $R^{У}$.

Деякі відстані, навпаки, мають *властивість стискання*. У міру росту кластера відстані від нього до інших кластерів зменшуються, і здається, що простір навколо кластера стискається. Природна кластеризація при цьому може зникнути. Для стискаючих відстаней характерне ущільнення дендрограми праворуч, в області найбільших значень R_t. Відстань близького сусіда $R^{б}$ є сильно стискаючою.

Властивості стискання й розтягування визначаються через відношення $R_t / \rho(\mu_U, \mu_V)$, де $R_t = R(U,V)$ – відстань між найближчими кластерами, поєднуваними на *t*-му кроці, μ_U, μ_V – центри цих кластерів. Якщо це відношення на кожному кроці більше за одиницю, то відстань *R* є розтягувальною; якщо воно завжди менше за одиницю, то стискаючою. Є й такі відстані, які не є ні стискаючими, ні розтягуючими, наприклад, $R^{с}$ і $R^{ц}$. Про них говорять, що вони *зберігають метрику простору*.

На практиці часто застосовують *гнучку відстань*, яка є компромісом між методами близького сусіда, далекого сусіда й середньої відстані. Вона визначається одним параметром β замість чотирьох:

$\alpha_U = \alpha_V = (1-\beta)/2, \gamma = 0, \beta < 1$.

Гнучка відстань є стискаючою при $\beta > 0$ і розтягувальною при $\beta < 0$. Стандартна рекомендація: $\beta = -0{,}25$.

Властивість редуктивності. Найбільш трудомісткою операцією в Алгоритмі 5 є пошук пари найближчих кластерів на кроці 3. Він вимагає

$O(m^2)$ операцій усередині основного циклу. Відповідно, побудова всього таксономічного дерева вимагає $O(m^3)$ операцій. Це обмежує застосовність алгоритму вибірками довжиною в кілька сотень об'єктів.

Ідея прискорення алгоритму полягає в тому, щоб перебирати лише найбільш близькі пари. Задається параметр δ, і перебір обмежується скороченою множиною пар $\{(U,V): R(U,V) \le \delta\}$. Коли всі такі пари будуть вичерпані, параметр δ збільшується, і формується нова скорочена множина пар. І так далі, до повного злиття всіх об'єктів в один кластер. Реалізацію представлено в Алгоритмі 6.

Алгоритм 6. Швидка агломеративна кластеризація на основі редуктивності

1: ініціалізувати множину кластерів C_1:
$t := 1; C_t = \{\{x_1\},\{x_2\},...,\{x_m\}\}$;
2: вибрати початкове значення параметра δ;
3: $P(\delta) := \{(U,V) \mid U,V \in C_t, R(U,V) \le \delta\}$;
4: **для всіх** $t = 2,...,m$ (t – номер ітерації):
5: **якщо** $P(\delta) = \varnothing$ **то**
6: збільшити δ так, щоб $P(\delta) \ne \varnothing$
7: знайти в $P(\delta)$ пару найближчих кластерів:
$(U,V) := \arg\min_{(U,V)\in P(\delta)} R(U,V)$
$R_t := R(U,V)$;
8: вилучити кластери U і V, додати злитий кластер $W = U \cup V$:
$C_t := C_{t-1} \cup \{W\} \setminus \{U,V\}$;
9: **для всіх** $S \in C_t$
10: обчислити відстань $R(W, S)$ за формулою Ланса-Вільямса;
11: **якщо** $R(W, S) \le \delta$ **то**
12: $P(\delta) := P(\delta) \cup \{(W,S)\}$.

Доведено, що цей алгоритм приводить до тієї ж кластеризації, що й Алгоритм 5, якщо відстань R має властивість редуктивності:

Визначення 1 (Брюінош, 1978). *Відстань R називають редуктивною, якщо для будь-якого $\delta > 0$ і будь-яких δ-близьких кластерів U і V об'єднання δ-околів U і V містить у собі δ-окіл кластера $W = U \cup V$:*

$$\{S \mid R(U \cup V, S) < \delta, R(U,V) \le \delta\} \subseteq \{S \mid R(S,U) < \delta \text{ або } R(S,V) < \delta\}.$$

Теорема 2 (Діде і Моро, 1984). Якщо виконуються наступні три умови, то відстань R є редуктивною:

1) $\alpha_U \ge 0, \alpha_V \ge 0$;

2) $\alpha_U + \alpha_V + \min\{\beta, 0\} \ge 1$;

3) $\min\{\alpha_U, \alpha_V\} + \gamma \geq 0$.

Порівняння умов теорем 2 і 1, показує, що будь-яка редуктивна відстань є монотонною, отже, дозволяє відобразити процес кластерізації у вигляді дендрограми. З перерахованих вище відстаней тільки R^{ij} не є редуктивною.

В алгоритмі 6 можливі різні евристичні стратегії вибору параметра δ на кроках 2 і 6. Загальні міркування такі: якщо δ настільки велике, що $P(\delta)$ містить практично всі пари кластерів, то ми фактично повертаємося до неефективного Алгоритму 5; якщо ж δ мале, то доводиться занадто часто формувати множину $P(\delta)$.

На практиці діють таким чином. Якщо число кластерів в C_t не перевищує поріг n_1, то як $P(\delta)$ беруть множину всіх можливих пар *(U, V)* з C_t. В іншому випадку вибирається випадковим чином n_2 відстаней *R(U, V)*, і δ покладається рівним найменшому з них. У разі редуктивності параметри алгоритму n_1 і n_2 впливають тільки на час виконання алгоритму, але не на результат кластеризації. Оптимальні значення для них підбираються за допомогою калібрувальних тестів і, взагалі кажучи, залежать від комп'ютера. Як початковий вибір можна запропонувати $n_1=n_2=20$.

13.1.6. Визначення числа кластерів

найпростіше робити шляхом відсікання правої ділянки дендрограми. На горизонтальній осі знаходиться інтервал максимальної довжини $|R_{t+1} - R_t|$, і як результуюча кластеризація видається множина кластерів C_t. Кількість кластерів дорівнює $K = m - t + 1$. За необхідності можна задати обмеження на мінімальну й максимальну кількість кластерів $K_0 \leq K \leq K_1$ і вибирати t, що задовольняють обмеженням $m - K_1 + 1 \leq t \leq m - K_0 + 1$.

У багатьох прикладних задачах інтерес представляє таксономічне дерево цілком, і визначати оптимальне число кластерів не має особливого сенсу.

13.1.7. Переваги й недоліки агломеративної кластеризації

Точної відповіді на запитання, який алгоритм кластеризації кращий, не існує. Кожна з відстаней, перерахованих вище, має свої недоліки й підходить не для всіх задач.

Метод близького сусіда має ланцюговий ефект, коли незалежно від форми кластера до нього приєднуються найближчі до границі об'єкти. У деяких випадках це приводить до того, що кластери «відрощують щупальця». Залежно від задачі ця властивість може бути як корисною, так і такою, що заважає. Метод близького сусіда добре підходить для виділення кластерів стрічкової форми.

Метод далекого сусіда ланцюгового ефекту не має, але на ранньому етапі може поєднувати досить несхожі групи.

Відстань між центрами мас не монотонна й не редуктивна, тому рідко використовується на практиці, хоча інтуїтивно видається «золотою серединою».

Метод Уорда виявився найкращим за результатами експериментального порівняння на представницькому наборі модельних задач. Він частіше за інші методи будує інтуїтивно кращу таксономію.

Запитання до розділу 13

1. Чи є можливість однозначної реалізації алгоритму кластеризації?

2. Які цілі створення та вирішення задач кластеризації?

3. Наведіть відомі вам типи кластерних структур.

4. Опишіть основні етапи алгоритму найкоротшого незамкнутого шляху, алгоритм FOREL.

5. Опишіть, як формують функціонал якості в алгоритмах кластеризації, кластеризація за допомогою ЕМ-алгоритму.

Розділ 14
Мережі Кохонена

14.1. Базові відомості про мережі Кохонена

Дотепер ми розглядали нейронні мережі, призначені для навчання з учителем, коли для кожного об'єкта x_i задана відповідна йому відповідь y_i. Мережі Кохонена вирішують задачі навчання без учителя, коли задаються тільки самі об'єкти x_i, і потрібно виділити відособлені «щільні згустки» об'єктів – кластери, і навчитися відносити нові об'єкти до цих кластерів. Кластеризація ґрунтується на припущенні, що в просторі X уведена метрика $\rho: X \times X \to \mathbb{R}$, що адекватно оцінює ступінь подібності будь-якої пари об'єктів.

14.1.1 Моделі конкурентного навчання

Нехай $X = \mathbb{R}^n$ – простір об'єктів, $Y = \{1,...,M\}$ – множина кластерів, число M фіксоване. Задана навчальна вибірка об'єктів $X_m = \{x_i\}_{i=1}^m$. Потрібно виробити правило віднесення об'єктів до кластерів $a: X \to Y$.

Правило жорсткої конкуренції WTA. Найбільш очевидний підхід полягає в тому, щоб увести вектори $w_s \in \mathbb{R}^n, s = 1,2,...,S$, що описують центри кластерів, і відносити довільний об'єкт $x \in X$ до найближчого кластера:

$$a(x) = \arg\min_{s \in Y} \rho(x, w_s) \qquad (14.1)$$

Образно говорячи, кластери змагаються за право приєднати до себе об'єкт x. Кластер, найближчий до *x*, називається кластером-переможцем, а вираз (14.1) –правилом WTA (winner takes all).

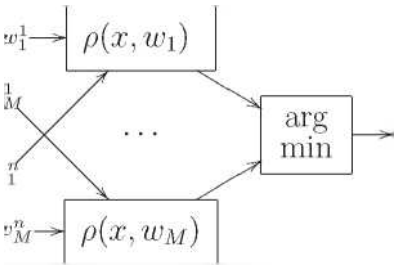


Рис. 14.1. Подання алгоритму кластеризації (14.1) у вигляді двошарової нейронної мережі.

Настроювання алгоритму кластеризації $a(x)$ зводиться до оптимізації розташування центрів w_s. Для цього мінімізується функціонал якості кластеризації, який дорівнює середньому квадрату відстані між об'єктами й центрами кластерів:

$$Q(w_1,...,w_S) = \frac{1}{2}\sum_{i=1}^{m} \rho^2\left(x_i, w_{a(x_i)}\right) \to \min_{\{w_s\}}$$

Припустимо, що метрика евклідова: $\rho(x,w) = \|x - w\|$. Тоді можливо виписати градієнт функціонала Q по вектору w_s:

$$\frac{\partial Q}{\partial w_s} = \sum_{i=1}^{m} (w_s - x_i)\left[a(x_i) = s\right].$$

Для пошуку центрів кластерів w_s можна застосувати метод стохастичного градієнта практично без модифікацій. Єдина відмінність полягає в тому, що тепер правило відновлення ваг на кроці 6 буде мати вигляд

$$w_s := w_s + \eta(x_i - w_s)\left[a(x_i) = s\right], \qquad (14.2)$$

де $x_i \in X_m$ – випадковим чином вибраний навчальний об'єкт, η – градієнтний крок, він же *темп навчання*. Зміст даного правила очевидний: якщо об'єкт відноситься до кластера s, то центр цього кластера w_s дещо зрушується в напрямку об'єкта x_i, інші центри не змінюються.

Формальна подібність формули (2) з персептронним правилом наводить на думку, що кластеризація здійснюється алгоритмом, аналогічним нейронній мережі. Вираз (14.1) і насправді представимо у вигляді двошарової суперпозиції, тільки замість звичних скалярних добутків $\langle x, w_s \rangle$ обчислюються функції відстані $\rho(x, w_s)$, а на виході замість підсумовування застосовується операція мінімуму, див. Рис. 14.1. При цьому центри кластерів w_s взаємно однозначно відповідають нейронам прихованого шара, які називаються *нейронами Кохонена*. Нейрон, вихід якого мінімальний, називається *нейроном-переможцем*.

Розглянутий різновид нейронних мереж називають *мережею з конкурентним навчанням*. Початково вона була запропонована як чисто математична модель. Пізніше нейрофізіологам удалося знайти деякі її аналоги в біологічних нейронних мережах.

Альтернативна назва – векторне квантування, що навчається (learning vector quantization, LVQ) – пов'язана з тим, що по вхідній вибірці з m об'єктів x_i будуються S нових об'єктів w_s, схожих на вхідні, – це центри гнізд, по яких розподіляються («квантуються») вхідні об'єкти. Як правило, $S \square m$, тому заміна об'єктів на найближчі до них центри дозволяє ефективно стискати дані при незначній втраті інформації. Обсяг інформації, що зберігається, регулюється єдиним параметром S, що досить зручно в додатках.

Правило справедливої конкуренції CWTA. Недолік конкурентного навчання за правилом WTA полягає в тому, що при випадковій ініціалізації ваг нейрон Кохонена може потрапити в таку область, де він ніколи не стане переможцем. У результаті з'явиться неінформативний порожній кластер.

Для подолання цього недоліку алгоритм дещо модифікується. Вводиться «механізм стомлення» переможців – *правило CWTA* (conscience WTA):

$$a(x) = \arg\min_{s \in Y} C_{s\rho(x, w_s)}, \qquad (14.3)$$

де C_s – кількість перемог *m*-го нейрона в ході навчання. Таким чином, кластери штрафуються за надто часте приєднання об'єктів.

Правило м'якої конкуренції WTM. Іншим недоліком правила WTA є повільна швидкість збіжності, пов'язана з тим, що на кожній ітерації модифікується тільки один нейрон-переможець w_s. Для прискорення збіжності, особливо на початкових ітераціях, можна підлаштовувати відразу кілька нейронів, близьких до об'єкта x_i. Для цього вводиться ядро невід'ємна монотонно спадна на $[0, +\infty]$ функція відстані $K(\rho)$. Іноді накладається додаткова вимога нормування $K(0) = 1$. Часто беруть гаусовське ядро

$K(\rho)=\exp(-\beta\rho^2)$ при деякому $\beta>0$. Замість правила жорсткої конкуренції WTA уводиться м'яка конкуренція – *правило WTM* (winner takes most):

$$w_s := w_s + \eta(x_i - w_s)K(\rho(x_i, w_s)), s = 1,2,...,S. \quad (14.4)$$

Тепер на кожній ітерації центри всіх кластерів зміщаються убік x_i, але чим далі центр перебуває від x_i, тем менша величина зсуву. Помітимо, що (14.2) є частковим випадком (14.4), якщо взяти $K(\rho(x_i, w_s)) = [a(x_i) = s]$.

На початкових ітераціях має сенс вибрати невелике значення коефіцієнта β, щоб усі вагові вектори встигли переміститься ближче до області вхідних векторів. Потім β можна збільшувати, роблячи конкуренцію усе більш жорсткою, і поступово переходячи до корекції тільки одного нейрона-переможця.

Завдяки здатності до згладжування, правило WTM має численні застосування. Один з найважливіших – карти Кохонена, що самоорганізуються.

14.1.2 Карти Кохонена, що самоорганізуються

Карти Кохонена, що самоорганізуються (self-organizing maps, SOM) застосовуються для візуалізації багатовимірних даних. Вони дають лише загальну картину, досить розмиту й піддану викривленням, оскільки спроєцирувати багатовимірну вибірку на площину без викривлень у загальному випадку неможливо. Проте, карти Кохонена дозволяють побачити ключові особливості кластерної структури вибірки. Вони використовуються на стадії *розвідницького аналізу даних*, скоріше для загального розуміння задачі, ніж для одержання яких-небудь точних результатів.

Ідея полягає в тому, щоб спроєцирувати усі об'єкти вибірки на плоску карту, точніше, на множину вузлів прямокутної сітки заздалегідь заданого розміру $S \times H$. На практиці S і H мають порядок десятків або сотень. Кожному вузлу сітки приписується нейрон Кохонена з вектором ваг $w_{sh} \in \mathbb{R}^n, s = 1,2,...,S; h = 1,2,...,H$. Таким чином, множина Y збігається з множиною вузлів сітки $Y = \{1,2,...,S\} \times \{1,2,...,H\}$.

Алгоритм кластеризації $a(x)$ видає пари індексів $(s,h) \in Y$, що показують, у який вузол сітки проектується об'єкт x. Щоб карта відбивала кластерну структуру вибірки, близькі об'єкти повинні потрапляти в близькі вузли сітки.

Навчання нейронів проводиться методом стохастического градієнта, див. Алгоритм 7. Після випадкового вибору об'єкта x_i на кроці 3 визначається нейрон-переможець згідно із правилом WTA. Відповідний йому вузол сітки позначається в алгоритмі через (s_i, h_i). Потім, згідно із правилом WTM, цей нейрон і нейрони, розташовані в найближчих вузлах сітки, зрушуються убік вектора x_i. Правило навчання, застосовуване на кроці 6, схоже з (4), тільки

замість метрики $\rho(x,x')$, визначеної на просторі об'єктів, використовується евклідова метрика на множині вузлів сітки Y:

$$r\big((s_i,h_i),(m,h)\big)=\sqrt{(s-s_i)^2+(h-h_i)^2}\ .$$

Алгоритм 7. Навчання карти Кохонена методом стохастичного градієнта

Вхід:

X_m – навчальна вибірка;

η – темп навчання;

Вихід:

Вектори синаптичних ваг $w_{sh}, s=1,2,...,S, h=1,2,...,H$;

1: ініціалізувати ваги:

$$w_{sh} := random\left(-\frac{1}{2SH},\frac{1}{2SH}\right);$$

2: **повторювати**

3: вибрати об'єкт x_i з X_m випадковим чином;

4: WTA: обчислити координати вузла, у який проектується об'єкт x_i:

$$(s_i,h_i) := a(x_i) \equiv \arg\min_{(s,h)\in Y} \rho(x_i,w_{sh})$$

5: **для всіх** $(s,h)\in Y$, достатньо близьких до (s_i,h_i)

6: WTM: зробити крок градієнтного спуска:

$$w_{sh} := w_{sh} + \eta(x_i - w_{sh})K\big(r\big((s_i,h_i),(s,h)\big)\big);$$

7: **поки** розміщення всіх об'єктів у вузлах сітки не стабілізується;

Як і раніше, $K(\rho)$ – ядро згладжування, наприклад, $K(\rho)=\exp(-\beta\rho^2)$. Параметр β задає ступінь згладженості карти: чим менше β, тем м'якше конкуренція нейронів, і тим більше згладженими будуть виглядати границі кластерів на карті. Має сенс збільшувати значення параметра β від ітерації до ітерації, щоб мережа Кохонена спочатку навчилася кластерній структурі «загалом», а потім зосередилася на деталях.

Алгоритм зупиняється, коли проекції всіх, або хоча б більшості, об'єктів вибірки $(s_i,h_i) := a(x_i)$ перестануть змінюватись від ітерації до ітерації.

Мистецтво інтерпретації карт Кохонена. Якщо через настроєну карту Кохонена (алгоритм $a(x)$) пропустити всі об'єкти навчальної вибірки X_m, то кластери вхідного простору відобразяться в згустки точок на карті. При цьому вектори ваг у вузлах-згустках повинні бути одночасно схожі на всі об'єкти відповідних кластерів.

Зазвичай для кожної точки карти обчислюють середню відстань до k найближчих точок вибірки, і отриманий розподіл середніх відстаней відображають у вигляді двоколірної півтонової карти. Чим менша середня відстань, тем вище локальна щільність точок на карті. На тій же карті

відображають і самі точки навчальної вибірки. На карті добре видно, скільки кластерів виділено, у яких місцях вони розташувалися, і які об'єкти вибірки в них потрапляють, Рис. 14.2.

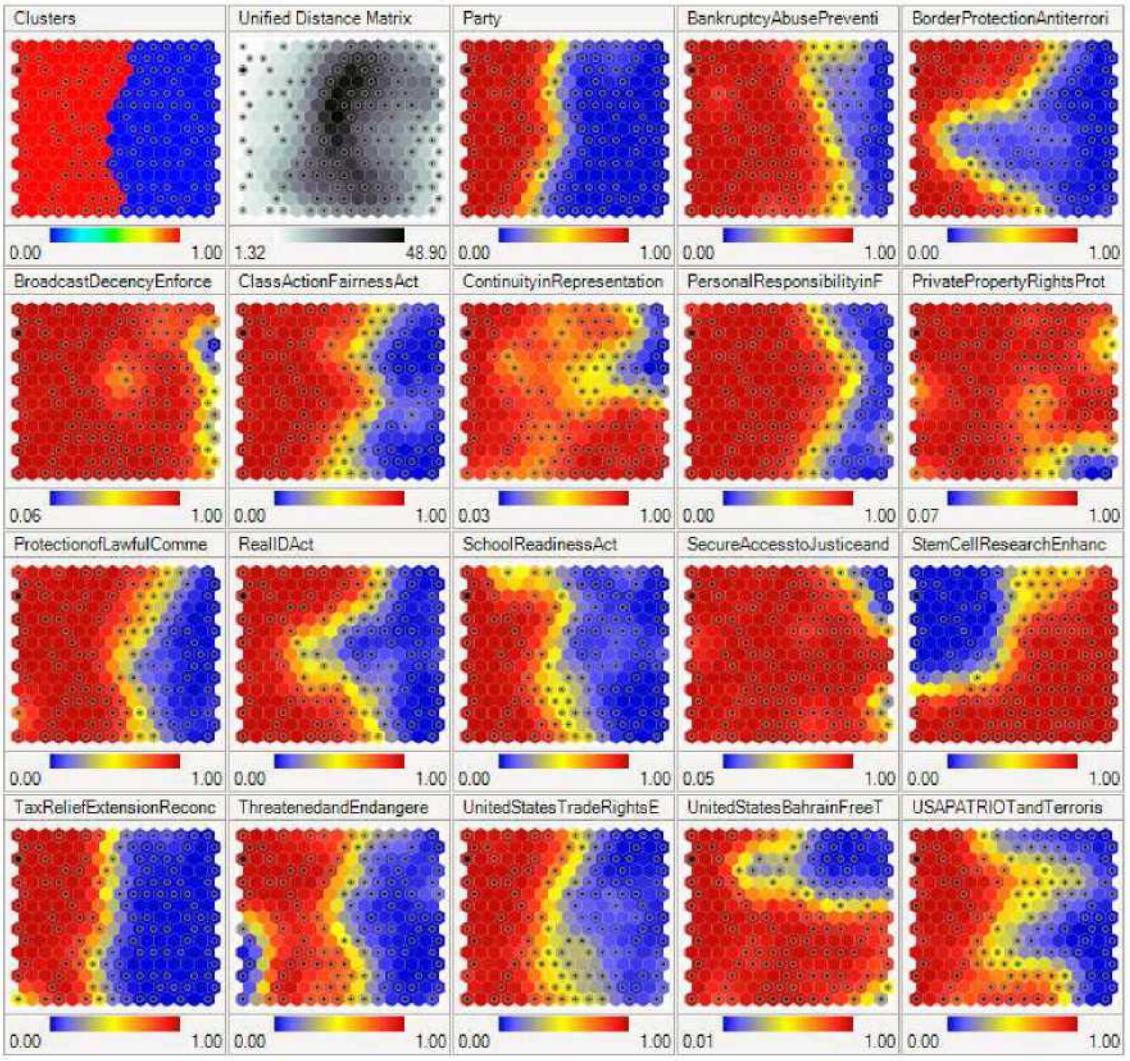


Рис. 14.2. Карти Кохонена для задачі UCI.house-votes (об'єкти – конгресмени, ознаки – результати голосування по 17 питаннях). Перші три графіка у верхньому ряді: (1) поділ вибірки на два кластери, (2) двоколірна півтонова карта, (3) кольорова карта за ознакою «партія» {демократ, республіканець}. Голосування по більшості питань добре узгоджуються з партійною приналежністю конгресменів.

Наступний крок – пошук інтерпретації кластерів. Для цього будуються ще n карт, по одній карті на кожну ознаку. Тепер колір вузла s,h відповідає значенню j-го компонента вектора ваг $w_{s,h}$. Зазвичай ці карти розфарбовують у геодезичні кольори від синього до коричневого. Сині ділянки карти відповідають найменшим значенням j-ї ознаки, червоні – найбільшим. Порівнюючи карти ознак із загальною картою кластерів, можна знайти кластери, яким властиві підвищені або знижені значення окремих ознак.

У результаті змістовна інтерпретація кластерів формулюється шляхом перерахування всіх ознак, що мають або підвищені, або знижені значення в кожному із кластерів.

Ще один спосіб інтерпретації – виділення на карті всіх вузлів, у які попадає обрана підмножина об'єктів. Якщо об'єкти згрупувалися в одному місці, отже, ми маємо справу із кластером або його частиною. Виділяючи різні підмножини об'єктів, що мають наперед відому змістовну інтерпретацію, можна знаходити інтерпретації різних зон на карті.

Недоліки карт Кохонена

- Суб'єктивність. Не завжди ясно, які особливості карти обумовлені кластерною структурою даних, а які – властивостями ядра, що згладжує. Від вибору параметра в суттєво залежить згладженість границь кластерів і ступінь деталізації карти.
- Наявність викривлень. Близькі об'єкти вхідного простору можуть переходити в далекі точки на карті. Зокрема кластери, що об'єктивно існують, можуть розриватись на фрагменти. І навпаки, далекі об'єкти можуть випадково виявитися на карті поряд, особливо, якщо вони були однаково далекі від усіх кластерів. Викривлення неминучі при проектуванні багатовимірної вибірки на площину. Розподіл крапок на карті дозволяє судити лише про локальну структуру багатовимірних даних, і то не завжди.
- Залежність від ініціалізації. Початковий розподіл ваг суттєво впливає на процес навчання й може позначатися не тільки на розташуванні кластерів, але навіть на їхній кількості. Іноді рекомендується побудувати кілька карт і вибрати з них найбільш «вдалу».

Не секрет, що популярність карт Кохонена обумовлена в значній мірі їх естетичною привабливістю й враженням наукоподібної загадковості, які вони справляють на недосвідченого користувача. На практиці вони застосовуються в основному як допоміжний засіб для попереднього візуального аналізу й виявлення неочевидних закономірностей у багатовимірних даних.

14.1.3. Гібридні мережі зустрічного поширення

Ще одне важливе застосування нейронів Кохонена з їхньою здатністю кластеризувати вхідні вектори – кусочна апроксимація функцій.

Розглянемо задачу відновлення регресії, коли на елементах навчальної вибірки $X_m = \{x_i\}_{i=1}^m$ задані дійсні відповіді $y_i = y*(x_i)$. Ідея полягає в тому, щоб спочатку розбити навчальні об'єкти (і весь простір X) на кластери, не користуючись відповідями y_i, потім для кожного кластера побудувати свою локальну апроксимацію цільової залежності $y*$. При використанні жорсткої конкуренції WTA ці апроксимації утворюють кусочно-неперервну функцію. М'яка конкуренція WTM «склеює» з локальних шматків гладку апроксимацію.

Кусочно-постійна апроксимація. Щоб алгоритм кластеризації перетворити в алгоритм кусочної апроксимації, до нього додається ще один нейрон з лінійною функцією активації, що утворює третій шар, з вагами $v_1, v_2, ..., v_S$:

$$a(x) = v_{s*(x)} = \sum_{s=1}^{S} v_s \left[s*(x) = s \right] \qquad (14.5)$$

$$s^*(x) = \arg\min_{s=1,\dots,S} \rho(x, w_s)$$

де $s^*(x)$ – номер нейрона-переможця на об'єкті x, що обчислюється за правилом WTA.

При квадратичній функції втрат задача настроювання ваг w_s легко вирішується аналітично:

$$Q(v) = \frac{1}{2}\sum_{i=1}^{m}\left(a(x_i) - y_i\right)^2 \to \min_v$$

$$\frac{\partial Q}{\partial w_s} = \sum_{i=1}^{m}\left(a(x_i) - y_i\right)\left[s^*(x_i) = s\right] = 0$$

Підставляючи сюди вирази для $a(x_i)$ з (5), одержуємо

$$v_s = \frac{\sum_{i=1}^{m} y_i\left[s^*(x_i) = s\right]}{\sum_{i=1}^{m}\left[s^*(x_i) = s\right]}$$

Простіше говорячи, оптимальне значення вагового коефіцієнта v_s є середнє y_i по всіх об'єктах x_i, що потрапили в s-й кластер. Відповідь алгоритму $a(x)$ для всіх об'єктів, що потрапляють в s-й кластер, буде однакова і дорівнюватиме v_s. Це означає, що $a(x)$ реалізує кусочно-постійну функцію.

Гла′дка апроксимація будується за допомогою правила м'якої конкуренції WTM при вибраному гла′дкому ядрі $K(\rho)$. Алгоритм $a(x)$ є формулою Надарая-Ватсона для згладжування відповідей v_s по S точках – центрах кластерів:

У цьому випадку задача мінімізації $Q(v)$ зводиться до розв'язку системи лінійних рівнянь розміру $S \times S$. Замість явного розв'язку системи можна знову скористатися методом стохастичного градієнта. На кожній ітерації обновляються й ваги шару Кохонена w_s, і ваги третього (вихідного) шару v_s:

$$\begin{cases} w_s = w_s - \eta(w_s - x_i)K\left(\rho(x_i, w_s)\right); \\ v_s = v_s - \eta\left(a(x_i) - y_i\right)\dfrac{K\left(\rho(x_i, w_s)\right)}{\sum_{l=1}^{S} K\left(\rho(x_i, w_l)\right)}; \end{cases}$$

Помітимо, що відновлення ваг w_s впливає на ваги v_s, а зворотного впливу немає. Даний метод одержав назву *зустрічного поширення помилки*, оскільки другий шар налаштовується за правилом Кохонена, а третій шар – так само, як у методі back-propagation.

14.2. Багатовимірне шкалування

Візуалізація кластерної структури заданої вибірки об'єктів X_m – непроста проблема, особливо якщо вибірка містить тисячі об'єктів, а простір об'єктів суттєво багатовимірний.

Задача *багатовимірного шкалування* (multidimensional scaling, MDS) полягає в наступному. Є навчальна вибірка об'єктів $X_m = \{x_1, x_2, ..., x_m\} \subset X$. Задані відстані $R_{ij} = \rho(x_i, x_j)$ для деяких пар навчальних об'єктів $(i, j) \in D$. Потрібно для кожного об'єкта $x_i \in X_m$ побудувати його ознаковий опис – вектор $x_i = (x_i^1, x_i^2, ..., x_i^n)$ в евклідовому просторі $\mathbb{R}^n$ так, щоб евклідові відстані d_{ij} між об'єктами x_i і x_j

$$d_{ij}^2 = \sum_{d=1}^{n} \left(x_i^d - x_j^d\right)^2$$

якомога точніше наближали вхідні відстані R_{ij} для всіх $(i, j) \in D$. Дану вимогу можна формалізувати по-різному; один з найпоширеніших способів – через мінімізацію функціонала, називаного *стресом*:

$$S(X_m) = \sum_{(i,j)\in D} w_{ij} \left(d_{ij} - R_{ij}\right)^2 \to \min,$$

де мінімум береться за сукупністю mn змінних $\left(x_i^d\right)_{i=1,m}^{d=1,n}$.

Розмірність n зазвичай задається невелика. Зокрема, при n=2 багатовимірне шкалування дозволяє відобразити вибірку у вигляді множини точок на площині (scatter plot). Плоский вигляд, як правило, викривлений (S>0), але в цілому відображає основні структурні особливості багатовимірної вибірки, зокрема, її кластерну структуру. Тому двовимірне шкалування часто використовують як інструмент попереднього аналізу й розуміння даних.

На практиці відстані частіше бувають відомі для всіх пар об'єктів, але ми будемо розглядати більш загальний випадок, коли D – довільно задана множина пар індексів об'єктів.

Ваги w_{ij} задають, виходячи із цілей шкалування. Зазвичай беруть $w_{ij} = \left(R_{ij}\right)^{\gamma}$. При $\gamma < 0$ пріоритет віддається більш точному наближенню малих відстаней; при $\gamma > 0$ – великих відстаней. Найбільш адекватним вважається значення $\gamma = -2$, коли функціонал стресу набуває фізичного сенсу потенційної енергії системи m матеріальних крапок, зв'язаних пружними зв'язками; потрібно знайти рівноважний стан системи, у якому потенційна енергія мінімальна.

Функціонал стресу $S(X_m)$ складним чином залежить від mn змінних, має величезну кількість локальних мінімумів, і його обчислення достатньо трудомістке. Тому багато алгоритмів багатовимірного шкалування засновані на ітераційному розміщенні об'єктів по одному. На кожній ітерації оптимізуються евклідові координати тільки одного з об'єктів при фіксованих інших об'єктах.

Розміщення одного об'єкта методом Ньютона-Рафсона. Розглянемо адитивну складову функціонала стресу, що залежить тільки від одного об'єкта $x \in X_m$ з координатами $x \equiv \left(x^1, x^2, ..., x^n\right)$, які й потрібно знайти:

$$S(x) = \sum_{x_i \in U} w_i \left(d_i(x) - R_i\right)^2 \to \min_x,$$

де $U \subseteq X_m$ – всі об'єкти з відомими відстанями $R_i = \rho(x, x_i)$; $d_i(x)$ – евклідова відстань між векторами $x \equiv \left(x^1, x^2, ..., x^n\right)$ і $x_i \equiv \left(x_i^1, x_i^2, ..., x_i^n\right)$.

Застосуємо метод Ньютона-Рафсона для мінімізації $S(x)$.

Оскільки функціонал багатоекстремальний, дуже важливо вибрати вдале початкове наближення $x^{(0)}$. Вектор x повинен бути схожий на ті вектори $x_i \in U$, для яких відстані R_i малі. Непогана евристика – вибрати як $x^{(0)}$ зважене середнє всіх векторів з U, наприклад, з вагами $c_i = R_i^{-2}$:

$x^{(0)} = \dfrac{\sum\limits_{x_i \in U} c_i x_i}{\sum\limits_{x_i \in U} c_i}$. Потім запускається ітераційний процес.

$$x^{(t+1)} = x^{(t)} - h_t \left(S''\left(x^{(t)}\right)\right)^{-1} S'\left(x^{(t)}\right),$$

де $S'(x^{(t)})$ – вектор перших похідних (градієнт) функціонала $S(x)$ у точці $x^{(t)}$, $S''(x^{(t)})$ – матриця других похідних (гессіан) функціонала $S(x)$ у точці $x^{(t)}$, h_t – величина кроку, який можна покласти рівним 1, але більш ретельний його добір здатний збільшити швидкість збіжності.

Знайдемо вирази для градієнта й гессіана.

$$\frac{\partial d_i}{\partial x^a} = \frac{x^a - x_i^a}{d_i};$$

$$\frac{\partial S}{\partial x^a} = 2\sum_{x_i \in U} w_i \left(1 - \frac{R_i}{d_i}\right)(x^a - x_i^a);$$

$$\frac{\partial^2 S}{\partial x^a \partial x^a} = 2\sum_{x_i \in U} w_i \left(\frac{R_i}{d_i}\left(\frac{x^a - x_i^a}{d_i}\right)^2 - \frac{R_i}{d_i} + 1\right);$$

$$\frac{\partial^2 S}{\partial x^a \partial x^b} = 2\sum_{x_i \in U} w_i \frac{R_i}{d_i}\left(\frac{x^a - x_i^a}{d_i}\right)\left(\frac{x^b - x_i^b}{d_i}\right)$$

Помітимо, що в просторах малої розмірності $n \le 3$ обернення гессіана – достатньо проста операція, однак з ростом розмірності обчислювальні витрати ростуть як $O(n^3)$.

Ітерації Ньютона-Рафсона припиняються, коли значення стресу $S(x)$ стабілізується або вектор x перестає суттєво змінюватися, тобто стабілізується норма різниці $\| x^{(t+1)} - x^{(t)} \|$.

Будемо вважати, що розміщення об'єкта x відносно множини вже розміщених об'єктів $U \subseteq X_m$ реалізується процедурою НьютонРафсон(x, U).

14.3. Субквадратичний алгоритм багатовимірного шкалування

Алгоритм 8 починає з того, що знаходить дві найвіддаленіші точки вибірки x_i, x_j. Достатньо розв'язати цю задачу приблизно. Зокрема, можна вибрати довільну точку, знайти найвіддаленішу від неї, потім для цієї точки знайти найвіддаленішу, і так далі. Зазвичай 3-4 ітерацій вистачає, щоб знайти пару досить далеких точок. Знайденим точкам приписують (у двовимірному випадку) евклідові координати (0, 0) і (0, R_{ij}). Потім знаходять третю точку x_k, найбільш відділену від перших двох, тобто для якої значення $\min\left\{R_{ki}, R_{kj}\right\}$ максимально. Евклідові координати x_k визначаються (у двовимірному випадку) виходячи з того, що трикутник $\square ijk$ жорстко заданий довжинами своїх сторін.

Потім починається почергове додавання точок і їх розміщення відносно вже наявних. Після розміщення перших K точок їх положення уточнюється відносно одна одної. Ці точки, розміщені з особливою старанністю, стають «кістяком», відносно якого розміщаються всі інші точки. Відстані між «некістяковими» точками в алгоритмі взагалі не задіюються.

Алгоритм 8. Субквадратичний алгоритм багатовимірного шкалування

Вхід:

R_{ij} – матриця попарних відстаней між об'єктами, можливо, розріджена;

K – розмір кістяка;

Вихід:

Евклідові координати всіх об'єктів вибірки $x_i \equiv \left(x_i^1, x_i^2, ..., x_i^n\right), i = 1, 2, ..., m$

1: Ініціалізувати кістяк:

$U :=$ три достатньо далеких одна від одної точок;

2: **поки** $|U| < K$ – нарощувати кістяк

3: $x := \arg\max\limits_{x_i \in X_m \setminus U} \left(\min\limits_{x_j \in U} R_{ij}\right)$

4: НьютонРафсон$\left(x, U\right)$

5: $U := U \cup \left\{x\right\}$;

6: **поки** координати точок кістяка не стабілізуються:

7: знайти найбільш напружену точку в кістяку:

$x := \arg\max\limits_{x_i \in U} S\left(x\right)$

8: НьютонРафсон$\left(x, U \setminus \left\{x\right\}\right)$;

9: **для** $x \in X_m \setminus U$

10: НьютонРафсон$\left(x, U\right)$.

Якщо $K = m$, то всі точки будуть «кістяковими»; у цьому випадку розміщення є найбільш точним, але й найбільш довгим, вимагаючи $O\left(m^2\right)$ операцій. У загальному випадку число операцій алгоритму $O\left(K^2\right) + O\left(Km\right)$.

Вибираючи розмір «кістяка» K, можна знаходити компроміс між точністю й часом побудови розв'язку.

Карта подібності відображає результат багатовимірного шкалування при $n = 2$ у вигляді плоского точкового графіка. Евклідова метрика й функціонал стресу інваріантні відносно довільних ортогональних перетворень карти подібності – зрушень, поворотів і дзеркальних відбиттів. Тому осі на карті не мають інтерпретації. Для розуміння карти на ній позначають *орієнтири* – ті об'єкти вибірки, інтерпретації яких добре відомі. Якщо викривлення відстаней на карті не великі, то об'єкти, близькі до орієнтирів, повинні мати схожі інтерпретації.

Карта подібності дає лише загальне уявлення про взаємне розташування об'єктів вибірки. Робити на її основі які-небудь кількісні висновки, як правило, не можна.

Діаграма Шепарда дозволяє сказати, наскільки сильно перекручені відстані на карті подібності. Це точковий графік; по горизонтальній осі відкладаються вхідні відстані R_{ij}; по вертикальній осі відкладаються евклідові відстані d_{ij}; кожна точка на графіку відповідає деякій парі об'єктів $(i, j) \in D$. Якщо число пар перевищує кілька тисяч, відображається випадкова підмножина пар. Іноді на діаграмі зображується згладжена залежність $d_{ij}(R_j)$, а також згладжені границі верхніх і нижніх довірчих інтервалів, у яких $d_{ij}\ (R)$ перебуває з високою ймовірністю (наприклад, 90%) при кожному значенні R.

Ідеальною діаграмою Шепарда є похила пряма – бісектриса першої чверті. Чим «товстіше» хмара точок, представлена на діаграмі, тем сильніше викривлення, і тем меншої довіри заслуговує карта подібності.

Приклад 1. Один з надійних способів тестування методів багатомірного шкалування – розміщення початково двовимірних (або близьких до двовимірних) вибірок. Наприклад, можна вибрати множину міст України, і задати R_{ij} як евклідові відстані між їхніми географічними координатами (довгота, широта). Гарний алгоритм шкалування повинен побудувати карту подібності, схожу на географічну карту (з точністю до поворотів і відбиттів), а діаграма Шепарда повинна виявитися практично діагональною.

Запитання до розділу 14

1. В чому полягає суть моделі конкурентного навчання?
2. Якою є сфера застосування карт Кохонена, що самоорганізуються?
3. Опишіть алгоритм навчання карт Кохонена методом стохастичного градієнта.
4. Які недоліки карт Кохонена?
5. Поясніть принцип роботи алгоритму багатовимірного шкалування.

Розділ 15
Методи відновлення регресії (оцінки регресії)

Задачу навчання по прецедентах при $Y = R$ прийнято називати задачею відновлення регресії. Основні позначення залишаються колишніми. Заданий простір об'єктів X і множина можливих відповідей Y. Існує невідома цільова залежність $y^* : X \to Y$, значення якої відомі тільки на об'єктах навчальної вибірки $X_m = \left(x_i, y_i\right)_{i=1}^m$, $y_i = y^*\left(x_i\right)$. Потрібно побудувати алгоритм, який у даній задачі прийнято називати «функцією регресії» $a : X \to Y$, яка апроксимує цільову залежність y* .

15.1. Метод найменших квадратів

Нехай задана *модель регресії* – параметричне сімейство функцій $g\left(x, \alpha\right)$, де $\alpha \in R^p$ – вектор параметрів моделі. Визначимо функціонал якості апроксимації цільової залежності на вибірці X_m як суму квадратів помилок:

$$Q\left(\alpha, X_m\right) = \sum_{i=1}^{m}\left(g\left(x_i, \alpha\right) - y_i\right)^2 \tag{15.1}$$

Навчання по методу найменших квадратів (МНК) полягає в тому, щоб знайти вектор параметрів α^*, при якому досягається мінімум середнього квадрата помилки на заданій навчальній вибірці X_m:

$$\alpha^* = \arg\min_{\alpha \in R^p} Q\left(\alpha, X_m\right) \tag{15.2}$$

Стандартний спосіб розв'язку цієї оптимизаційної задачі – скористатися необхідною умовою мінімуму. Якщо функція $g\left(x, \alpha\right)$ достатнє число раз диференційована по α, то в точці мінімуму виконується система p рівнянь відносно p невідомих:

$$\frac{\partial Q}{\partial \alpha}\left(\alpha, X_m\right) = 2\sum_{i=1}^{m}\left(g\left(x_i, \alpha\right) - y_i\right)\frac{\partial g}{\partial \alpha}\left(x_i, \alpha\right) = 0 \tag{15.3}$$

15.2. Непараметрична регресія: ядерне згладжування

Непараметричне відновлення регресії засноване на тій же ідеї, що й непараметричне відновлення щільності розподілу.

Значення $a\left(x\right)$ обчислюється для кожного об'єкта x по декількох найближчих до нього об'єктах навчальної вибірки. Щоб можна було говорити про «близькість» об'єктів, на множині X повинна бути задана функція відстані $\rho\left(x, x'\right)$.

15.2.1. Формула Надарая-Ватсона

Візьмемо найпростішу модель регресії, яка тільки можлива – константу $g\left(x,\alpha\right)=\alpha,\ \alpha\in R$. Але при цьому, щоб не одержати тривіального розв'язку, задамо ваги об'єктів $w_i\left(x\right)$, що залежать від того об'єкта x, у якому ми збираємося обчислювати значення $a\left(x\right)=g\left(x,\alpha\right)$. Можна сказати і так, що навчання регресійної моделі буде проводитися окремо в кожній точці x простору об'єктів X.

Щоб обчислити значення $a\left(x\right)=\alpha$ для довільного $x\in X$, скористаємося методом найменших квадратів:

$$Q\left(\alpha;X_m\right)=\sum_{i=1}^{m}w_i\left(x\right)\left(\alpha-y_i\right)^2\to\min_{\alpha\in R}$$

Задамо ваги w_i навчальних об'єктів так, щоб вони убували в міру збільшення відстані $\rho\left(x,x_i\right)$. Для цього введемо незростаючу, гладку, обмежену функцію $K:\left[0,\infty\right)\to\left[0,\infty\right)$, яку назвемо ядром:

$$w_i\left(x\right)=K\left(\frac{\rho\left(x,x_i\right)}{h}\right)$$

Параметр h називають *шириною ядра* або *шириною вікна згладжування*. Чим менше h, тим швидше будуть убувати ваги $w_i\left(x\right)$ по мірі віддалення x_i від x.

Дорівнявши нулю похідну $\frac{\partial Q}{\partial\alpha}=0$, одержимо формулу ядерного згладжування Надарая-Ватсона:

$$a_h\left(x;X_m\right)=\frac{\sum_{i=1}^{m}y_iw_i\left(x\right)}{\sum_{i=1}^{m}w_i\left(x\right)}=\frac{\sum_{i=1}^{m}y_iK\left(\frac{\rho\left(x,x_i\right)}{h}\right)}{\sum_{i=1}^{m}K\left(\frac{\rho\left(x,x_i\right)}{h}\right)}\qquad(15.4)$$

Ця формула інтуїтивно очевидна: значення $a\left(x\right)$ є середнє y_i по об'єктах x_i, найближчих до x.

В одновимірному випадку $X=R$ метрику задають як $\rho\left(x,x_i\right)=\left|x-x_i\right|$.

15.2.2. Вибір ядра й ширини вікна

Ядерне згладжування – це доволі простий метод з погляду реалізації. Навчання алгоритму $a_h\left(x;X_m\right)$ зводиться до запам'ятовування вибірки, добору ядра K і ширини вікна h .

Вибір ядра K мало впливає на точність апроксимації, але визначальним чином впливає на ступінь гладкості функції $a_h\left(x\right)$. В одномірному випадку функція $a_h\left(x\right)$ стільки ж раз диференційована, скільки і ядро $K\left(r\right)$. Часто використовувані ядра показані на Рис. 15.1.

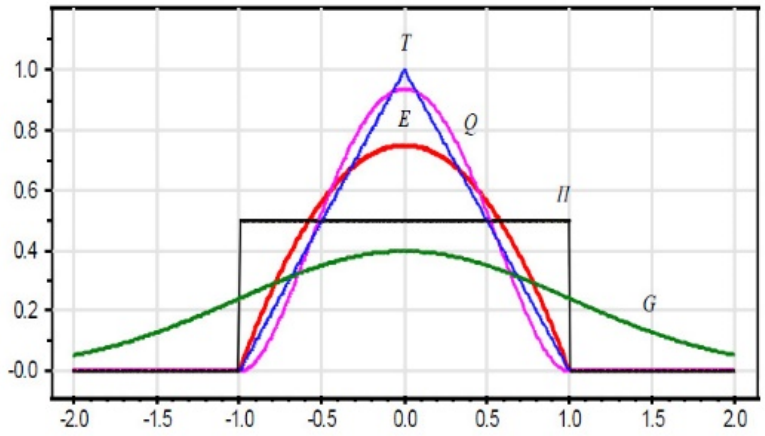


Рис. 15.1. Ядра, які часто застосовуються:
E-експоненційне; Q-квадратичне; T-трикутне; G-гаусівське; П-прямокутне

Для ядерного згладжування найчастіше беруть гаусівське ядро

$$K_G\left(r\right)=\exp\left(-\frac{1}{2}r^2\right)$$

або квартичне

$$K_Q\left(r\right)=\left(1-r^2\right)^2\left[\left|r\right|<1\right]$$

Якщо ядро $K\left(r\right)$ є фінітним, тобто $K\left(r\right)=0$ при $r\geq 1$, то ненульові ваги одержать тільки ті об'єкти x_i, для яких $\rho\left(x,x_i\right)<h$. Тоді у формулі (15.4) достетньо додавати тільки по найближчих сусідах об'єкта x. В одномірному випадку $X=R$ для ефективної реалізації цієї ідеї вибірка повинна бути впорядкована по зростанню x_i. У загальному випадку необхідна спеціальна структура даних, що дозволяє швидко знаходити множину найближчих сусідів для будь-якого об'єкта x.

Вибір ширини вікна h вирішальним чином впливає на якість відновлення залежності. При занадто вузькому вікні $\left(h\to 0\right)$ функція $a_h\left(x\right)$ прагне пройти через усі точки вибірки, але реагує на шум і тому має різкі коливання. При

занадто широкому вікні функція надмірно згладжується й у граничному випадку $\left(h \to \infty\right)$ вироджується в константу. Таким чином, повинно існувати оптимальне значення ширини вікна h^* – компроміс між точністю опису вибірки й гладкістю апроксимуючої функції.

Проблема локальних згущень виникає, коли об'єкти вибірки розподілені нерівномірно в просторі X. В областях локальних згущень оптимальною є менша ширина вікна, ніж в областях розрідженості. У таких випадках використовується *вікно змінної ширини* $h\left(x\right)$, що залежить від об'єкта x. Відповідно, ваги обчислюються по формулі $w_i\left(x\right) = K\left(\frac{\rho\left(x, x_i\right)}{h\left(x\right)}\right)$

Найпростіший спосіб – взяти в якості ширини вікна $h\left(x\right)$ відстань від об'єкта x до його $\left(k+1\right)$-го сусіда: $h_k\left(x\right) = \rho\left(x, x_x^{\left(k+1\right)}\right)$. Недолік цього способу в тому, що функція $h_k\left(x\right)$ є неперервною, але не гладкою, тому у функцій $w_i\left(x\right)$ та $a_{h_k}\left(x\right)$ будуть розривні перші похідні, навіть якщо ядро гладке. Для усунення цього недоліку можна згладити саму функцію $h_k\left(x\right)$ по вузлах рівномірної сітки, при постійній ширині вікна та якому-небудь гладкому ядрі, скажемо, K_Q.

Оптимізація ширини вікна. Щоб оцінити за даним h або k точність локальної апроксимації в точці x_i, саму цю точку необхідно виключити з навчальної вибірки. Якщо цього не робити, мінімум помилки буде досягатися при $h \to 0$. Такий спосіб оцінювання називається ковзним контролем з виключенням об'єктів по одному (leave-one-out, LOO):

$$LOO\left(h, X_m\right) = \sum_{i=1}^{m}\left(a_h\left(x_i; X_m \setminus \left\{x_i\right\}\right) - y_i\right)^2 \to \min_h$$

де мінімізація здійснюється по ширині вікна h або по числу сусідів k.

15.2.3. Проблема викидів: робастна непараметрична регресія

Оцінка Надарая-Ватсона вкрай чутлива до більших одиночних викидів. Ідея виявлення викидів полягає в тому, що чим більше величина помилки

$$\varepsilon_i = \left|a_h\left(x_i; X_m \setminus \left\{x_i\right\}\right) - y_i\right|$$

тим більшою мірою прецедент $\left(x_i, y_i\right)$ є викидом, і тем меншою повинна бути його вага. Ці міркування приводять до ідеї домножити ваги $w_i\left(x\right)$ на

коефіцієнти $\gamma_i = \tilde{K}\left(\varepsilon_i\right)$, де $\tilde{K}$ – ще одне ядро, загалом кажучи, відмінне від $K\left(r\right)$.

Коефіцієнти γ_i, як і помилки ε_i, залежать від функції a_h, яка, у свою чергу, залежить від γ_i. Зрозуміло, це не «зачароване коло», а гарний привід для організації ітераційного процесу. На цьому принципі побудовано Алгоритм 1.

Алгоритм 1. LOWESS – локально зважене згладжування.
Вхідні дані:
X_m – навчальна вибірка;
Вихідні дані:
Коефіцієнти γ_i, $i = 1,2,\ldots,m$;

1: ініціалізація: γ_i, $i = 1,2,\ldots,m$;

2: **повторювати**
3: обчислити оцінки ковзного контролю на кожному об'єкті:

$$a_i := a_h\left(x_i; X_m \setminus \left\{x_i\right\}\right) = \frac{\sum\limits_{j=1, j\neq i}^{m} y_j \gamma_j K\left(\frac{\rho\left(x_i, x_j\right)}{h\left(x_i\right)}\right)}{\sum\limits_{j=1, j\neq i}^{m} \gamma_j K\left(\frac{\rho\left(x_i, x_j\right)}{h\left(x_i\right)}\right)}, \; i = 1,2,\ldots,m$$

4: обчислити коефіцієнти γ_i : $\gamma_i = \tilde{K}\left(\left|a_i - y_i\right|\right); \; i = 1,2,\ldots,m$

5: **поки** коефіцієнти γ_i не стабілізуються.

На кожній ітерації будується функція a_h, потім уточнюються вагові множники γ_i. Як правило, цей процес сходиться досить швидко. Він називається локально зваженим згладжуванням (locally weighted scatter plot smoothing, LOWESS).

Методи відновлення регресії, що стійкі до шуму у вихідних даних, називають робастними, що означає «розумний, здоровий» (robust).

Можливі різні варіанти завдання ядра $\tilde{K}\left(\varepsilon\right)$.

Жорстка фільтрація: будується варіаційний ряд помилок $\varepsilon^{(1)} \leq \ldots \varepsilon^{(m)}$, і відкидається деяка кількість t об'єктів з найбільшою помилкою. Це відповідає ядру $\tilde{K}\left(\varepsilon\right) = \left[\varepsilon \leq \varepsilon^{(m-t)}\right]$.

М'яка фільтрація: використовується квартичне ядро $\tilde{K}\left(\varepsilon\right) = K_Q\left(\frac{\varepsilon}{6\ \mathsf{med}\left\{\varepsilon_i\right\}}\right)$, де $\mathsf{med}\left\{\varepsilon_i\right\}$ – медіана варіаційного ряду помилок.

15.2.4. Проблема крайових ефектів

В одновимірному випадку $X = R$ часто спостерігається значний зсув апроксимуючої функції $a_h\left(x\right)$ від дійсної залежності $y^*\left(x\right)$ поблизу мінімальних і максимальних значень x_i. Зсув виникає, коли об'єкти вибірки x_i розташовуються тільки по одну сторону (а не навколо) об'єкта x . Чим більше розмірність простору об'єктів, тем частіше виникає така ситуація.

Для розв'язку цієї проблеми залежність апроксимується в околі точки $x \in X$ не константою $a\left(u\right) = a$, а лінійною функцією $a\left(u\right) = \alpha\left(u - x\right) + \beta$.

Введемо для стислості скорочені позначення $w_i = w_i\left(x\right), d_i = x_i = x$ і запишемо задачу найменших квадратів:

$$Q\left(\alpha, \beta; X_m\right) = \sum_{i=1}^{m} w_i\left(\alpha d_i + \beta - y_i\right)^2 \to \min_{\alpha,\beta \in R}$$

Прирівнюючи нулю похідні $\frac{\partial Q}{\partial \alpha} = 0$ і $\frac{\partial Q}{\partial \beta} = 0$, одержимо систему лінійних рівнянь 2 х 2, розв'язок якої дає аналог формули Надарая-Ватсона:

$$a_h\left(x; X_m\right) - \frac{\sum_{i=1}^{m} w_i d_i^2 \sum_{i=1}^{m} w_i y_i - \sum_{i=1}^{m} w_i d_i \sum_{i=1}^{m} w_i d_i y_i}{\sum_{i=1}^{m} w_i \sum_{i=1}^{m} w_i d_i^2 - \left(\sum_{i=1}^{m} w_i d_i\right)^2}$$

У багатомірному випадку $X = R^n$ для обчислення коефіцієнтів у лінійній формі $a\left(u\right) = \alpha^T\left(u - x\right) + \chi$ доводиться вирішувати задачу багатомірної лінійної регресії. Причому вона повинна вирішуватися заново для кожної товчки $x \in X$, що пов'язане з великим обсягом обчислень.

15.3. Лінійна регресія

Нехай кожному об'єкту відповідає його ознаковий опис $\left(f_1\left(x\right), \ldots, f_n\left(x\right)\right)$, де $f_j : X \to R$ – числові ознаки, $j = 1, 2, \ldots, n$. Лінійною моделлю регресії називають лінійну комбінацію ознак з коефіцієнтами $\alpha \in R^n$:

$$g\left(x, \alpha\right) = \sum_{j=1}^{n} \alpha_j f_j\left(x\right)$$

Введемо матричні позначення:

$F = \left(f_j\left(x_i\right)\right)_{m\times n}$ – матриця об’єкти-ознаки;

$y = \left(y_i\right)_{m\times 1}$ -цільовий вектор;

$\alpha = \left(\alpha_j\right)_{n\times 1}$ -вектор параметрів.

У матричних позначеннях функціонал Q матиме вигляд:

$$Q\left(\alpha\right) = \left\|F\alpha - y\right\|^2$$

Запишемо необхідну умову мінімуму виразу (3) у матричному вигляді:

$$\frac{\partial Q}{\partial \alpha}\left(\alpha\right) = 2F^T\left(F\alpha - y\right) = 0,$$

звідки випливає $F^TF\alpha = F^Ty$. Ця система лінійних рівнянь відносно α називається нормальною системою для задачі найменших квадратів. Якщо матриця F^TF розміру $n\times n$ невироджена, то розв’язком нормальної системи є вектор

$$\alpha^* = \left(F^TF\right)^{-1}F^Ty = F^+y$$

Матрицю $F^+ = \left(F^TF\right)^{-1}F^T$ називають псевдооберненою для прямокутної матриці F. Підставляючи знайдений розв’язок у вихідний функціонал, одержуємо

$$Q\left(\alpha^*\right) = \left\|P_Fy - y\right\|^2$$

де $P_F = FF^+ = F\left(F^TF\right)^{-1}F^T$ – проекційна матриця.

Розв’язок має просту геометричну інтерпретацію. Добуток P_Fy є проекція цільового вектора h на лінійну оболонку стовпців матриці F. Різниця $\left(P_Fy - y\right)$ є проекція цільового вектора y на ортогональне доповнення цієї лінійної оболонки. Значення функціонала $Q\left(\alpha^*\right) = \left\|P_Fy - y\right\|^2$ є квадрат довжини перпендикуляра, опущеного з y на лінійну оболонку. Таким чином, МНК знаходить найкоротшу відстань від у до лінійної оболонки стовпців F .

Відома велика кількість чисельних методів розв’язування нормальної системи. Найбільшою популярністю користуються методи, засновані на ортогональних розкладаннях матриці F. Ці методи ефективні, мають гарну чисельну стійкість і дозволяють будувати різні модифікації й узагальнення.

15.3.1. Сингулярне розкладання

Довільну $m \times n$-матрицю рангу n можна представити у вигляді сингулярного розкладання (singular value decomposition, SVD)

$$F = VDU^T$$

яка має чудові властивості:

1) $n \times n$-матриця D діагональна, $D = diag\left(\sqrt{\lambda_1},...,\sqrt{\lambda_n}\right)$,

де $\lambda_1,...,\lambda_n$ – загальні ненульові власні значення матриць F^TF і FF^T.

2) $m \times n$ -матриця $V = \left(v_1,...,v_n\right)$ ортогональна, $V^TV = I_n$ стовпці v_j є власними векторами матриці FF^T, що відповідають $\lambda_1,...,\lambda_n$;

3) $n \times n$-матриця $U = \left(u_1...,u_n\right)$ ортогональна, $U^TU = I_n$ стовпці u_j є власними векторами матриці F^TF, що відповідають $\lambda_1,...,\lambda_n$;

Маючи сингулярне розкладання, легко записати псевдообернену матрицю:

$$F^+ = \left(UDV^TVDU^T\right)^{-1} UDV^T = UD^{-1}V^T = \sum_{j=1}^{n} \frac{1}{\sqrt{\lambda_j}} u_j v_j^T$$

Вектор розв'язку за методом найменших квадратів

$$\alpha^* = F^+y = UD^{-1}V^Ty = \sum_{j=1}^{n} \frac{1}{\sqrt{\lambda_j}} u_j \left(v_j^Ty\right); \qquad (5)$$

Вектор $F\alpha^*$ – МНК- апроксимація цільового вектора y:

$$F\alpha^* = P_Fy = \left(VDU^T\right)UD^{-1}V^Ty = \sum_{j=1}^{n} v_j \left(v_j^Ty\right) \qquad (6)$$

і норма вектора коефіцієнтів:

$$\left\|\alpha^*\right\|^2 = y^TVD^{-1}U^TUD^{-1}V^Ty = y^TVD^{-2}V^Ty = \sum_{j=1}^{n} \frac{1}{\lambda_j}\left(v_j^Ty\right)^2 \qquad (7)$$

Отже, якщо є сингулярне розкладання, то обертати матриці вже не потрібно. Однак обчислення сингулярного розкладання практично настільки ж трудомістко, як і обернення. Ефективні чисельні алгоритми, що обчислюють SVD, реалізовані в багатьох стандартних математичних пакетах.

15.3.2. Проблема мультиколінеарності

Якщо коваріаційна матриця $\Sigma = F^TF$ має неповний ранг, то її обернення неможливе. Тоді доводиться відкидати лінійно залежні ознаки або застосовувати регуляризацію. На практиці частіше зустрічається проблема

мультиколінеарності – коли матриця Σ має повний ранг, але близька до деякої матриці неповного рангу. Тоді говорять, що Σ – матриця *неповного псевдорангу* або що вона *погано обумовлена*. Геометрично це означає, що об'єкти вибірки зосереджені поблизу лінійного підпростору меншої розмірності $k < n$. Ознакою мультиколінеарності є наявність у матриці Σ власних значень, близьких до нуля.

Число обумовленості матриці Σ є

$$\mu\left(\Sigma\right) = \left\|\Sigma\right\|\left\|\Sigma^{-1}\right\| = \frac{\max\limits_{u:\left\|u\right\|=1}\left\|\Sigma u\right\|}{\min\limits_{u:\left\|u\right\|=1}\left\|\Sigma u\right\|} = \frac{\lambda_{\max}}{\lambda_{\min}}$$

де $\lambda_{\max}$ і $\lambda_{\min}$ – максимальне й мінімальне власні значення матриці Σ, всі норми евклидові. Матрицю вважають погано обумовленою, якщо $\mu\left(\Sigma\right) \geq 10^2 ... 10^4$. Обернення такої матриці чисельно нестійке. При множенні зворотної матриці на вектор, $z = \Sigma^{-1}u$, відносна погрішність підсилюється в $\mu\left(\Sigma\right)$ раз:

$$\frac{\left\|\delta z\right\|}{\left\|z\right\|} \leq \mu\left(\Sigma\right)\frac{\left\|\delta u\right\|}{\left\|u\right\|}$$

Саме це й відбувається із МНК-Розв'язком у випадку поганої обумовленості. У формулі (7) близькі до нуля власні значення виявляються в знаменнику, у результаті збільшується розкид коефіцієнтів α^* , з'являються великі по абсолютній величині позитивні й негативні коефіцієнти. МНК-розв'язок стає нестійким – малі погрішності вимірювання ознак або відповідей у навчальних об'єктів можуть суттєво вплинути на вектор розв'язку α^* , а погрішності виміру ознак у тестового об'єкта x – на значення функції регресії $g\left(x, \alpha^*\right)$. Мультиколінеарність тягне за собою не тільки нестійкість і перенавчання, але й неможливість інтерпретації коефіцієнтів, тому що по абсолютній величині коефіцієнта α_j стає неможливо судити про ступінь важливості ознаки f_j .

15.3.3. Гребенева регресія

Для розв'язку проблеми мультиколінеарності додамо до функціонала Q регуляризатор, що штрафує більші значення норми вектора ваг $\left\|\alpha\right\|$:

$$Q_\tau\left(\varepsilon\right) = \left\|F\alpha - y\right\|^2 + \tau\left\|\alpha\right\|^2$$

де τ – невід'ємний параметр. У випадку мультиколінеарності є нескінченно багато векторів α, що доставляють функціоналу Q значення,

близькі до мінімального. Штрафний доданок виконує роль регуляризатора, завдяки якому серед них вибирається розв'язок з мінімальною нормою. Дорівнюючи нулю похідну $Q_\tau(\alpha)$ по параметру α, знаходимо:

$$\alpha_\tau^* = \left(F^T F + \tau I_n\right)^{-1} F^T y$$

Таким чином, перед оберненням матриці до неї додається «гребінь» – діагональна матриця τI_n . Звідси й назва методу – *гребенева регресія* (ridge regression). При цьому всі її власні значення збільшуються на τ, а власні вектори не змінюються. У результаті матриця стає добре обумовленою, залишаючись у той же час «схожою» на початкову.

Виразимо регуляризований МНК-розв'язок через сингулярне розкладання:

$$\alpha_\tau^* = \left(UD^2U^T + \tau I_n\right)^{-1} UDV^T y = U\left(D^2 + \tau I_n\right)^{-1} DV^T y = \sum_{j=1}^{n} \frac{\sqrt{\lambda_j}}{\lambda_j + \tau} u_j \left(v_j^T y\right)$$

Тепер знайдемо регуляризованную Мнк-Апроксимацію цільового вектора y:

$$F\alpha_\tau^* = VDU^T \alpha_\tau^* = Vdiag\left(\frac{\lambda_i}{\lambda_j + \tau}\right) V^T y = \sum_{j=1}^{n} \frac{\lambda_j}{\lambda_j + \tau} v_j \left(v_j^T y\right) \qquad (8)$$

Як і колись в (6), МНК-апроксимація представляється у вигляді розкладання цільового вектора y по базису власних векторів матриці FF^T . Тільки тепер проекції на власні вектори скорочуються, множачись на $\frac{\lambda j}{\lambda_j + \tau} \in (0,1)$. У порівнянні з (7) зменшується й норма вектора коефіцієнтів:

$$\left\|\alpha_\tau^*\right\|^2 = \left\|D^2\left(D^2 + \tau I_n\right)^{-1} D^{-1} V^T y\right\|^2 = \sum_{j=1}^{n} \frac{1}{\lambda_j + \tau}\left(v_j^T y\right) < \sum_{j=1}^{n} \frac{1}{\lambda_j}\left(v_j^T y\right)^2 = \left\|\alpha^*\right\|^2$$

Звідси ще одна назва методу – стискання (shrinkage) або скорочення ваг (weight decay).

Поняття ефективної розмірності. З формул видно, що в міру збільшення параметра τ вектор коефіцієнтів α_τ^* стає усе більш стійким і жорстко визначеним. Фактично, відбувається зниження *ефективної розмірності розв'язку* – це другий зміст терміну «стискання».

Можна показати, що роль розмірності відіграє слід проекційної матриці. Дійсно, у нерегуляризованому випадку маємо

$$trF\left(F^T F\right)^{-1} F^T = tr\left(F^T F\right)^{-1} F^T F = trI_n = n$$

При використанні регуляризації ефективна розмірність набуває значення від 0 до n, не обов'язкове ціле, і убуває при зростанні τ:

$$n_{эфф} = trF\left(F^TF + \tau I_n\right)^{-1} F^T = tr\,diag\left(\frac{\lambda_j}{\lambda_j + \tau}\right) = \sum_{j=1}^{n}\frac{\lambda_j}{\lambda_j + \tau} < n$$

Проблема вибору константи регуляризації. При $\tau \to 0$ регуляризований розв'язок прямує до МНК-розв'язку: $\alpha_\tau^* \to \alpha^*$. При $\tau \to \infty$ надмірна регуляризація призводить до виродженого розв'язку: $\alpha_\tau^* \to 0$. Обидва крайніх випадку небажані, тому оптимальним є деяке проміжне значення τ^*. Для його знаходження можна застосовувати ковзний контроль. Залежність оцінки ковзного контролю від параметра τ , як правило, має характерний мінімум.

Ковзний контроль – обчислювально трудомістка процедура. Відома практична рекомендація брати τ у відрізку [0.1, 0.4], якщо стовпці матриці F заздалегідь стандартизовані (центровані й нормовані). Ще одна евристика – вибрати τ так, щоб число обумовленості прийняло задане не занадто велике значення: $M_0 = \mu\left(F^TF + \tau I_n\right) = \frac{\lambda_{\max} + \tau}{\lambda_{\min} + \tau}$ звідки випливає рекомендація $\tau^* \approx \frac{\lambda_{\max}}{M_0}$

15.4. Нелінійні методи відновлення регресії

Припущення про те, що модель регресії лінійна по параметрах, зручно для побудови чисельних методів, але не завжди добре узгоджується зі знаннями про предметну область. Нелінійними будемо називати випадки, коли модель регресії нелінійна по параметрах, коли в лінійну модель додаються нелінійні перетворення вихідних ознак або цільової ознаки, а також коли вводиться неквадратична функція втрат.

Загальна ідея у всіх цих випадках одна: нелінійна задача зводиться до розв'язку послідовності більш простих лінійних задач.

15.4.1. Нелінійна модель регресії

Нехай задана нелінійна модель регресії $f\left(x,\alpha\right)$ і потрібно мінімізувати функціонал якості по вектору параметрів $\alpha \in R^p$:

$$Q\left(\alpha, X_m\right) = \sum_{i=1}^{m}\left(f\left(x_i,\alpha\right) - y_i\right)^2$$

Для виконання чисельної мінімізації функціонала Q скористаємося методом Ньютона-Рафсона. Виберемо початкове наближення $\alpha^0 = \left(\alpha_1^0,\ldots,\alpha_p^0\right)$ і організуємо ітераційний процес

$$\alpha^{t+1} := \alpha^t - h_t\left(Q''\left(\alpha^t\right)\right)^{-1} Q'\left(\alpha^t\right),$$

де $Q'\left(a^t\right)$ -градієнт функціонала Q у точці $\alpha^t, Q''\left(\alpha^t\right)$-гесіан (матриця других похідних) функціонала Q у точці α^t, h^t – величина кроку, який можна регулювати, а в найпростішому варіанті просто вважається рівним одиниці.

Запишемо компоненти градієнта:

$$\frac{\partial}{\partial\alpha_j}Q\left(\alpha\right)=2\sum_{i=1}^{m}\left(f\left(x_i,\alpha\right)-y_i\right)\frac{\partial}{\partial\alpha_j}\left(x_i,\alpha\right).$$

Запишемо компоненти гесіана:

$$\frac{\partial^2}{\partial\alpha_j\partial\alpha_k}Q\left(\alpha\right)=2\sum_{i=1}^{m}\frac{\partial f}{\partial\alpha_j}\left(x_i,\alpha\right)\frac{\partial f}{\partial\alpha_k}\left(x_i,\alpha\right)-\underbrace{2\sum_{i=1}^{m}\left(f\left(x_i,\alpha\right)-y_i\right)\frac{\partial^2 f}{\partial\alpha_j\partial\alpha_k}\left(x_i,\alpha\right)}_{\text{при лінеаризації}\ =\ 0}$$

Оскільки функція f задана, градієнт і гесіан легко обчислюються чисельно, то основна складність методу Ньютона-Рафсона полягає в обернені гесіана на кожній ітерації.

Більш ефективної з обчислювальної точки зору є наступна модифікація цього методу. Якщо функція f досить гладка (двічі неперервно диференційована), то її можна лінеаризувати в околі поточного значення вектора коефіцієнтів α^t :

$$f\left(x_i,\alpha\right)=f\left(x_i,\alpha^t\right)+\sum_{j=1}^{p}\frac{\partial f}{\partial\alpha_j}\left(x_i,\alpha_j\right)\left(\alpha_j-\alpha_j^t\right)$$

Замінимо в гесіані функцію f на її лінеаризацію. Це однаково, що покласти другий доданок у гесіані рівним нулю. Тоді не потрібно буде обчислювати другі похідні $\frac{\partial^2 f}{\partial\alpha_j\partial\alpha_k}\left(x_i,\alpha\right)$. Цей метод називають методом Ньютона-Гауса. В іншому він нічим не відрізняється від методу Ньютона-Рафсона.

Введемо матричні позначення:

$F_t=\left(\frac{\partial f}{\partial\alpha_j}\left(x_i,\alpha^t\right)\right)_{\substack{i=1,m\\j=1,p}}$ – матриця перших похідних розміру $m\times p$ на t -й ітерації;

$f_t=\left(f\left(x_i,\alpha^t\right)\right)_{i=1,m}$ – вектор значень апроксимуючої функції на t -й ітерації.

Тоді формула t-ї ітерації методу Ньютона- Гауса в матричному записі матиме вигляд:

$$\alpha^{t+1} := \alpha^t - h_t \underbrace{\left(F_t^T F_t\right)^{-1} F_t^T \left(f^t - y\right)}_{\delta}$$

У правій частині записаний розв'язок стандартної задачі многомірної лінійної регресії

$$\left\| F_t \delta - \left(f^t - y\right) \right\|^2 \to \min_{\delta}$$

Таким чином, у методі Ньютона-Гауса нелінійна регресія зводиться до послідовності лінійних регресійних задач. Швидкість збіжності в нього практично така ж, як і в методу Ньютона-Рафсона (обоє є методами другого порядку), але обчислення трохи простіше й виконуються стандартними методами лінійної регресії.

15.4.2. Нелінійні одновимірні перетворення ознак

На практиці зустрічаються ситуації, коли лінійна модель регресії представляється необґрунтованою, але запропонувати адекватну нелінійну модель $f\left(x,\alpha\right)$ також не вдається. Тоді в якості компромісу будується модель виду

$$f\left(x,\alpha\right) = \sum_{j=1}^{n} \phi_j \left(f_j\left(x\right)\right)$$

де $\phi_j : R \to R$ – деякі перетворення початкових ознак, у загальному випадку нелінійні. Задача полягає в тому, щоб підібрати невідомі одновимірні перетворення ϕ_j , при яких досягається мінімум квадратичного функціонала (15.1).

Метод налаштування з поверненнями (backfitting) запропонований Хасти й Тибширани в 1986 році.

Схема реалізації показана в алгоритмі 2.

Вхідні дані:

F – матриця « об'єкти-ознаки»

y – вектор відповідей;

Вихідні дані:

$\phi_j\left(x\right)$ – функції перетворення ознак, у загальному випадку нелінійні.

Хід роботи алгоритму:

1: нульове наближення:

$\alpha :=$ розв'язок задачі МЛР з ознаками $f_j\left(x\right)$;

$\phi_j\left(x\right) := \alpha_j f_j\left(x\right),\ \ j = 1,2,\ldots,n$;

2: **повторювати**

3: **для** $j = 1,2,\ldots,n$

4: $z_i := y_i - \sum_{k=1,k\neq j}^{n} \phi_k\left(f_k\left(x_i\right)\right),\ i = 1,\ldots,m;$

5: $\phi_j := \arg\min_{\phi} \sum_{i=1}^{m}\left(\phi\left(f_j\left(x\right)\right) - z_i\right)^2;$

6: $Q_j := \sum_{i=1}^{m}\left(\phi_j\left(f_j\left(x\right)\right) - z_i\right)^2;$

7: **поки** значення Q_j не стабілізуються

Метод заснований на ітераційному уточненні функцій ϕ_j. На першому кроці вони покладаються лінійними, $\phi_j\left(x\right) = \alpha_j f_j\left(x\right)$, і невідомі коефіцієнти α_j налаштовуються методами багатовимірної лінійної регресії. На кожному наступному кроці вибирається одна з функцій ϕ_j, усі інші фіксуються, і обрана функція будується заново. Для цього вирішується стандартна задача найменших квадратів

$$Q\left(\phi_j, X_m\right) = \sum_{i=1}^{m}\left(\phi_j f_j\left(x_i\right)\right) - \underbrace{\left(y_i - \sum_{\substack{k=1\\k\neq j}}^{n} \phi_k\left(f_k\left(x_i\right)\right)\right)^2}_{z_i = const\left(\phi_j\right)} \to \min_{\phi_j}$$

з навчальною вибіркою $Z_j^m = \left(f_j\left(x_i\right), z_i\right)_{i=1}^{m}$. Для розв'язку даної задачі підходять будь-які одновимірні методи: ядерне згладжування, сплайни, поліноміальна або Фурье-апроксимація.

Запитання до розділу 15

1. Опишіть критерії вибору ядра і ширини вікна при застосуванні формули Надарая-Ватсона.
2. На яких принципах побудовано алгоритм локально зваженого згладжування LOWESS?
3. Опишіть причину реалізації сингулярного розкладання матриці.
4. В чому полягає проблема мультиколінеарності?
5. Які ви знаєте нелінійні методи відновлення регресії.

ЛІТЕРАТУРА

1. Воронцов К.В. Введение в машинное обучение // https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie
2. Paliouras G. Machine Learning and Its Applications / G. Paliouras, V. Karkaletsis, C.D. Spyropoulos, “Springer”, 2001. – 334 p.
3. Флах П. Машинное обучение: наука и искусство построения алгоритмов, которые извлекают знания из данных / П. Флах, «ДМК Пресс», 2015. – 400 с.
4. Воронцов К.В. Математические методы обучения по прецедентам // http://www.machinelearning.ru/wiki/index.php?title=%D0%A3%D1%87%D0%B0%D1%81%D1%82%D0%BD%D0%B8%D0%BA:Vokov
5. Воронцов К.В. Курс «Машинное обучение» 2019 (Школа анализа данных) // https://ya-r.ru/2020/05/07/vorontsov-kurs-mashinnoe-obuchenie-2019-shkola-analiza-dannyh/
6. Hastie T. The Elements of Statistical Learning: Data Mining, Inference, and Prediction / T. Hastie, R. Tibshirani, J. Friedman , “Springer”, 2016 . – 767 p.
7. Mohri M. Foundations of Machine Learning / M. Mohri , A. Rostamizadeh, A. Talwalkar, “The MIT Press”, 2012 . – 412 p.
8. Haykin S. Neural Networks and Learning Machines / S. Haykin, “Pearson”, 2008. – 936 p.
9. Murphy K.P. Machine Learning: A Probabilistic Perspective / K.P. Murphy, “The MIT Press”, 2012 . – 1104 p.
10. Bresset E. SciPy and NumPy: An Overview for Developers / E. Bresset, O'Reilly Media, 2012. – 57 p.
11. Harrison M. Learning the Pandas Library: Python Tools for Data Munging, Analysis, and Visualization / M. Harrison, CreateSpace Independent Publishing Platform, 2016. – 212 p.
12. Yim A. Matplotlib for Python Developers / A.Yim, C. Chung, A. Yu, Packt Publishing, 2018. – 300 p.
13. Molin S. Hands-On Data Analysis with Pandas / S. Molin, Packt Publishing, 2019. – 740 p.
14. Gooodfellow I. Machine Learning. Basics / I. Gooodfellow, 2015 // https://www.deeplearningbook.org/slides/05_ml.pdf
15. Daume H. A Course in Machine Learning / H. Daume / 2012 // http://ciml.info/dl/v0_9/ciml-v0_9-ch03.pdf
16. Бастиан Ш. Крупномасштабное машинное обучение вместе с Pyton / Ш. Бастиан, «ДМК Пресс»б 2017. – 358 с.
17. Nilson N. J. Introduction to Machine Learning / N.J. Nilson, Stanford University, 1998 // http://robotics.stanford.edu/~nilsson/MLBOOK.pdf?roistat_visit=10865700

18. Barber D. Bayesian Reasoning and Machine Learning / D. Barber, 2010 // http://web4.cs.ucl.ac.uk/staff/D.Barber/textbook/090310.pdf?roistat_visit=10865700
19. Pandas documentation // https://pandas.pydata.org/docs/
20. Python documentation // https://docs.python.org/3/