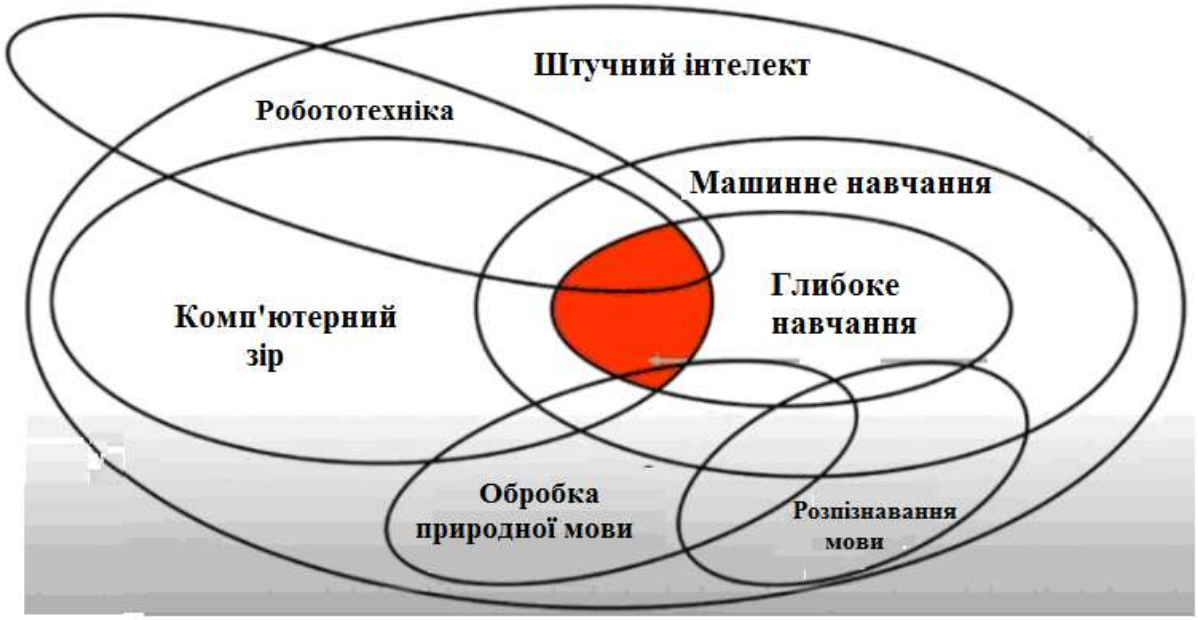
М. М. Биков, В. В. Ковтун, Т. В. Грищук



ОСНОВИ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ

Частина 2 . Технології машинного навчання

Міністерство освіти і науки України
Вінницький національний технічний університет

М. М. Биков, В. В. Ковтун, Т. В. Грищук

ОСНОВИ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ

Частина 2. Технології машинного навчання

Електронний навчальний посібник
комбінованого (локального та мережного) використання

Вінниця
ВНТУ
2024

УДК 681.3.07
Б60

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 12 від 30.05.2024 р.)

Рецензенти:
Т. О. Говорущенко, доктор технічних наук, професор
О. І. Бісікало, доктор технічних наук, професор
О. М. Козачко, кандидат технічних наук, доцент

Биков, М. М.

Б60 Основи інтелектуальних технологій. Частина 2. Технології машинного навчання : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс] / Биков М. М., Ковтун В. В., Грищук Т. В. – Вінниця : ВНТУ, 2024. – 153 с.

У Частині 2 посібника наводяться теоретичні відомості, що дають можливість ознайомитись з колом основних задач машинного навчання, моделями, які їх описують для даних різного типу, алгоритмами реалізації таких задач для різних моделей; принципами попередньої підготовки даних та способами оцінювання якості розв’язків задач машинного навчання. Теоретичний матеріал супроводжується прикладами практичних задач і їх розв’язанням, а також алгоритмами вирішення окремих задач з використанням технологій машинного навчання.

Матеріал посібника буде корисним здобувачам вищої освіти баклаврського і магістерського рівня, які навчаються за спеціальностями 151 «Автоматизація та ком’ютерно-інтегровані технології» і 174 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка», під час вивчення дисциплін «Основи інтелектуальних технологій» та «Машинне навчання і аналіз даних».

УДК 681.3.07

ЗМІСТ

РОЗДІЛ 8
МАШИННЕ НАВЧАННЯ І ЙОГО ВИДИ

8.1 Методи машинного навчання і їх класифікація

Останнім часом машинне навчання є основним напрямком сучасних комп'ютерних технологій. Можна впевнено сказати, що, крім уміння програмувати, саме знання основ і практичного застосування методів машинного навчання стає стандартом для IT фахівця.

Терміном **«машинне навчання» (англ. Mashine learning, ML) на сьогодні визначають клас методів штучного інтелекту, характерною рисою яких є не пряме розв'язання задачі, а навчання в процесі застосування розв'язків множини подібних задач.** Предметом цього розділу ШІ є створення систем, здатних отримувати знання з даних, а також шляхом навчання покращувати показники своєї роботи. Для побудови методів ML використовуються засоби алгебри, математичної статистики, дискретної математики, теорії оптимізації, чисельних методів і інших розділів математики.

Метою машинного навчання є добування знань з даних. Ця наукова сфера знаходиться на перетині комп'ютерних наук, статистики та штучного інтелекту і інакше відома як статистичне навчання або прогнозна аналітика. На сьогодні словосполучення «машинне навчання» є більш поширеним, ніж слова «штучний інтелект», оскільки поняття «штучний інтелект» застосовується найчастіше до задачі розробки інтелектуальних роботів. Нині стало повсякденним явищем використання методів ML в повсякденному житті. Алгоритми машинного навчання застосовує велика кількість сучасних пристроїв і веб-сайтів, починаючи з автоматичних рекомендацій щодо перегляду фільмів, замовлення їжі чи купівлі продуктів і закінчуючи розпізнаванням осіб на фотографіях та персоналізованими онлайн радіотрансляціями. Якщо вникнути в роботу таких складних сайтів як Facebook, Amazon або Netflix, то можна побачити, що кожен розділ сайту містить кілька моделей машинного навчання.

Вийшовши за межі комерційних додатків, машинне навчання вже справило величезний вплив на наукові дослідження, керовані даними. Його інструменти використовувалися для вирішення різних наукових завдань (дослідження зірок, пошук далеких планет, відкриття нових частинок, аналіз послідовностей ДНК, розробка персоналізованих методів лікування раку).

Розрізняють два типи навчання:

1. Індуктивне навчання, або навчання по прецедентах, яке грунтується на знаходженні емпіричних закономірностей в даних.

2. Дедуктивне навчання, що грунтується на формалізації знань експертів і формуванні на їх основі бази знань в комп'ютері.

Останній тип навчання зазвичай відносять до сфери експертних систем, тому терміни навчання по прецедентах і машинне навчання можна вважати синонімами.

Багато методів індуктивного навчання розроблялися як альтернатива класичним статистичним підходам. Частина методів тісно пов'язані з витяганням інформації шляхом інтелектуального аналізу даних (data mining).

Загальна постановка задачі навчання на прецедентах описується в такому вигляді. Є в наявності множина об'єктів чи ситуацій і множина можливих відповідей (реакцій на них). Також існує деяка залежність між відповідями і об'єктами, але вона не є відомою. Відомою є також кінцева сукупність прецедентів – пар «об'єкт, відповідь», її називають навчальною вибіркою. Грунтуючись на цих даних, потрібно знайти неявну залежність шляхом побудови алгоритму, здатного для всіх можливих вхідних об'єктів видати достатньо точну відповідь з його класифікації. Ця залежність не обов'язково виражається аналітично, і тут методи ML застосовують методи емпіричного формування рішення. За такої умови важливо отримати для навченої системи здатність до узагальнення, тобто до правильного відгуку на дані, що не входили до наявної навчальної вибірки. Для вимірювання правильності відгуків використовується вибраний оціночний функціонал якості.

З одного боку, розділ машинного навчання утворився внаслідок розподілу науки про нейромережі на парадигми навчання мереж та види топологій їхньої архітектури, а з іншого боку – увібрав в себе методи матстатистики. Класифікація способів машинного навчання витікає переважно із способу використання нейромереж. Проте існують й інші методи, основані на використанні навчальної вибірки, зокрема, дискримінантний аналіз, що оперує узагальненою дисперсією і коваріацією спостережуваної статистики, або байєсовські класифікатори. Такі базові види нейромереж, як перцептрон чи багатошарова мережа (і їх модифікації), можуть навчатися як без учителя, так і з учителем, з підкріпленням і самоорганізацією. Більшість статистичних методів і частину нейромереж можна віднести тільки до одного із способів навчання.

Для орієнтації в виборі з великої кількості методів ML потрібного для розв’язання тієї чи іншої задачі необхідно виконати їх класифікацію. Один із способів класифікації наведено на рис. 8.1.

Всі подані методи можна поділити на чотири основних напрямки:

1. Класичне навчання. Таке навчання застосовують для вирішення нескладних задач, які мають справу з простими даними і зрозумілими ознаками, які використовуються для їх опису.

2. Ансамблі. Використовують у випадку низької якості опису даних.

3. Нейромережі і глибоке навчання. Ці методи використовують для вирішення задач, які містять складні дані, ознаки для опису яких здебільшого апріорі невідомі.

4. Навчання з підкріпленням. Використовується в задачах, в яких відсутні початкові дані, але є опис середовища, з яким взаємодіє об’єкт-агент. Використовується здебільшого в задачах проектування інтелектуальних роботів різного призначення.

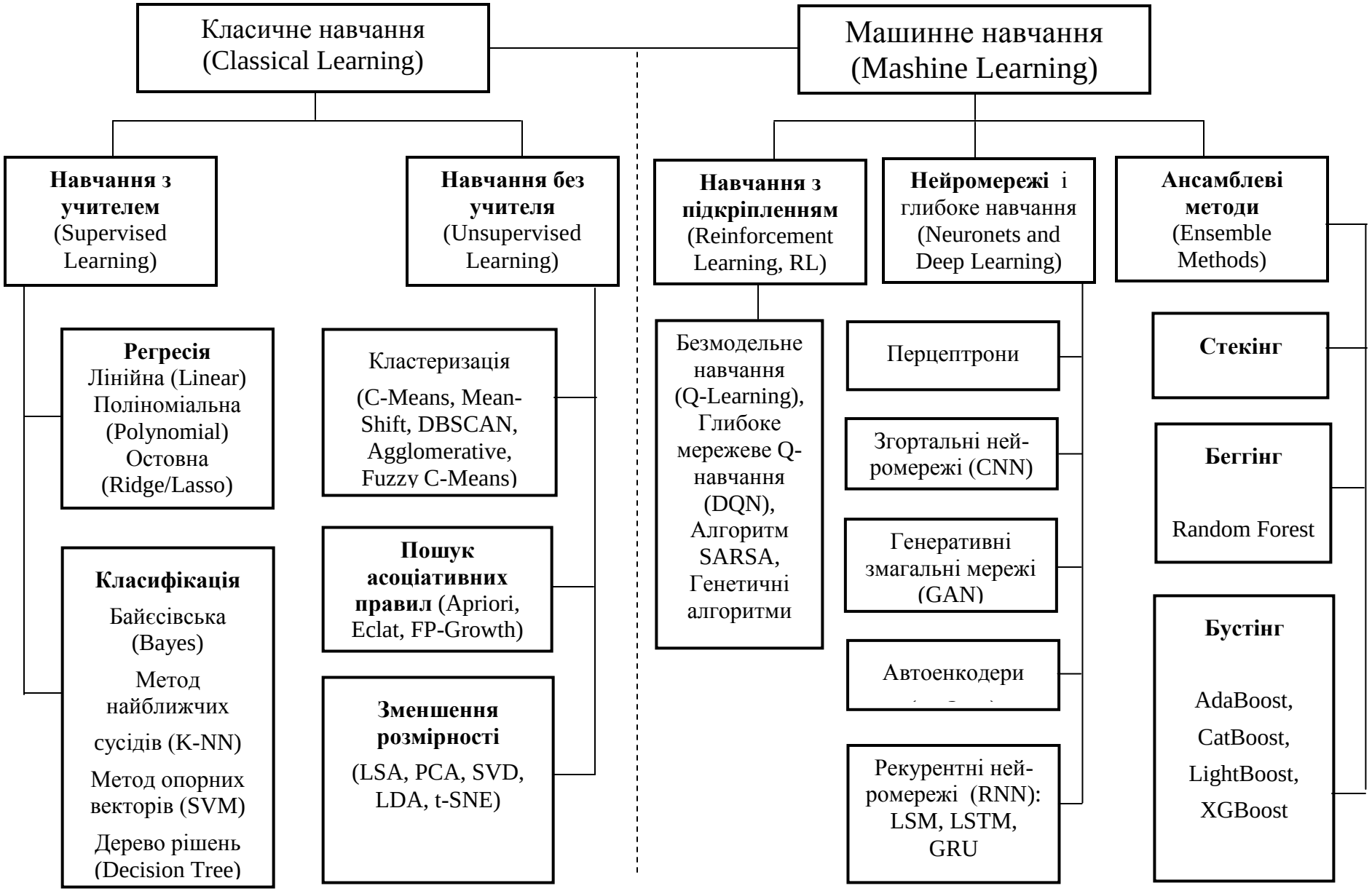


Рисунок 8.1 – Класифікація методів машинного навчання

8.2 Коротка характеристика методів машинного навчання

Традиційно машинне навчання поділяють на навчання з учителем (supervised), навчання без вчителя (unsupervised) і навчання з підкріпленням (reinforcement learning). Розглянемо їх роботу і застосування, для чого наведемо короткий опис методів навчання, поданих в класифікаційній схемі на рис. 8.1.

8.2.1 Навчання з учителем. Навчання з учителем – найрозвиненіший і популярний напрям машинного навчання. Основна ідея такого навчання полягає в тому, що задається набір вхідних параметрів і очікуваний результат. Таким чином учитель навчає алгоритм правильних відповідей – звідси і назва.

Для навчання з учителем потрібно мати марковані (labeled) дані. Це означає, що поряд з вхідними параметрами, дані мають містити відповіді, задані мітками (labels). Наприклад, для задачі прогнозування курсу валют міткою буде слугувати значення курсу обміну валют.

Математичну постановку задачі навчання з учителем для задач класифікації методами опорних векторів (SVM) і Байєса наведено відповідно в розділах 3 і 4 першої частини посібника, навчання класифікаторів і регресійного аналізу – відповідно в розділах 9 і 11 цієї частини посібника. Загальну постановку задачі машинного навчання з учителем можна виконати таким чином:

Задано

G=(X,Y); $\underline{U}$={(x_i, y_k)}; $x_i \in X$; $y_k \in Y$; i = 1, . . . , m; k=1, …, n,

де X – множина об'єктів;

Y – множина реакцій (міток об’єктів, тобто можливих відповідей);

U – навчальна вибірка, в якій кожен прецедент (пара об’єкт – мітка) відповідає правильній реакції y_k на об’єкт x_i.

Необхідно:

Розробити алгоритм машинного навчання A: X → Y, що будує модель у вигляді вирішувальної функції F, яка знаходить правильні відповідно до деякого критерію якості реакції y∈Y не лише на навчальній вибірці, а й на всій множині об’єктів X.

Для різних типів задач множина об'єктів X та відповідей Y може мати різний характер. Наприклад, для задачі класифікації можуть існувати такі випадки:

1) Y = {−1; +1} – класифікація на два класи;
2) Y = {1, . . . , M} – класифікація на M класів, що не перетинаються.

Для задачі регресії:

1) Y = R^n – неперервні значення вихідної величини в n – вимірному просторі;
2) Y = R_m, m=1, … , k – дискретні значення вихідної величини.

Для задачі ранжування: Y – скінченна упорядкована множина. Зазвичай множину об'єктів X визначають не самими об'єктами, а їх описами, найчастіше – за допомогою ознакового опису у вигляді векторів ознак.

Спрощено можна сказати, що в навчанні з учителем для створення моделі задаються запитання і надаються відповіді на них. Після того як модель побудовано, можна ставити нові запитання та отримувати на них відповіді. Під моделюванням в такому випадку розуміється використання алгоритму машинного навчання для пошуку інформації в існуючих даних для конкретної прикладної задачі.

Рисунок 8.2 – Побудова моделі за допомогою машинного навчання з учителем

Процедура реалізації машинного навчання з учителем ілюструється рисунком 8.2.

Найбільш поширеними і вживаними у практичному використанні вважаються алгоритми навчання з учителем, які вирішують задачі, що відносяться до класифікації та регресійного аналізу.

Регресія. Класифікаційні алгоритми працюють тільки для тих випадків, де у нас є обмежений набір можливих результатів. Вони не підходять для випадків, коли результатом має бути число, яке ми намагаємося передбачити. Прикладом такої задачі є прогнозування курсу валют. В цій задачі є часовий ряд значень і відповідний їм ряд числових значень курсів валют як відповідних вхідних параметрів та вимога передбачити числове значення курсу валют. Курс валют у цьому випадку подано набором чисел в деякому неперервному діапазоні.

Для вирішення таких задач існують алгоритми ***регресійного аналізу***.

Вони використовуються для побудови лінійної, нелінійної, багатофакторної та логістичної моделей регресії. Остання, як правило, застосовується для бінарної класифікації. Лінійну модель регресії визначають у вигляді прямої лінії, яка найкращим чином відображає залежність між вхідними та вихідними змінними. Для складання цього

рівняння потрібно визначити відповідні коефіцієнти для вхідних змінних. У випадку, коли простір ознак є неперервним, а простір відповідей – дискретним (наприклад, поданий набором класів), доцільно використовувати логістичну модель регресії. У логістичних моделях регресії вихідні дані перетворюються за допомогою відповідних логістичних функцій. Математичний апарат, що застосовується в цих моделях, містить метод Ньютона-Рафсона. Багатофакторна модель регресії побудована на основі матричного подання та її сингулярного розкладу.

Для побудови моделі навчання у вигляді регресійної функції дата-інженер реалізує процес, поданий на рис. 8.2. Він збирає і готує належним чином дані, які містять вхідні параметри і правильні відповіді. Ці дані завантажуються в алгоритм регресійного аналізу, який створює навчену модель. Отриману після навчання модель можна використовувати для прогнозування нових значень, використовуючи нові вхідні параметри.

До найбільш поширених напрямків застосування регресійного аналізу відносяться прогнозування цін на акції, прогнозування курсів валют, оцінення вартості нерухомості, оцінення вартості старих авто, передбачення енергоспоживання будівель, прогнозування попиту на товари в роздрібних мережах, оцінення вартості лотів в аукціонах, оцінення часу очікування подачі машини. Детально принципи побудови регресійних моделей методами машинного навчання розглянуто в розділах 11-13 цього посібника

Класифікація. Задачі навчання класифікаторів застосовуються у випадках, коли є великі обсяги даних, припустимо – тисячі фотографій домашніх тварин з маркерами (мітками, ярликами): це кішка, а це собака. На практиці задачі класифікації є досить поширеними. Алгоритми їх вирішення відповідають на питання, входить щось в обмежений набір відповідей чи ні. Наприклад, у нас є зображення і нам потрібно класифікувати об'єкт на ньому (рис. 8.3). Наприклад, що є на зображенні – кіт чи собака? Або ж в медичній діагностиці класифікатор має визначити наявність або відсутність певного захворювання у пацієнта.

За допомогою машинного навчання необхідно створити модель, за використання якої машина могла б за фотографією, яку «не бачила» раніше, визначити, хто на ній зображений: кішка або собака. У ролі «вчителя» в цьому випадку виступає людина, яка заздалегідь проставила мітки. В процесі контрольованого навчання машина може сама вибрати ознаки, за якими вона відрізняє кішок від собак. В деяких випадках задача вибору інформативних ознак для опису об’єктів, що підлягають розпізнаванню, вирішується попередньо дослідником. У випадку, коли машина сама вибирає ознаки для розрізнення об’єктів, що підлягають класифікації, побудована нею модель в подальшому може бути швидко переналаштована на вирішення іншої задачі, наприклад, на розпізнавання курей і качок. Як приклад можна назвати моделі машинного навчання у вигляді згорткових штучних нейронних мереж, які на сьогодні широко

використовуються для розпізнавання символів тексту і зорових зображень різних об’єктів.

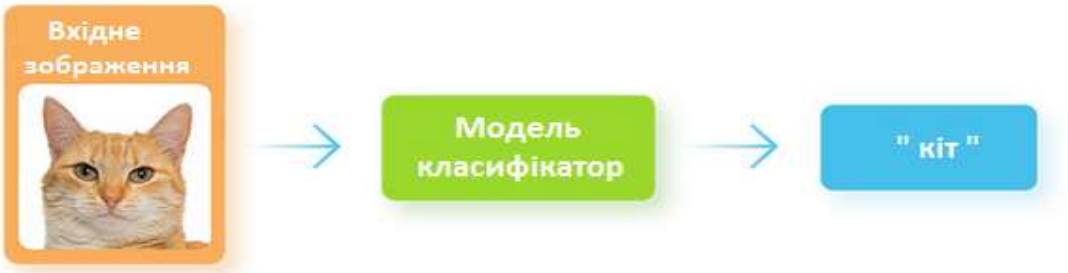


Рисунок 8.3 – Використання моделі класифікатора для визначення об'єкта

Стандартне обмеження класифікаційних алгоритмів полягає в тому, що вони можуть давати відповіді тільки на ті питання, за якими вони навчалися. Наприклад, якщо ви задали безліч зображень з котами і промаркували їх як такі, на яких є коти, кінцева модель буде здатна визначати котів на нових зображеннях. Але визначити собак вона не зможе.

Байєсівська класифікація. Алгоритм «Наївний Байєсівський класифікатор» базується на припущенні, що кожна вхідна змінна є незалежною згідно з теоремою Байєса. Ця модель використовує два типи ймовірностей – ймовірність для кожного класу та умовну ймовірність для кожного класу для всіх значень вхідної змінної, що розраховуються на тренувальних даних. Після цього модель можна використовувати для нових даних згідно з теоремою Байєса.

Метод найближчих сусідів (K-NN). Алгоритм «К-найближчих сусідів» базується на оціненні подібності об'єктів. Оскільки кожен об'єкт може мати різноманітні ознаки, то основна проблема цього алгоритму полягає у виборі метрики відстані між об'єктами. Цю проблему можна вирішити шляхом відбору невеликої кількості інформативних ознак, для кожної з яких будується своя функція близькості і проводиться їх згортка.

Метод опорних векторів (SVM – support vector mashine) фокусується на ідеї використання поняття гіперплощини, тобто лінії, яка розділяє об’єкти, подані точками у просторі ознак. Суттю є проведення двох прямих між категоріями так, щоб утворився найбільший проміжок між ними. Найкращою гіперплощиною вважається лінія з найбільшою маржею – відстанню між гіперплощиною та найближчими точками даних. Ці точки відомі як опорні вектори і відіграють головну роль у побудові гіперплощини та класифікатора. Для визначення коефіцієнтів гіперплощини, що максимізують відстань, використовуються спеціальні методи оптимізації, зокрема, оптимізаційна процедура з використанням множників Лагранжа. Зараз цей алгоритм вважається найефективнішим

класичним класифікатором і найпопулярнішим для використання у спам-фільтрах.

Дерева рішень. Алгоритми навчання моделей дерев рішень входять до найпопулярніших і потужних інструментів для ефективного розв'язання задач класифікації. Основу роботи дерев рішень становить процес рекурсивного розбиття вхідної множини спостережень або об'єктів на підмножини, що відповідають класам. Дерево рішень будується на основі навчальної вибірки з використанням поняття інформаційної ентропії. Алгоритм «Дерева рішень» можна подати у вигляді двійкового дерева, де кожен вузол є вхідною змінною та точкою розбиття для цієї змінної за умови, що змінна є числом. Існують різні критерії розбиття, найбільш відомі з яких – це міра ентропії та індекс Gini, на основі яких розраховується нормований приріст інформації. Листові вузли є вихідною змінною, що використовується для передбачення, яке проводиться шляхом проходження по дереву до листового вузла і виведення значення класу в цьому вузлі.

Загалом алгоритми класифікації та регресійного аналізу дуже схожі і відрізняються лише потенційними результатами, які вони можуть зробити.

Ансамблеві методи. Під ансамблем розуміють композицію алгоритмів машинного навчання – об'єднання кількох алгоритмів в один. Перевага використання декількох алгоритмів навчання в одному ансамблі полягає в тому, що якщо взяти декілька не дуже ефективних методів навчання і навчити виправляти помилки один одного, то якість навченої моделі буде набагато вищою, ніж за використання кожного з методів окремо.

Кожен алгоритм може вивчати свої закономірності в даних, і якщо алгоритмів багато, то відповідно є багато закономірностей, що є краще. Суть ідеї полягає в тому, щоб навчити алгоритми, а потім усереднити отримані від них відповіді. Головні підходи до побудови ансамблевих моделей містять стекінг (stacking), беггінг (bagging/bootstrap aggregation) та бустинг (boosting).

Стекінг – спочатку навчають кілька моделей різними алгоритмами, потім результати їх роботи показують останньому алгоритму, який приймає остаточне рішення. Як вирішальний алгоритм найчастіше беруть регресію. Стекінг є хорошим, але найменш точним ансамблем серед інших методів, тому на практиці застосовується рідко.

Беггінг (абревіатура від Bootstrapping aGGregatING, об’єднання початкового завантаження) – це тип навчання, коли ансамбль навчається багато разів на випадкових вибірках даних, тобто тренувальні дані розбиваються на множину вибірок, для кожної з яких створюється модель. У кінцевому підсумку усереднюється відповідь за кожною з моделей, що можна порівняти з голосуванням за найбільш популярну відповідь, де багато моделей працюють паралельно. Найбільш поширеним варіантом

цього методу є алгоритм Random Forest, який використовує модель дерева рішень.

Бустинг – це послідовне навчання алгоритмів, коли спочатку навчається перший алгоритм, потім навчається другий, приділяючи особливу увагу місцям, де перший помилився, і так далі до досягнення необхідного результату. Нові моделі додаються, поки тренувальні дані не будуть ідеально передбачатись або не буде досягнуто максимальної кількості моделей. Як і в методі «Беггінг», на кожному етапі навчання формуються групи прецедентів з навчальної вибірки вихідних даних, але не зовсім випадково: у кожну нову групу береться тільки та частина даних, на яких попередній алгоритм відпрацював неправильно. Таким чином, кожен наступний алгоритм донавчається на помилках попереднього.

До переваг цього методу можна віднести дуже високу точність класифікації і більшу швидкість роботи, ніж у нейромереж. До недоліків можна віднести його послідовну, а не паралельну роботу, як у методі «Беггінг». Мінуси вже названі – не паралельний. Хоча все одно працює швидше нейромереж, які як навантажені КамАЗи з піском порівняно з спритним бустінгом.

Найбільш популярними на сьогодні вважаються три методи бустінга: CatBoost, LightGBM і XGBoost.

Нейромережі і глибоке навчання. Штучна нейронна мережа (ШНМ) – це взаємопов'язана сукупність простих обробних елементів (штучних нейронів), які з певним (практично обумовленим) ступенем наближення реалізують функції біологічних нейронів. Нейронні мережі мають здатність до навчання, тобто є інформаційно активними системами.

Нейронні мережі перевершують послідовні машини в рішенні тих самих завдань, в яких машину перевершує людина. Завдання, що потребують великого обсягу обчислень або високої точності, краще виконуються звичайним комп’ютером. До завдань, що успішно вирішуються ШНМ на сучасному етапі їх розвитку відносяться: розпізнавання зорових, слухових образів; розпізнавання тексту і цілей на екрані радара: системи голосового управління; асоціативний пошук інформації та створення асоціативних моделей; синтез мови; формування природної мови; формування моделей і різних нелінійних та інших систем, які важко математично описати; прогнозування розвитку цих систем в часі: застосування на виробництві; прогнозування розвитку циклонів й інших природних процесів, прогнозування змін курсів валют та інших фінансових процесів; системи управління і регулювання з прогнозом; управління роботами, іншими складними пристроями – у вигляді різноманітних скінченних автоматів; системи прийняття рішень і діагностики; фінансова сфера тощо.

8.2.2 Машинне навчання без учителя

Машинне навчання без вчителя намагається знайти відповіді в немаркованих даних (unlabeled data). В цьому випадку для наданих даних правильні відповіді не надаються. Алгоритм має самостійно з'ясувати що-небудь, без попереднього навчання.

Робота алгоритму навчання без учителя. Машинне навчання без вчителя не навчає модель. Замість цього використовуються безпосередньо вхідні параметри. У машинному навчанні без учителя можна виділити такі категорії алгоритмів:

асоціативні;

кластеризація;

зниження розмірності.

Для вирішення **асоціативних** задач дуже популярним є алгоритм Apriori. Він дозволяє знаходити предмети або поняття, які найбільш часто використовуються разом. Таким чином, стандартний функціонал типу «покупці, які придбали це, також придбали ось це», може бути реалізований за допомогою якоїсь з варіацій такого алгоритму (рис. 8.4).

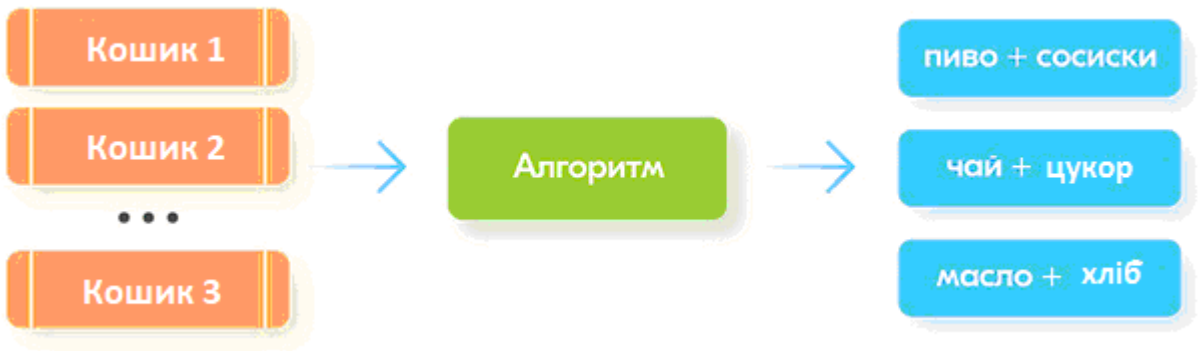


Рисунок 8.4 – Принцип роботи алгоритму асоціативного навчання без учителя

В наведеному прикладі стоїть задача розсортувати інформацію про продукти в різних кошиках і Apriori алгоритм виявить комбінації продуктів, які найбільш часто зустрічаються.

Ця інформація корисна для роздрібних мереж, оскільки для збільшення продажів можливо розмістити такі товари поруч або навіть зробити групу товарів зі знижкою.

Кластеризація. Алгоритми кластеризації дозволяють групувати дані, описані в просторі деяких параметрів, в групи подібних між собою об’єктів, які називаються кластерами (рис. 8.5). Детально деякі алгоритми кластеризації було розглянуто в Розділі 5, Частина 1 цього навчального посібника.

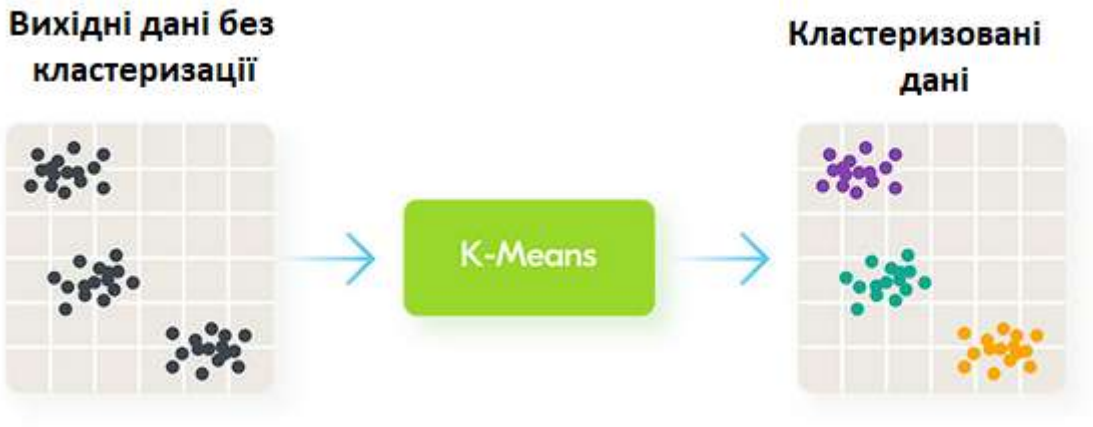


Рисунок 8.5 – Ілюстрація суті процедури кластеризації

Один із досить популярних алгоритмів в цій категорії – це метод *k*-середніх (*k*-Means). Цей алгоритм, описаний в Розділі 5 першої частини посібника, закладено в одну із моделей бібліотеки машинного навчання і працює в один прохід (рис. 8.5). На вхід моделі потрібно внести вхідні дані задачі у вигляді зображень об’єктів в ознаковому просторі, і алгоритм їх згрупує. В наведеному прикладі він використовує дві ознаки, на практиці це може бути ознаковий простір більшої розмірності.

Як приклади популярних задач, які використовували для вирішення кластерний аналіз, можна навести такі:

- групування схожих статей в Google News;
- сегментування ринку для таргетування різних груп покупців;
- об'єднання будинків в райони;
- аналіз соціальних графів для визначення груп друзів (в соцмережах);
- кластеризація фільмів за набором властивостей.

Зниження розмірності. Деякі складні задачі машинного навчання зазвичай використовують сотні або тисячі вхідних параметрів (ознак). Обробка такого обсягу даних потребує значних обчислювальних ресурсів і пам’яті та може збільшити час отриманя результату до неприйнятного. В такому випадку потрібно попередньо вирішити проблему зниження розмірності ознакового простору для опису об’єктів таким чином, щоб уникнути істотної втрати інформації. В Розділі 7 цього посібника було описано декілька методів зниження простору розмірності ознак, як один з них був детально описаний метод головних компонент (Principal Component Analysis або PCA). Основна його ідея геометрично ілюструється на рис. 8.6. Докладно цей метод описаний в Розділі 7 першої частини нашого посібника.

В наведеному на рис. 8.6 прикладі PCA знаходить спосіб трансформувати двовимірне подання даних в одновимірне. Замість двох

вхідних параметрів *x* і *y* він знаходить новий параметр *k*, який і є проекцією двовимірного простору в одновимірний.

Рисунок 8.6 – Приклад зниження розмірності простору ознак

З рис. 8.6 можна зробити висновок, що під час трансформації простору ознак відбувається деяка втрата даних. На лівому графіку видно, що точки не лежать точно на осі *k*, але на правому вони розміщуються безпосередньо на ній.

Метод PCA на практиці у випадку тисяч вхідних параметрів може скоротити їх кількість в 5-10 разів.

8.2.3 Навчання з підкріпленням

Навчання з підкріпленням (Reinforcement Learning, або RL) більшу частину часу працює з цілями безпосередньо штучного інтелекту – створенням агента, який зможе виробляти ефективні дії в заданому середовищі. Алгоритми RL використовують винагороду як зворотний зв'язок для виконаних дій і намагаються його максимізувати.

Популярність навчання з підкріпленням стала зростати після відомого матчу зі гри «Го» між системою штучного інтелекту AlphaGo, розробленої британською компанією Google DeepMind, і азіатським чемпіоном Лі Седолем. Система AlphaGo була створена з використанням алгоритмів RL. Навіть перша версія штучного інтелекту становила серйозний виклик будь-якій людині. Наступна версія – AlphaZero – дійшла до рівня складності, недосяжного для людей. Відмінна риса AlphaZero в тому, що вона навчилася грати сама з собою, а не використовувати людські партії для навчання.

Нині основні дослідження в навчанні з підкріпленням спрямовані на побудову штучного інтелекту для різних класичних відеоігор без опису правил гри.

Інакше кажучи, спочатку штучний інтелект нічого не знає про ігрове середовище, а знає лише кілька дій. Застосовуючи ці дії, він одержує відгук від гри і модифікує себе через механізм винагород/покарань.

Рисунок 8.7 – Схема машинного навчання з підкріпленням

Крім комп'ютерних ігор, навчання з підкріпленням дуже популярне для тренування роботів. Основна складність у застосуванні RL в робототехніці полягає в тому, що реальний світ дуже складно змоделювати з необхідною точністю. Внаслідок цього отриманий ШІ може ідеально виконувати завдання у віртуальному середовищі, але бути практично непридатним в продакшн-умовах.

В цьому розділі розглянуто основні особливості трьох напрямків машинного навчання: з учителем, без вчителя і з підкріпленням. У кожного з них є сфери практичного застосування в реальних умовах і свої відмінні риси.

8.2.4 Узагальнені висновки до характеристики методів машинного навчання

Навчання з учителем на сьогодні є найбільш розвиненим і застосовуваним різновидом машинного навчання. Воно реалізується на практиці в задачах класифікаційного або регресійного аналізу за наявності достатнього набору маркованих даних. На сьогодні також існують десятки готових класичних алгоритмів машинного, а також різні алгоритми глибинного навчання (Deep Learning) для вирішення більш складних завдань, таких як обробка зображення, тексту і голосу.

З іншого боку, машинне **навчання без учителя** набагато менше може бути застосовано в дійсності. У той час як асоціативні алгоритми допомагають в аналізі даних для роздрібних і онлайн-магазинів, кластеризація і зниження розмірності більше застосовуються як допоміжний інструмент для алгоритмів машинного навчання з учителем.

Нині проводиться багато досліджень щодо застосування нейронних мереж для розпізнавання складних паттернів в немаркованих даних. Потенційно вони можуть привести до прориву. Маючи в своєму

розпорядженні лише деякі довільні дані, алгоритми навчання без вчителя можуть бути здатними виявляти нетривіальні залежності або навіть, в деякому розумінні, складні закони.

Навчання з підкріпленням – дуже перспективний напрям для вирішення проблем, з якими може впоратися тільки людина. Зараз основні дослідження сконцентровані навколо навчання штучного інтелекту різних видів ігор. Основна перешкода для застосування RL на практиці – висока складність реального світу.

8.3 Відмінності у використанні традиційного програмування і машинного навчання для розв'язання інженерних задач

Для кращого розуміння того, як працює машинне навчання, розглянемо його відмінності від традиційного програмування.

Насамперед, машинне навчання не замінює традиційне програмування. Так, інженер-спеціаліст з обробки даних не стане створювати сайт за допомогою алгоритмів машинного навчання.

Зазвичай машинне навчання і штучний інтелект доповнюють стандартні інструменти програмування. Наприклад, для трейдингової системи алгоритм прогнозування може бути створений за допомогою машинного навчання, тоді як інтерфейс, візуалізація даних та інші складові будуть реалізовані звичною мовою програмування (Ruby, Python, Java тощо).

Основним правилом використання машинного навчання є його застосування там, де традиційні методи програмування не ефективні для вирішення задачі.

Для наочності розглянемо, як може бути вирішена класична задача прогнозування курсу обміну валют за допомогою обох технік – машинного навчання і традиційного програмування.

8.3.1 Розв'язання задачі за допомогою традиційного програмування

Етапи розв'язання інженером вказаної задачі **за допомогою традиційного програмування** подано на рис. 8.8.

У традиційному програмуванні, щоб отримати рішення, інженеру необхідно розробити алгоритм і написати код. Потім він задає вхідні параметри і за допомогою розробленого алгоритму отримує результат.

Для передбачення курсу валют алгоритм може використовувати багато різних вхідних параметрів:

- ✓ вчорашній курс;
- ✓ вчорашні значення обмінних курсів інших валют;
- ✓ економічні зміни в країні, яка випускає цю валюту;
- ✓ зміни в світовій економіці та ін.

Рисунок 8.8 – Етапи знаходження рішення за допомогою традиційного програмування.

Таким чином, за допомогою традиційного програмування ми самі створюємо рішення, яке може прийняти набір параметрів і на підставі вхідних даних передбачити новий курс обміну валют.

Основна проблема полягає в тому, що людині вкрай складно працювати з великим обсягом параметрів, тоді як з обмеженим набором можна побудувати тільки дуже просту модель.

8.3.2 Розв'язання задачі за допомогою машинного навчання

Для вирішення того самого завдання методами машинного навчання інженер застосовує зовсім інший підхід. Він не розробляє алгоритм вирішення задачі самостійно, а спочатку збирає масив історичних (апріорних даних, після чого використовує його для напівавтоматичної побудови моделі машинного навчання (рис. 8.39).

Зібравши представницьку навчальну вибірку даних, інженер завантажує її в різні алгоритми машинного навчання. Результатом цього етапу є навчальна модель (МН), яка уже може створювати нові прогнози за подання на вхід нових даних.

Інженер може використовувати різні «регулятори», щоб налагодити алгоритм навчання і отримати різні моделі. Модель, яка видає найкращий результат, відбирається для подальшого застосування.

Використання готової моделі в такому випадку подібне до процедури, що має місце в традиційному програмуванні, тільки замість алгоритму використовується розроблена модель. Модель отримує вхідні

дані і виробляє результат. Весь процес виглядає так, як показано на рисунку 8.9.

Рисунок 8.9 – Послідовність вирішення задачі методом машинного навчання

Основна відмінність між традиційним програмуванням і машинним навчанням в тому, що в машинному навчанні нам не потрібно будувати модель самостійно. Це завдання виконують алгоритми машинного навчання, з хіба що невеликими правками, які дата-інженер вносить в налаштування алгоритму.

Інша важлива відмінність в кількості вхідних параметрів, які модель здатна обробити. Для коректного прогнозу погоди в тій чи іншій локації, теоретично знадобиться ввести тисячі параметрів, які вплинуть на результат. Людина апріорі не може побудувати алгоритм, який буде використовувати всі з них в розумний спосіб. Для машинного навчання таких обмежень не існує. Поки вистачає потужності процесора і пам'яті, можна використовувати стільки вхідних параметрів, скільки вважається потрібним.

8.4 Огляд кращих бібліотек для роботи з машинним навчанням (ML) і глибоким навчанням (DL)

8.4.1 Загальна характеристика мови Python

Сьогодні існує величезна кількість бібліотек для машинного і глибокого навчання. Для полегшення їх вибору розглянемо тільки найпопулярніші і необхідні бібліотеки, які покривають всі базові потреби для початку роботи з ML і DL. Найбільш популярними мовами програмування, які використовують для реалізації алгоритмів машинного навчання, вважаються мови Python і R.

Мова R створювалась для вирішення статистичних задач і є дуже популярною в сереовищі дата-аналітиків. Основною проблемою в її використанні є труднощі застосування для вирішення задач, не пов'язаних з аналізом і візуалізацією даних.

Зі свого боку Python – це мова загального призначення і може бути успішно застосована для вирішення різних задач. Завдяки цьому в останні декілька років Python набрав значної популярності і став основною робочою мовою в співтоваристві машинного навчання. Практично будь-яка сучасна ML або DL бібліотека надає програмний інтерфейс застосунку мовою Python – Python API.

Тому нині вибір мови для реалізацій задач машинного навчання припадає на використання мови Python. Вона є досить нескладною і легкою у вивченні, а для застосування цієї мови в машинному навчанні не потрібно знати всі її тонкощі.

8.4.2 Основні бібліотеки мови Python

Jupyter Notebook: робота з даними, кодом і графіками

Jupiter Notebook – це простий і потужний інструмент для аналізу даних. На відміну від традиційного програмування, під час якого більша частина часу витрачається на роботу в текстових редакторах або програмних середовищах (IDE), в Data Science велика частина коду може бути написана в Jupyter Notebook.

Він дозволяє писати код на Python, R та іншими мовами, додавати текстові описи в Markdown, вбудовувати графіки та діаграми безпосередньо в інтерактивну веб-сторінку. На додаток до цього Google випустив безкоштовний сервіс Google Colab, який надає хмарну версію Jupyter Notebook і дає можливість реалізувати обчислення на CPU і GPU. Всі потрібні ML бібліотеки мовою Python вже встановлені, так що можна починати відразу там, не встановлючи все локально.

Scikit-learn: найкраща бібліотека для класичних ML алгоритмів

Scikit-learn – одна з найпопулярніших ML бібліотек на сьогодні. Вона підтримує більшість алгоритмів навчання, як з учителем, так і без: лінійну і логістичну регресії, метод опорних векторів (SVM), Байєсівську класифікацію, градієнтний бустінг, кластеризацію методами найближчого сусіда (KNN) і *k*- внутрішньогрупових середніх (*k*-mains) та багато інших.

Крім цього, Scikit-learn містить безліч корисних утиліт для підготовки даних і аналізу результатів. Ця бібліотека призначена переважно для класичних алгоритмів машинного навчання, тому її функціонал для нейронних мереж дуже обмежений, а для задач глибокого навчання вона не може бути використана зовсім.

На додаток до дуже якісної документації, Scikit-learn містить розділ з керівництвами, в якому показано, як працювати з бібліотекою, а також даються базові знання з машинного навчання.

Pandas: витяг і підготовка даних

Аналіз і підготовка даних часто займає більшу частину часу в процесі вирішення завдань. Дані можуть бути отримані в CSV, JSON, Excel або іншому структурованому форматі, і їх потрібно обробити для того, щоб використовувати в ML моделях.

Для ефективної реалізації цього завдання зручно використати бібліотеку Pandas. Застосування цього потужного інструменту дозволяє швидко аналізувати, модифікувати і готувати дані для подальшого використання в інших ML і DL бібліотеках, таких як **Scikit-learn, TensorFlow** або **PyTorch**.

У Pandas можна завантажувати дані з різних джерел: SQL баз, CSV, Excel, JSON файлів і інших менш популярних форматів.

Коли дані завантажено в пам'ять, з ними можна виконувати безліч різних операцій для аналізу, трансформації, заповнення відсутніх значень і очищення набору даних. Pandas дозволяє виконувати багато SQL-подібних операцій над наборами даних: об'єднання, групування, агрегування тощо. Також вона надає вбудований набір популярних статистичних функцій для базового аналізу.

Pandas також підтримується в Jupyter Notebook, що дозволяє реалізувати якісну візуалізацію його структур даних.

Сайт Pandas містить дуже докладну документацію. Її вивчення найкраще почати з 10-хвилинного керівництва (tutorial), яке показує всі основні особливості і можливості цієї бібліотеки.

Бібліотека NumPy: багатовимірні масиви і лінійна алгебра

Основний функціонал NumPy полягає в підтримці багатовимірних масивів даних і швидких алгоритмів лінійної алгебри. Саме тому NumPy є також ключовим компонентом для реалізації утиліт бібліотек Scikit-learn, SciPy і Pandas.

Зазвичай NumPy використовують як допоміжну бібліотеку для виконання різних математичних операцій зі структурами даних Pandas, тому варто вивчити її базові можливості. Для цього зручно використати вступний tutorial до Numpy, а також основи NumPy.

Matplotlib і **Seaborn: побудова графіків і візуалізація даних**

Matplotlib є стандартним інструментом в наборі інженера з Data Science. Він дозволяє створювати різноманітні графіки та діаграми для візуалізації отриманих результатів.

Графіки, створені в Matplotlib, легко інтегруються в Jupyter Notebook і Python. Це дає можливість візуалізувати дані і результати, отримані в процесі обробки моделей. Для цієї бібліотеки створено багато додаткових пакетів. Одним з найпопулярніших є **Seaborn**. Його основна перевага – це готовий набір найбільш часто використовуваних статистичних діаграм і графіків.

Традиційно ці дві бібліотеки мають розділ з туторіалами на їх сайтах, але більш ефективним підходом до їх вивчення є реєстрація на сайті

Kaggle і ознайомлення в розділі «Kernels» з готовими прикладами використання, наприклад, **Comprehensive Data Exploration** with **Python**.

Tensorflow і Keras: бібліотеки глибокого навчання

Будь-яка бібліотека глибокого навчання містить три ключові компоненти: багатовимірні масиви (вони ж тензори), оператори лінійної алгебри та обчислення похідних.

У Tensorflow, бібліотеці глибокого навчання від Google, відмінно реалізовано всі три компоненти. Поряд з CPU, вона підтримує обчислення на GPU і TPU (тензорних процесорах Google).

Нині це найпопулярніша бібліотека глибокого навчання, внаслідок чого створено безліч керівництв і онлайн-курсів щодо роботи з нею. Але ця довершеність має і породжені нею досить коструватий API та більш високий поріг входу порівняно з тією ж PyTorch.

Надбудовою над цією бібліотекою є **Keras** – бібліотека, яка вирішує більшість проблем застосування Tensorflow. Її головна перевага – це можливість будувати архітектуру нейронної мережі з використанням Python DSL. Для Keras також написано багато навчальних матеріалів, тому вивчити правила роботи з нею нескладно.

PyTorch: альтернативна бібліотека глибокого навчання

PyTorch – це друга за популярністю DL бібліотека після Tensorflow, яка створена в Facebook. Її сильна сторона полягає в тому, що вона була розроблена для Python, і тому використовує його стандартні вирази. Порівняно з Tensorflow, тут складність вивчення набагато нижча, а будь-яку нейронну мережу можна побудувати з використанням стандартних класів ООП і об'єктів.

Також її легше налагоджувати, оскільки код виконується як звичайний Python код – немає етапу компіляції, як в TensorFlow. Тому можна користуватися навіть налагоджувачем мови Python.

У PyTorch теж є своя надбудова – це бібліотека **Fastai**. Вона дозволяє вирішити більшість стандартних DL завдань за допомогою декількох рядків коду. Привабливою особливістю для вивчення Fastai є наявність на її сайті онлайн-курсу Practical Deep Learning for Codersnull.

На що звернути увагу під час вивчення машинного навчання

Щоб зробити процес навчання більш гладким, є сенс почати експерименти з класичних задач ML та сфокусуватися на використанні Scikit-learn і Pandas. І після цього можна рухатися в бік глибокого навчання. Під час вибору для вивчення DL бібліотеки з TensorFlow/Keras або PyTorch можна орієнтуватися на наявність такого онлайн-курсу, який вам подобається. Після ознайомлення з основними інструментами і хоча б базового розуміння, що вони дозволяють робити, наступний важливий крок – вибір навчальних матеріалів. Курсів з вивчення ML і DL існує величезна безліч і часу пройти всі просто не вистачить.

Контрольні питання та завдання

1. Що розуміють під терміном «машинне навчання»?
2. На яких наукових дисциплінах базуються методи машинного навчання?
3. Назвіь напрямки застосування методів машинного навчання.
4. Які два основних типи навчання розрізняють?
5. Назвіть відмінності між навчанням на прецедентах і дедуктивним навчанням.
6. Сформулюйте загальну постановку задачі машинного навчання.
7. На які основні чотири напрямки поділяють методи машинного навчання?
8. Наведіть схему класифікації методів машинного навчання.
9. В чому полягає різниця між розв'язанням задачі за допомогою традиційного програмування і методами машинного навчання?
10. Опишіть метод машинного навчання з учителем (контрольоване навчання).
11. Які типи задач можна вирішувати методом контрольованого навчання?
12. В чому полягає суть задачі класифікації?
13. Що являє собою задача регресійного аналізу?
14. Опишіть метод машинного навчання без учителя (неконтрольоване навчання).
15. Які категорії алгоритмів розрізняють у машинному навчанні без учителя?
16. Як працюють асоціативні алгоритми машинного навчання?
17. В чому полягає задача кластеризації даних?
18. Що таке зниження розмірності даних?
19. Опишіть принцип зниження розмірності даних методом PCA.
20. Що таке «навчання з підкріпленнями» і в яких завданнях воно використовується?
21. Які мови програмування використовуються для реалізації методів машинного навчання?
22. Які основні переваги мови Python в програмуванні задач машинного навчання?
23. Які можливості для роботи з даними надає бібліотека Jupiter Notebook?
24. В яких задачах використовуються утиліти бібліотеки Scikit-learn?
25. Яку бібліотеку ефективно використовувати для аналізу і підготовки даних?
26. Яка бібліотека використовується для обробки багатовимірних масивів даних і швидких алгоритмів лінійної алгебри?
27. В яких випадках використовується бібліотека Matplotlib?

РОЗДІЛ 9
АЛГОРИТМИ НАВЧАННЯ КЛАСИФІКАТОРІВ

9.1 Математична постановка задачі навчання

Навчанням системи розпізнавання образів в загальному випадку називається надання їй уміння відносити подані на її вхід зображення об'єктів до відповідних класів. В процесі навчання система, шляхом аналізу навчальної вибірки зображень, має вибрати множину інформативних ознак $\{x_1, x_2, x_3, ..., x_n\}$, визначити на основі вибраного критерію подібності множину класів для розпізнавання $\mathbf{\Omega} = \{\Omega_1, \Omega_2 .... \Omega_K\}$, і визначити вигляд вирішувальних функцій для віднесення нових зображень до одного з K класів. В тому випадку, коли система розпізнавання розв'язує вказані задачі самостійно, то така процедура називається навчанням «без учителя» або неконтрольованим навчанням.

Головна особливість контрольованого методу навчання (навчання «з учителем») полягає в обов'язковій наявності «апріорних» знань про належність до певного класу кожного вектора зображень, що входить в навчальну вибірку, тобто вибіркові зображення мають бути марковані мітками класів. За цих умов розв'язування задач навчання полягає в уточненні і оптимізації процедури прийняття рішень про належність зображення до одного з відомих класів на основі вирішувальних функцій.

За приклад навчання «без учителя» можна вважати розглянуті в Розділі 5 алгоритми кластеризації, хоча їх можна вважати такими тільки умовно, оскільки вказані алгоритми містять евристичні правила, знайдені і привнесені в них людиною. Більш-менш відповідним терміну «навчання без учителя» є алгоритм кластеризації з використанням потенціальних функцій, оскільки він використовує природно закладену в сукупність близьких точок міру їх купності у вигляді сумарного потенціалу.

Контрольовані алгоритми навчання можна поділити на детерміновані і статистичні. В цьому розділі будуть розглянуті детерміновані алгоритми, які конструюються незалежно від будь-яких передбачень про статистичні властивості образів.

В Розділі 3 було показано, що розв'язування задачі про розподіл на два класи еквівалентно розв'язанню системи лінійних нерівностей:

$$\vec{w}^T \vec{x} > 0, \tag{9.1}$$

$$\vec{w}^T \vec{x} < 0, \tag{9.2}$$

де ліві частини нерівностей (9.1) і (9.2) являють собою або лінійні, або узагальнені вирішувальні функції.

Якщо помножити нерівність (9.2) на -1, то задача навчання розпізнавання класів буде зведена до знаходження такого вектора ваг $\vec{w}$, який буде задовольняти умову

$$\mathbf{X}\vec{w} > \vec{0}, \tag{9.3}$$

$$\mathbf{X} = \begin{bmatrix} \vec{x}_1^T \\ \vec{x}_2^T \\ \dots \\ \vec{x}_N^T \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} \dots x_{1n} & 1 \\ x_{21} & x_{22} \dots x_{2n} & 1 \\ \dots & \dots & 1 \\ x_{N1} & x_{N2} \dots x_{Nn} & 1 \end{bmatrix}, \tag{9.4}$$

де N – кількість вибіркових векторів – об'єктів в n – вимірному просторі ознак,

$\mathbf{X}$ – матриця координат векторів $\vec{x}_i$ розміром $N \times (n+1)$;

$\vec{w}^T = (w_1, w_2, \dots, w_n, w_{n+1})$ – поповнений ваговий вектор;

$\vec{0}$ – нульовий вектор;

$\vec{x}_i^T$ – транспоновані поповнені вектори зображень.

Якщо вектор ваг $\vec{w}$, що задовольняє умову (9.3) існує, то нерівності називаються сумісними, і класи відповідно розділимі в просторі ознак. В іншому випадку нерівності несумісні і класи відповідно є нероздільними.

9.2 Перцептронний підхід до навчання

Першою моделлю класифікатора образів був перцептрон (рис. 9.1). Перцептрон складається із сітки S сенсорних елементів, які випадковим чином з'єднані з асоціативними елементами другої (аналітичної) сітки А.

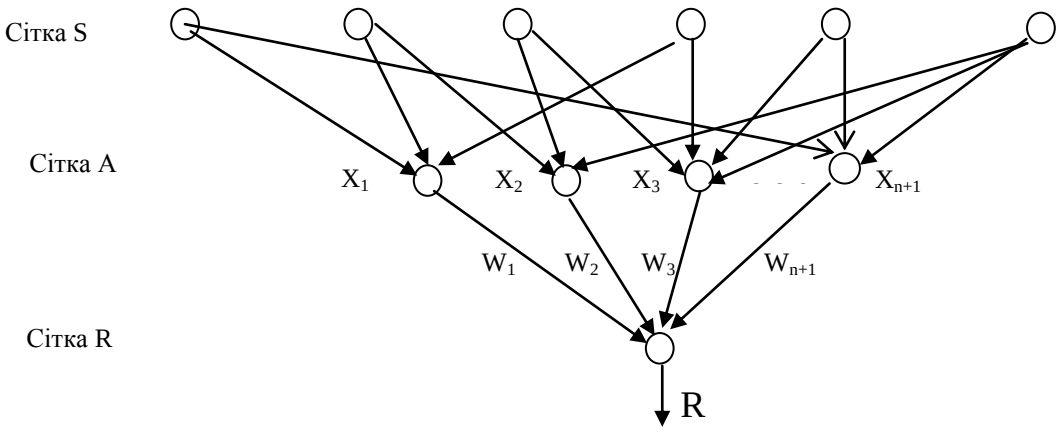


Рисунок 9.1 – Основний варіант моделі перцептрона

Кожен з елементів другої сітки виробляє вихідний сигнал тільки в тому випадку, коли достатня кількість сенсорних елементів, з'єднаних з його входом, знаходиться в збудженому стані. Сенсорні елементи можна вважати за деякі вимірювальні перетворювачі, які сприймають з зовнішнього середовища стимули, а асоціативні елементи – як вхідну частину системи.

Реакція всієї системи пропорціональна сумі взятих з деякими вагами елементів асоціативної сітки

$$R = \sum_{i=1}^{n+1} w_i x_i = \vec{w}^T \vec{x}\,, \tag{9.5}$$

де x_i – реакція *i*-того асоціативного елемента,

w_i – вага зв'язку між цим елементом і виходом.

Якщо $R > 0$, то зображення, подане на вхід системи, належить класу до Ω_1, якщо $R < 0$, то зображення належить класу до Ω_2.

Алгоритм навчання перцептрона зводиться до ітераційного визначення вектора ваг $\vec{w}$ за принципом «заохочення – покарання». Наведемо короткий його опис.

Нехай задано дві навчальних множини векторів, які зображають в просторі ознак об'єкти першого Ω_1 і другого Ω_2 образів, відповідно, і вибрано довільно початкове значення вектора ваг $\vec{w}(1)$.

В цьому випадку *k*-ий крок навчання виглядає таким чином.

Якщо $\vec{x}(k) \in \Omega_1$ і $\vec{w}^T(k) \cdot \vec{x}(k) \le 0,$ то вектор ваг $\vec{w}(k)$ замінюється вектором

$$\vec{w}(k+1) = \vec{w}(k) + c\vec{x}(k), \tag{9.6}$$

де c – приріст корекції.

Якщо $\vec{x}(k) \in \Omega_2$ і $\vec{w}^T(k) \cdot \vec{x}(k) \ge 0,$ то вектор $\vec{w}(k)$ замінюється вектором

$$\vec{w}(k+1) = \vec{w}(k) - c \cdot \vec{x}(k). \tag{9.7}$$

В іншому випадку вектор $\vec{w}(k)$ не змінюється, тобто

$$\vec{w}(k+1) = \vec{w}(k)\,. \tag{9.8}$$

Інакше кажучи, алгоритм вносить корекцію в вектор ваг $\vec{w}$ тільки в тому випадку, коли подане на *k*-ому кроці навчання зображення було неправильно класифіковано вектором $\vec{w}(k)$. Приріст корекції c має бути додатним і може мати сталу або змінну величину.

З наведеного прикладу видно, що в випадку правильної класифікації в вектор ваг $\vec{w}$ ніякі зміни не вносяться. Якщо ж образ класифікований

неправильно і добуток $\vec{w}^T(k)\cdot\vec{x}(k)$ є меншим нуля, система розпізнавання «карається» збільшенням значення вектора ваг $\vec{w}(k)$ на величину, пропорційну $\vec{x}(k)$. Таким самим чином, якщо добуток $\vec{w}^T(k)\cdot\vec{x}(k)$ є більшим нуля, коли він має бути меншим нуля, система «карається» зменшенням вектора ваг.

Приклад 9.1. Визначити ваговий вектор віршувальної функції за методом перцепрона для класифікації образів (рис 9.2).

Розв'язування. Спочатку доповнимо всі вектори зображень в класах Ω_1 і Ω_2, отримаємо $\vec{x}_1=(0,0,1)^T$, $\vec{x}_2=(0,1,1)^T$, $\vec{x}_3=(1,0,1)^T$, $\vec{x}_4=(1,1,1)^T$. Візьмемо $c=1$ і $\vec{w}(1)=0$, і подамо вектори зображень на вхід перцептронного алгоритму у вказаному порядку для корекції вектора $\vec{w}$:

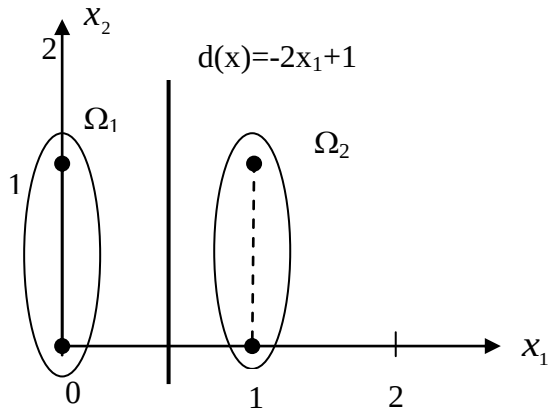


Рисунок 9.2 – Навчальна вибірка векторів для визначення вагового вектора вирішувальної функції за алгоритмом перцептрона

$$\vec{w}^T(1)\cdot\vec{x}_1(1)=(0,0,0)\cdot\begin{pmatrix}0\\0\\1\end{pmatrix}=0,$$

$$\vec{w}(2)=\vec{w}(1)+\vec{x}_1(1)=\begin{pmatrix}0&0\\0+0\\0&1\end{pmatrix}=\begin{pmatrix}0\\0\\1\end{pmatrix},$$

$$\vec{w}^T(2)\cdot\vec{x}_2(2)=(0,0,1)\cdot\begin{pmatrix}0\\1\\1\end{pmatrix}=1,$$

$$\vec{w}(3)=\vec{w}(2)=(0,0,1)^T$$

$$\vec{w}^T(3)\vec{x}_3(3)=(0,0,1)\cdot\begin{pmatrix}1\\0\\1\end{pmatrix}=1,$$

$$\vec{w}(4) = \vec{w}(3) - \vec{x}(3) = \begin{pmatrix} 0-1 \\ 0-0 \\ 1-1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 0 \end{pmatrix},$$

$$\vec{w}^T(4) \cdot \vec{x}_4(4) = (-1,0,0) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = -1,$$

$$\vec{w}(5) = \vec{w}(4) = (-1,0,0)^T.$$

Корекція вектора ваг здійснювалась на першому і третьому кроках згідно з формулами (9.6) і (9.7) відповідно до зроблених помилок класифікації. Оскільки алгоритм у циклі подання всіх зображень припустився двох помилок, то навчальну вибірку потрібно подати на вхід класифікатора ще раз:

$$\vec{w}^T(5) \cdot \vec{x}_1(5) = (-1,0,0) \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = 0,$$

$$\vec{w}(6) = \vec{w}(5) + \vec{x}_1(5) = \begin{pmatrix} -1+0 \\ 0+0 \\ 0+1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix},$$

$$\vec{w}^T(6) \cdot \vec{x}_2(6) = (-1,0,1) \cdot \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix} = 1,$$

$$\vec{w}\ (7) = \vec{w}(6) = (-1,0,1)^T,$$

$$\vec{w}^T(7) \cdot \vec{x}_3(7) = (-1,0,1) \cdot \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix} = 0,$$

$$\vec{w}(8) = \vec{w}(7) - \vec{x}(7) = \begin{pmatrix} -1-1 \\ 0-0 \\ 1-1 \end{pmatrix} = \begin{pmatrix} -2 \\ 0 \\ 0 \end{pmatrix},$$

$$\vec{w}^T(8) \cdot \vec{x}_4(8) = (-2,0,0) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = -2 < 0,$$

отже $\vec{w}^T(9) = \vec{w}^T(8) = (-2,0,0)$.

Оскільки в цьому ітераційному циклі зроблено дві помилки, виконаємо ще один цикл подання образів:

$$\vec{w}^T(9)\cdot\vec{x}_1(9)=(-2,0,0)\cdot\begin{pmatrix}0\\0\\1\end{pmatrix}=0,$$

$$\vec{w}(10)=\vec{w}(9)+\vec{x}_1(9)=\begin{pmatrix}-2+0\\0\ +0\\0\ +1\end{pmatrix}=\begin{pmatrix}-2\\0\\1\end{pmatrix},$$

$$\vec{w}^T(10)\cdot\vec{x}_2(10)=(-2,0,1)\cdot\begin{pmatrix}0\\1\\1\end{pmatrix}=1,$$

$$\vec{w}(11)=\vec{w}(10)=(-2,0,1)^T,$$

$$\vec{w}^T(11)\cdot\vec{x}_3(11)=(-2,0,1)\cdot\begin{pmatrix}1\\0\\1\end{pmatrix}=-1,$$

$$\vec{w}(12)=\vec{w}(11)=(-2,0,1)^T,$$

$$\vec{w}^T(12)\cdot\vec{x}_4(12)=(-2,0,1)\cdot\begin{pmatrix}1\\1\\1\end{pmatrix}=-1,$$

$$\vec{w}(13)=\vec{w}(12)=(-2,0,1)^T.$$

Оскільки в циклі зроблено одну помилку, то виконуємо ще одну ітерацію, і впевнюємося в тому, що всі зображення класифікуються за допомогою отриманого вагового вектора правильно. Таким чином, вектор рішення має вигляд $\vec{w}=(-2,0,1)^T$ і йому відповідає вирішувальна функція $d(\vec{x})=-2x_1+1$. Відповідна їй розподільна границя зображена на рис. 9.2.

Згідно з аналізом, проведеним в розділі 9.1, алгоритм перцептрона можна подати в іншій еквівалентній формі, якщо помножити всі зображення класу Ω_2 на -1:

$$\vec{w}(k+1)=\begin{cases}\vec{w}(k), & якщо\ \ \vec{w}^T(k)\vec{x}(k)>0\\ \vec{w}^T+c\vec{x}(k), & якщо\ \ \vec{w}(k)\vec{x}(k)\le 0\end{cases}, \qquad (9.9)$$

де c – додатний приріст корекції.

Доведено в роботах різних авторів, що у випадку лінійної роздільності класів алгоритм є збіжним.

Змінюючи спосіб вибору приросту корекції c, можна отримати декілька модифікацій алгоритму перцептрона. До них належать алгоритм

фіксованого приросту, алгоритм корекції абсолютної величини і алгоритм дробової корекції.

Приклад застосування алгоритму навчання з фіксованим приростом розглянуто раніше, коли було вибрано приріст у вигляді константи $c=1$.

В алгоритмі корекції абсолютної величини приріст c вибирається достатньо великим, щоб гарантувати правильну класифікацію вектора після корекції ваг. Іншими словами, якщо $\vec{w}^T(k)\vec{x}(k) \le 0$, то коефіцієнт c вибирається таким чином,

$$\vec{w}^T(k+1)\vec{x}(k) = \left[\vec{w}(k) + c\vec{x}(k)\right]^T \vec{x}(k) > 0. \qquad (9.10)$$

Виходячи з (9.10), приріст c має вибиратися з умови

$$c = \left\lceil \frac{\left|\vec{w}^T(k)\vec{x}(k)\right|}{\vec{x}^T(k)\vec{x}(k)} \right\rceil, \qquad (9.11)$$

де вираз $\lceil a \rceil$ означає найменше ціле, більше за a;

$|a|$ – означає модуль a.

В алгоритмі дробової корекції c вибирається таким чином, щоб величина $\left|\vec{w}^T(k)\vec{x}(k) - \vec{w}^T(k+1)\vec{x}(k)\right|$ була додатною і становила деяку частину λ від величини $\left|\ \vec{w}^T(k)\vec{x}(k)\ \right|$, тобто

$$\left|\vec{w}^T(k)\vec{x}(k) - \vec{w}^T(k+1)\vec{x}(k)\right| = \lambda\left|\vec{w}^T(k)\vec{x}(k)\right|. \qquad (9.12)$$

Підставимо в вираз (9.12) значення $\vec{w}(k+1) = \vec{w}(k) + c\vec{x}(k)$, отримаємо

$$c = \lambda \frac{\left|\vec{w}^T(k)\vec{x}(k)\right|}{\vec{x}^T(k)\vec{x}(k)}. \qquad (9.13)$$

Алгоритм дробової корекції потребує, щоб початковий вектор ваг відрізнявся від 0.

Якщо $\lambda > 1$, то образ класифікується правильно після кожної корекції ваг. Можна показати, що за $0 < \lambda < 2$ цей алгоритм збіжний.

Алгоритм навчання перцептрона можна застосувати і для випадку, коли є більше двох класів. Розглянемо перцептронний алгоритм, який можна застосувати для 3-го випадку класифікації (Розділ 3), в якому для M класів існує M вирішувальних функцій таких, що

$$\vec{x} \in \Omega_i, \text{ якщо } d_i(\vec{x}) > d_j(\vec{x}) \text{ для всіх } j \ne i, \qquad (9.14)$$

де $j = 1,\ 2,\ ...,M$.

Нехай на k-ому ітераційному кроці класифікатору подається вектор зображення $\vec{x}(k)$, який належить образу Ω_i. Обчислюють всі M вирішувальних функцій

$$d_j(\vec{x}(k)) = \vec{w}_j^T(k)\vec{x}(k).$$

Якщо виконується умова (9.14), то вектори ваг не змінюються , тобто

$$\vec{w}_j(k+1) = \vec{w}_j(k),\ j = 1,2,...,M. \qquad (9.15)$$

Нехай, з іншого боку, для деякого l

$$d_i(\vec{x}(k)) \le d_l(\vec{x}(k)),$$

тоді здійснюється корекція ваг таким чином:

$$\begin{aligned} \vec{w}_i(k+1) &= \vec{w}_i(k) + c\vec{x}(k), \\ \vec{w}_l(k+1) &= \vec{w}_l(k) - c\vec{x}(k), \\ \vec{w}_j(k+1) &= \vec{w}_j(k),\ j = 1,2,...,M;\ j \ne i,\ \ j \ne l, \end{aligned} \qquad (9.16)$$

де c – додатна константа.

Якщо для випадку 3 класи є роздільними, то можна показати, що цей алгоритм збігається за скінченну кількість ітерацій за довільних початкових значень векторів ваг $\vec{w}_j(1)$, $j = 1,2,...,M$. Наведемо приклад застосування алгоритму.

Приклад 9.2. Нехай задано класи, кожен з яких містить по одному об'єкту: $\Omega_1 : \{\vec{x}_1 = (0,0)^T\}$; $\Omega_2 : \{\vec{x}_2 = (1,1)^T\}$; $\Omega_3 : \{\vec{x}_3 = (-1,1)^T\}$;

Побудувати вирішувальні функції для цих класів $d_1(\vec{x}) = \vec{w}_1^T\vec{x}$; $d_2(\vec{x}) = \vec{w}_2^T\vec{x}$; $d_3(\vec{x}) = \vec{w}_3^T\vec{x}$, визначивши шляхом навчання вектори ваг $\vec{w}_1, \vec{w}_2, \vec{w}_3$ за допомогою алгоритму перцептрона.

Розв'язування. Спочатку поповнимо вектори зображень: $\vec{x}_1 = (0,0,1)^T; \vec{x}_2 = (1,1,1)^T; \vec{x}_3 = (-1,1,1)^T$.

За початкові значення векторів ваг виберемо $\vec{w}_1(1) = \vec{w}_2(1) = \vec{w}_3(1) = (0,0,0)^T$, покладемо $c = 1$ і подамо вектори зображень в записаній послідовності:

$$\begin{aligned} d_1[\vec{x}_1(1)] &= \vec{w}_1^T(1)\vec{x}_1(1) = 0, \\ d_2[\vec{x}_1(1)] &= \vec{w}_2^T(1)\vec{x}_1(1) = 0, \\ d_3[\vec{x}_1(1)] &= \vec{w}_3^T(1)\vec{x}_1(1) = 0. \end{aligned}$$

Оскільки $\vec{x}_1(1) \in \Omega_1$, то перший ваговий вектор збільшується, а два інших зменшуються згідно з (9.16), тобто

$$\vec{w}_1(2) = \vec{w}_1(1) + \vec{x}_1(1) = \begin{pmatrix} 0+0 \\ 0+0 \\ 0+1 \end{pmatrix} = (0,0,1)^T;$$

$$\vec{w}_2(2) = \vec{w}_2(1) - \vec{x}_1(1) = \begin{pmatrix} 0-0 \\ 0-0 \\ 0-1 \end{pmatrix} = (0,0,-1)^T;$$

$$\vec{w}_3(2) = \vec{w}_3(1) - \vec{x}_1(1) = (0,0,-1)^T.$$

Наступний поданий вектор $\vec{x}_2(2) = (1,1,1)^T$ належить класу Ω_2; для нього одержуємо:

$$d_1[\vec{x}_2(2)] = \vec{w}_1^T(2)\vec{x}_2(2) = (0,0,1) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = 1,$$

$$d_2[\vec{x}_2(2)] = \vec{w}_2^T(2)\vec{x}_2(2) = (0,0,-1) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = -1,$$

$$d_3[\vec{x}_2(2)] = \vec{w}_3^T(2)\vec{x}_2(2) = (0,0,-1) \cdot \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = -1.$$

Оскільки всі добутки більші або дорівнюють $\vec{w}_2^T\vec{x}_2(2)$, то вводимо корекцію

$$\vec{w}_1(3) = \vec{w}_1(2) - \vec{x}_2(2) = (-1,-1,0)^T,$$
$$\vec{w}_2(3) = \vec{w}_2(2) + \vec{x}_2(2) = (1,1,0)^T,$$
$$\vec{w}_3(3) = \vec{w}_3(2) - \vec{x}_2(2) = (-1,-1,-2)^T.$$

Наступний вектор $\vec{x}_3(3) = (-1,1,1)^T$, що подається на вхід класифікатора, належить класу Ω_3; для нього маємо такі значення вирішувальних функцій:

$$\vec{w}_1^T(3)\vec{x}_3(3) = 0,$$
$$\vec{w}_2^T(3)\vec{x}_3(3) = 0,$$
$$\vec{w}_3^T(3)\vec{x}_3(3) = -2.$$

Всі ці значення свідчать про неправильну класифікацію, тому вводимо корекцію

$$\vec{w}_1(4) = \vec{w}_1(3) - \vec{x}_3(3) = (0,-2,-1)^T,$$
$$\vec{w}_2(4) = \vec{w}_2(3) - \vec{x}_3(3) = (2,0,-1)^T,$$
$$\vec{w}_3(4) = \vec{w}_3(3) + \vec{x}_3(3) = (-2,0,-1)^T.$$

Оскільки в цьому ітераційному циклі були помилки, проводимо новий цикл навчання:

$$\vec{w}_1^T(4)\vec{x}_1(4) = -1,$$
$$\vec{w}_2^T(4)\vec{x}_1(4) = -1,$$
$$\vec{w}_3^T(4)\vec{x}_1(4) = -1.$$

Значення всіх добутків неправильне, оскільки $\vec{x}_1 \in \Omega_1$, тому коректуємо вектори ваг:

$$\vec{w}_1(5) = \vec{w}_1(4) + \vec{x}_1(4) = (0,-2,0)^T,$$
$$\vec{w}_2(5) = \vec{w}_2(4) - \vec{x}_1(4) = (2,0,-2)^T,$$
$$\vec{w}_3(5) = \vec{w}_3(4) - \vec{x}_1(4) = (-2,0,-2)^T.$$

На наступному кроці подаємо образ $\vec{x}_2(5) = (1,1,1)^T$, отримуємо:

$$\vec{w}_1^T(5)\vec{x}_2(5) = -2,$$
$$\vec{w}_2^T(5)\vec{x}_2(5) = 0,$$
$$\vec{w}_3^T(5)\vec{x}_2(5) = -4.$$

Оскільки $d_2[\vec{x}_2(5)] > d_1[\vec{x}_2(5)]$ і $d_2[\vec{x}_2(5)] > d_3[\vec{x}_2(5)]$, а $\vec{x}_2 \in \Omega_2$, то вектор $\vec{x}_2$ в цьому випадку класифікований правильно. Тому вектори ваг залишаємо незмінними:

$$\vec{w}_1(6) = \vec{w}_1(5) = (0,-2,0),$$
$$\vec{w}_2(6) = \vec{w}_2(5) = (2,0,-2),$$
$$\vec{w}_3(6) = \vec{w}_3(5) = (-2,0,-2).$$

Для вектора $\vec{x}_3(6)=(-1,1,1)^T$, $\vec{x}_3 \in \Omega_3$, маємо такі результати класифікації:

$$\vec{w}_1^T(6)\vec{x}_3(6) = -2,$$
$$\vec{w}_2^T(6)\vec{x}_3(6) = -4,$$
$$\vec{w}_3^T(6)\vec{x}_3(6) = 0.$$

Цей вектор зображень класифікований також правильно, тому вектори ваг не коригуються:

$$\vec{w}_1(7) = \vec{w}_1(6) = (0,-2,0)^T,$$
$$\vec{w}_2(7) = \vec{w}_2(6) = (2,0,-2)^T,$$
$$\vec{w}_3(7) = \vec{w}_3(6) = (-2,0,-2)^T.$$

Оскільки в цьому циклі навчання було допущено помилки класифікації, виконаємо ще один цикл навчання. Для вектора $\vec{x}_1 = (0,0,1)^T$ маємо:

$$\vec{w}_1^T(7)\vec{x}_1(7) = 0,$$
$$\vec{w}_2^T(7)\vec{x}_1(7) = -2,$$
$$\vec{w}_3^T(7)\vec{x}_1(7) = -2.$$

Класифікацію виконано правильно, тому:

$$\vec{w}_1(8) = \vec{w}_1(7) = (0,-2,0)^T,$$
$$\vec{w}_2(8) = \vec{w}_2(7) = (2,0,-2)^T,$$
$$\vec{w}_3(8) = \vec{w}_3(7) = (-2,0,-2)^T.$$

Для вектора $\vec{x}_2(8) = (1,1,1)$ маємо:

$$\vec{w}_1^T(8)\vec{x}_2(8) = -2,$$
$$\vec{w}_2^T(8)\vec{x}_2(8) = 0,$$
$$\vec{w}_3^T(8)\vec{x}_2(8) = -4.$$

Класифікацію виконано правильно, тому вектор ваг не коригуємо:

$$\vec{w}_1(9) = \vec{w}_1(8),$$
$$\vec{w}_2(9) = \vec{w}_2(8),$$
$$\vec{w}_3(9) = \vec{w}_3(8).$$

Подамо вектор зображення $\vec{x}_3(9)=(-1,1,1)^T$, отримаємо:

$$\vec{w}_1^T(9)\ \vec{x}_3(9)=-2;$$
$$\vec{w}_2^T(9)\ \vec{x}_3(9)=-4;$$
$$\vec{w}_3^T(9)\ \vec{x}_3(9)=0.$$

В останньому циклі помилок класифікації не було, тому можемо записати кінцеві значення вагових векторів: $\vec{w}_1 =(0,-2,0)^T$, $\vec{w}_2 =(2,0,-2)^T$, $\vec{w}_3=(-2,0,-2)^T$. Таким чином, шукані вирішувальні функції мають такий вигляд:

$$d_1(\vec{x}) = 0 \cdot x_1 - 2 \cdot x_2 + 0 = -2x_2,$$
$$d_2(\vec{x}) = 2 \cdot x_1 + 0 \cdot x_2 - 2 = 2x_1 - 2,$$
$$d_3(\vec{x}) = -2 \cdot x_1 + 0 \cdot x_2 - 2 = -2x_1 - 2.$$

9.3 Метод градієнта

В цьому розділі розглядається більш загальний підхід до задачі навчання, який базується на добре відомому методі градієнта. Як буде показано, алгоритм перцептрона можна отримати на підставі цього методу.

Як відомо, градієнтні схеми є засобом знаходження мінімуму функції. Градієнт деякої функції $J(\vec{y})$ за вектором $\vec{y}=(y_1, y_2, ...y_n)^T$ визначається таким чином:

$$\nabla J(\vec{y}) = grad\ J(\vec{y}) = \frac{dJ(\vec{y})}{d\vec{y}} = \left(\frac{\partial J}{\partial y_1}, \frac{\partial J}{\partial y_2}, ... \frac{\partial J}{\partial y_u} \right)^T. \qquad (9.17)$$

Як видно із (9.17), градієнт функції $\nabla J(\vec{y})$ є вектором. Його величина визначає швидкість зміни функції $J(\vec{y})$, а напрям збігається з напрямком найбільшого зростання цієї функції. Вектор $-\nabla J(\vec{y})$, який вказує напрям найбільшого зменшення функції $J(\vec{y})$, називається антиградієнтом. Використовуючи цю властивість градієнта, можна будувати ітераційні схеми знаходження мінімуму функції. В точці мінімуму функції градієнт функції дорівнює нулю:

$$\nabla J(\vec{y}^*) = 0, \qquad (9.18)$$

де $\vec{y}*$ – точка мінімуму функції.

Функцію $J(\vec{y}*)$ в околі $\Delta\vec{y}$ точки мінімуму $\vec{y}*$ можна розкласти в ряд Тейлора ($\Delta\vec{y} = (\Delta y_1, \Delta y_2 .. \Delta y_u)$):

$$J(\vec{y}*+\Delta\vec{y}) = J(\vec{y}*) + \Delta\vec{y}^T \nabla J(\vec{y}) + \frac{1}{2}\Delta\vec{y}^T \nabla\nabla^T J(\vec{y}*)\cdot\Delta y + ... \qquad (9.19)$$

Знайдемо градієнт функції (9.19)

$$\nabla\left[J(\vec{y}*+\Delta\vec{y})\right] = \nabla J(\vec{y}*) + \nabla\nabla^T J(\vec{y}*)\cdot\Delta\vec{y} + ...$$

і припустимо, що на цьому кроці оптимуму досягнуто, тобто, з (9.18) записуємо $\nabla\left[J(\vec{y}*+\Delta\vec{y})\right] = 0,$ звідки

$$\Delta\vec{y} = -\left[\ \nabla\nabla^T J(\vec{y}*)\ \right]^{-1}\cdot\nabla J(\vec{y}*),$$
$$\vec{y}*-\vec{y} = -\left[\nabla\nabla^T J(\vec{y}*)\right]^{-1}\cdot\Delta J(\vec{y}*). \qquad (9.20)$$

З (9.20) випливає рекурентне співвідношення для пошуку оптимального значення вектора $\vec{y}*$, яке мінімізує функцію $J(\vec{y}*)$:

$$\vec{y}* = \vec{y} - H^{-1}\nabla J(\vec{y}*), \tag{9.21}$$

де $H = \nabla\nabla^T J(\vec{y}*)$ – гессіан функції $J(\vec{y}*)$, що є матричною формою, елементи якої h_{ij} обчисляються як

$$h_{ij} = \frac{1}{2}\frac{\partial^2 J(\vec{y}*)}{\partial y_i \cdot \partial y_j} \ . \tag{9.22}$$

Якби існували прості методи обчислення матриці H^{-1}, то оптимальне значення вектора $\vec{y}*$ можна було б знайти відразу. Оскільки в більшості випадків таких методів не існує, то знаходження точки мінімуму за формулою (9.21) замінюють ітеративною процедурою послідовного наближення до цього мінімуму

$$\vec{y}(k+1) = \vec{y}(k) - c \cdot \nabla J(\vec{y}),$$

або

$$\vec{y}(k+1) = \vec{y}(k) - c \cdot \left(\frac{\partial J(\vec{y})}{\partial \vec{y}} \right). \tag{9.23}$$

Застосуємо градієнтний метод для навчання класифікатора. Для цього потрібно ввести деяку критеріальну функцію вагового вектора і вектора ознак $J(\vec{w}, \vec{x})$. Беручи до уваги умову (9.9), цю функцію потрібно задати таким чином, щоб вона досягла мінімального значення за умови $\vec{w}^T \vec{x}_i > 0$, де $\vec{x}_i$ – це i-тий рядок матриці **X** розміру $N \times (N+1)$ системи нерівностей (9.3). За цих умов пошук мінімуму функції $J(\vec{w}, \vec{x})$ для всіх i, $i = 1, 2, ..., N$, еквівалентний розв'язку системи лінійних нерівностей.

Розглянемо, наприклад, функцію критерію

$$J(\vec{w}, \vec{x}) = (\left|\vec{w}^T(k+1)\vec{x}\right| - \vec{w}^T(k)\vec{x}), \tag{9.24}$$

де $\left|\vec{w}^T \vec{x}\right|$ – абсолютне значення (модуль) $\vec{w}^T \vec{x}$. Очевидно що мінімум цієї функції є $J(\vec{w}, \vec{x}) = 0$ і досягається він за виконання $\vec{w}^T \vec{x} > 0$.

Градієнтний метод в такому випадку полягає в ітеративному збільшенні значень $\vec{w}$ в напрямку від'ємного градієнта (антиградієнта) функції $J(\vec{w}, \vec{x})$ для знаходження мінімуму цієї функції. Іншими словами,

якщо $\vec{w}(k)$ є значенням вектора $\vec{w}$ на в k-ому кроці, то в загальному вигляді алгоритм градієнтного спуску можна записати таким чином:

$$\vec{w}(k+1) = \vec{w}(k) - c\left[\frac{\partial J(\vec{w},\vec{x})}{\partial \vec{w}}\right]_{\vec{w}=\vec{w}(k)}, \tag{9.25}$$

де $\vec{w}(k+1)$ є новим значенням вектора $\vec{w}$, а величина $c > 0$ визначає розмір корекції. З виразу (9.25) видно, що для $(\partial J / \partial \vec{w}) = 0$, тобто за досягнення мінімуму критеріальної функції $J(\vec{w},\vec{x})$, значення вектора $\vec{w}$ ніяк не змінюється.

Якщо система нерівностей (9.3) сумісна і функція $J(\vec{w},\vec{x})$ задана належним чином, то алгоритм (9.25) забезпечить отримання розв'язку. В іншому випадку він зациклюється.

Покажемо, що підстановка ряду різних функцій критерію $J(\vec{w},\vec{x})$ в рівнянні (9.25) дозволяє отримати декілька окремих алгоритмів.

Нехай критеріальна функція має вигляд

$$J(\vec{w},\vec{x}) = \frac{1}{2}(\left|\vec{w}^T\vec{x}\right| - \vec{w}^T\vec{x}). \tag{9.26}$$

Градієнт функції за вектором $\vec{w}$ визначається, як

$$\frac{\partial J}{\partial \vec{w}} = \frac{1}{2}(\ \left[\vec{x}\cdot \mathrm{sgn}(\vec{w}^T\vec{x}) - \vec{x}\right], \tag{9.27}$$

де, за означенням,

$$\mathrm{sgn}(\vec{w}^T\vec{x}) = \begin{cases} 1, & якщо\ \ \vec{w}^T\vec{x} > 0, \\ -1, & якщо\ \ \vec{w}^T\vec{x} \le 0. \end{cases} \tag{9.28}$$

В виразі (9.28) ми об'єднали в одну дві умови: $\vec{w}^T\vec{x} = 0$ і $\vec{w}^T\vec{x} < 0$.

Підстановка виразу для частинної похідної (9.27) в рівняння (9.25) дає

$$\vec{w}(k+1) = \vec{w}(k) + \frac{c}{2}\{\vec{x}(k) - \vec{x}(k)\cdot \mathrm{sgn}\left[\vec{w}^T(k)\vec{x}(k)\right]\}, \tag{9.29}$$

де $\vec{x}(k)$ є образом з навчальної вибірки, який подається на k-ому кроці ітерації.

Підстановка виразу (9.28) в (9.29) дає формулу для алгоритму:

$$\vec{w}(k+1)=\vec{w}(k)+c\begin{cases}0, & якщо\ \ \vec{w}^T(k)\vec{x}(k)>0,\\ \vec{x}(k), якщо & \vec{w}^T(k)\vec{x}(k)\le 0,\end{cases} \tag{9.30}$$

де $c>0$ і $\vec{w}(1)$ – довільні.

Таким чином, отримано алгоритм перцептрона з фіксованим приростом.

Алгоритм дробової корекції можна отримати з градієнтного методу, якщо вибрати критеріальну функцію у такому вигляді:

$$J(\vec{w},\vec{x})=\frac{1}{4\vec{x}^T\vec{x}}(\left|\vec{w}^T\vec{x}\right|^2-\left|\vec{w}^T\vec{x}\right|\vec{w}^T\vec{x}). \tag{9.31}$$

Частинна похідна функції за вектором $\vec{w}$ (градієнт функції) визначається таким чином:

$$\frac{\partial J}{\partial\vec{w}}=\frac{1}{4\vec{x}^T\vec{x}}\left[2\left|\vec{w}^T\vec{x}\right|\vec{x}\ \ \mathrm{sgn}(\vec{w}^T\vec{x})-\left|\vec{w}^T\vec{x}\right|\vec{x}-(\vec{w}^T\vec{x})\vec{x}\ \ \mathrm{sgn}(\vec{w}^T\vec{x})\right], \tag{9.32}$$

або, виконавши спрощення,

$$\frac{\partial J}{\partial\vec{w}}=\frac{1}{2\vec{x}^T\vec{x}}\left[\left|\vec{w}^T\vec{x}\right|\vec{x}\ \ \mathrm{sgn}(\vec{w}^T\vec{x})-\left|\vec{w}^T\vec{x}\right|\vec{x}\right]. \tag{9. 33}$$

Підстановка останнього виразу в алгоритм градієнта (9.25) дає рівняння алгоритму

$$\vec{w}(k+1)=\vec{w}(k)+\frac{\lambda\left|\vec{w}^T(k)\vec{x}(k)\right|}{2\vec{x}^T(k)\vec{x}(k)}\times\left\{\vec{x}(k)-\vec{x}(k)\mathrm{sgn}[\vec{w}^T(k)\vec{x}(k)]\right\}, \tag{9.34}$$

де приріст корекції c тимчасово позначено через λ з метою уникнення плутанини в процесі подальшого порівняння алгоритму.

Використавши вираз (9.25), остаточно отримуємо:

$$\vec{w}(k+1)=\vec{w}(k)+\frac{\lambda\left|\vec{w}^T(k)\vec{x}(k)\right|}{2\vec{x}^T(k)\vec{x}(k)}\cdot\begin{cases}0, & \text{якщо } \vec{w}^T(k)\vec{x}(k)>0,\\ \vec{x}(k), & \text{якщо } \vec{w}^T(k)\vec{x}(k)\le 0.\end{cases} \tag{9.35}$$

Порівняння виразів (9.34) і (9.35) показує, що отримано алгоритм дробової корекції.

9.4 Алгоритм навчання за методом мінімізації середньоквадратичної помилки (НСКП-алгоритм)

В попередньому розділі було відзначено, що в випадку нероздільних класів перцептронні алгоритми навчання зациклюються. Оскільки вони самі собою не містять ознаки нероздільнсті класів, то вивести їх із стану

може тільки зовнішнє втручання. Тому розглянемо алгоритм, позбавлений цього недоліку. Він називається алгоритмом найменшої середньоквадратичної помилки (НСКП), є збіжним за умови роздільності класів, а за відсутності такої виробляє в процесі виконання ознаку цієї нероздільності і зупиняється.

Для пояснення алгоритму НСКП змінимо постановку задачі, що задається системою нерівностей (9.2), таким чином: замість того, щоб розглядати задачу знаходження вектора ваг $\vec{w}$, який забезпечує виконання умови $\mathbf{X}\vec{w}>0$, будемо намагатися знайти вектори $\vec{w}$ і $\vec{b}$, які забезпечують виконання рівності

$$\mathbf{X}\vec{w}=\vec{b}, \tag{9.36}$$

де всі компоненти вектора $\vec{b}=(b_1,b_2,...,b_n)^T$ є додатними. Зрозуміло, що ці два формулювання задачі є еквівалентними.

Виберемо за критеріальну таку функцію:

$$J\left(\vec{w},\vec{x},\vec{b}\right)=\frac{1}{2}\sum_{j=1}^{N}\left(\vec{w}^T\vec{x}_j-\vec{b}_j\right)^2=\frac{1}{2}\left\|\mathbf{X}\vec{w}-\vec{b}\right\|^2, \tag{9.37}$$

де $\left\|\mathbf{X}\vec{w}-\vec{b}\right\|$ є модулем вектора, тобто його довжиною.

Функція (9.37) досягає мінімуму за виконання умови (9.36). Оскільки ця функція залежить від векторів $\vec{w}$ і $\vec{b}$, будемо виконувати її мінімізацію за цими двома змінними. Вираз $\left(\vec{w}^T\vec{x}-b_j\right)^2$ подає квадратичну помилку, тобто квадрат відхилення величини вирішувальної функції $\vec{w}^T\vec{x}$ від деякого додатного порогу b_j. Оскільки сума цих відхилень пропорційна середньому їх значенню (математичному сподіванню), і оскільки алгоритм намагається його мінімізувати, то він називається алгоритмом найменшої середньоквадратичної помилки (НСКП алгоритмом) або процедурою Хо–Каш'япа.

Знайдемо градієнт функції $J\left(\vec{w},\vec{x},\vec{b}\right)$ за векторами $\vec{w}$ і $\vec{b}$

$$\frac{\partial J}{\partial\vec{w}}=\mathbf{X}^T\left(\mathbf{X}\vec{w}-\vec{b}\right) \tag{9.38}$$

і

$$\frac{\partial J}{\partial\vec{b}}=-\left(\mathbf{X}\vec{w}-\vec{b}\right). \tag{9.39}$$

Оскільки вектор $\vec{w}$ якимось обмеженням не підлягає, то будемо шукати його з рівняння (9.38) і умови мінімуму функції

$$\vec{w} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \mathbf{X}^T \vec{b} = \mathbf{X}^{\#} \vec{b}, \tag{9.40}$$

де $\mathbf{X}^{\#}$ – узагальнене обернення матриці $\mathbf{X}$ або псевдо-обернена матриця $\mathbf{X}$.

Пошук оптимального значення вектора $\vec{b}$ будемо здійснювати покроковою процедурою наближення до мінімуму (9.23), визначеною градієнтним методом. Водночас потрібно враховувати обмеження, яке потребує додатних значень вектора $\vec{b}$. Останнє можна забезпечити, поклавши

$$\vec{b}(k+1) = \vec{b}(k) + \Delta\vec{b}(k), \tag{9.41}$$

де
$$\Delta b_i(k) = \begin{cases} 2c\left[\mathbf{X}\vec{w}(k) - \vec{b}\right]_i, \text{якщо } \left[\mathbf{X}\vec{w}(k) - \vec{b}(k)\right]_i > 0 \\ 0, \qquad\qquad \text{, якщо } \left[\mathbf{X}\vec{w}(k) - \vec{b}(k)\right]_i \le 0. \end{cases} \tag{9.42}$$

В формулах (9.41) і (9.42) k позначає номер кроку ітераційної процедури, i – індекс компоненти вектора, c – додатний приріст корекції, величина якого буде визначена далі.

Рівняння (9.42) можна записати в такій формі:

$$\Delta\vec{b}(k) = c\left[\mathbf{X}\vec{w}(k) - \vec{b}(k) + \left|\mathbf{X}\vec{w}(k) - \vec{b}(k)\right|\right], \tag{9.43}$$

де вираз $\left|\mathbf{X}\vec{w}(k) - \vec{b}(k)\right|$ визначає абсолютну величину кожної компоненти вектора $\left(\mathbf{X}\vec{w}(k) - \vec{b}(k)\right)$.

Підставляючи (9.41) в (9.40), отримаємо:

$$\begin{aligned} \vec{w}(k+1) &= \mathbf{X}^{\#}\vec{b}(k+1) = \mathbf{X}^{\#}\left[\vec{b}(k) + \Delta\vec{b}(k)\right] = \\ &= \mathbf{X}^{\#}\vec{b}(k) + \mathbf{X}^{\#}\Delta\vec{b}(k) = \vec{w}(k) + \mathbf{X}^{\#}\vec{b}(k). \end{aligned} \tag{9.44}$$

Поклавши

$$\mathbf{X}\vec{w}(k) - \vec{b}(k) = \vec{e}(k), \tag{9.45}$$

отримуємо такий алгоритм навчання НСКП:

$$\begin{cases} \vec{w}(1) = \mathbf{X}^{\#}\vec{b}(1), \qquad b_i(1) > 0 \\ \vec{e}(k) = \mathbf{X}\vec{w}(k) - \vec{b}(k), \\ \vec{w}(k+1) = \vec{w}(k) + c\mathbf{X}^{\#}\left[\vec{e}(k) + \left|\vec{e}(k)\right|\right], \\ \vec{b}(k+1) = \vec{b}(k) + c\left[\vec{e}(k) + \left|\vec{e}(k)\right|\right], \end{cases} \tag{9.46}$$

де $|\vec{e}(k)|$ – абсолютна величина вектора відхилення $\vec{e}(k)$.

Якщо нерівність $\mathbf{X}\vec{w} > 0$ має розв'язок, то цей алгоритм збігається для $0 < c \le 1$.

Більш того, якщо на деякому ітераційному кроці всі компоненти вектора $\vec{e}(k)$ стають не додатними ($\vec{e}(k) \le 0$), але не всі дорівнюють нулю, то це означає, то задані класи не можна розділити границями вибраного типу.

В тих випадках коли $\vec{e}(k) = 0$, вектор $\vec{w}(k)$ є розв'язком, тому що $\mathbf{X}\vec{w}(k) = \vec{b}(k)$ і $\vec{b}(k)$ додатний. Наявність критерію роздільності класів є значною перевагою цього алгоритму.

Приклад 9.3. Найти вектор ваг $\vec{w}$ вирішувальної функції для розподілу класів, поданих на рис. 9.2, використавши алгоритм навчання НСКП.

Розв'язування. Поповнимо вектори зображення, тоді

$$\Omega_1 : \left\{ \vec{x}_1 = (0,0,1)^T, \vec{x}_2 = (0,1,1)^T \right\},$$
$$\Omega_2 : \left\{ \vec{x}_3 = (1,0,1)^T, \vec{x}_4 = (1,1,1)^T \right\}.$$

Помножимо вектори другого класу на -1, тоді матриця $\mathbf{X}$ в нерівності (9.2) має вигляд:

$$\mathbf{X} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ -1 & 0 & -1 \\ -1 & -1 & -1 \end{bmatrix}.$$

Знайдемо псевдо-обернену матрицю

$$\mathbf{X}^{\#} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T:$$
$$\mathbf{X}^{\#} = \frac{1}{2}\begin{bmatrix} -1 & -1 & -1 & -1 \\ -1 & 1 & 1 & -1 \\ \frac{3}{2} & \frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \end{bmatrix}.$$

Покладемо $\vec{b}(1) = (1,1,1,1)^T$, $c = 1$ і застосуємо алгоритм (9.46), отримаємо

$$\vec{w}(1) = \mathbf{X}^{\#} \cdot \vec{b}(1) = (-2,0,1)^T.$$

Оскільки $\mathbf{X}\cdot\vec{w}(1)=(1,1,1,1)^T$ то з (9.36) випливає, що вектор $\vec{w}(1)$ є рішенням. Вектор помилки $\vec{e}(1)$ дорівнює нулю.

Відповідь: вектор ваг $\vec{w}=(-2,0,1)$, а вирішувальна функція $d(\vec{x})=\vec{w}^T\vec{x}=-2x_1+1$.

Як бачимо з прикладу 9.3, алгоритм дав рішення уже на першому кроці. Швидкість збіжності НСКП алгоритму навчання пояснюється двома факторами:

1. На кожному кроці здійснюють зміну обох векторів $\vec{w}$ і $\vec{b}$.
2. Процедура являє собою адаптивну схему, яка враховує на кожному кроці інформацію зразу про всі образи обох класів.

Єдиним і явним недоліком застосування НСКП алгоритму є необхідність обернення матриці $[\mathbf{X}^T\mathbf{X}]$. У випадку невеликої розмірності векторів зображення цей недолік не викликає великих обчислювальних затрат, оскільки обернення матриці здійснюється тільки один раз. Крім того, можна використовувати різні рекурентні формули у разі появи нових векторів зображення.

В тих випадках, коли ранг матриці $\mathbf{X}$ не дорівнює $n+1$ (n – розмірність простору ознак), $[\mathbf{X}^T\mathbf{X}]$ не має оберненої матриці, тому в цьому разі застосовують модифікації алгоритму НСКП. Розглянемо одну з них. Вона відрізняється від описаного вище алгоритму НСКП тим, що матриця X виразу (9.36) будується з векторів зображень деякого одного класу Ω_{m}, а за вектор $\vec{b}$ вибирається еталон $\overrightarrow{Mg}$, цього класу

$$\vec{\mu}_{\text{m}}=\frac{1}{Q}\sum_{q=1}^{Q}\vec{x}_q\ ,$$

де Q – кількість векторів у класі Q,

$\vec{X}_{\text{g}}$ – деякий вектор класу $\vec{x}_q=(\dot{x}_{q1},\dot{x}_{q2},...,\dot{x}_{qn})$, q = 1, 2, 3, …, Q. Як бачимо, за такого вибору вектор $\vec{\mu}_{\text{m}}$ не є обов'язково додатним, а матриця $\mathbf{X}$ записується, як:

$$x=[\vec{x}_1,\vec{x}_2,...,\vec{x}_q,...,\vec{x}_Q] \qquad (9.47)$$

На рис. 9.3 зображено процедуру навчання лінійного класифікатора, яка відповідає цій модифікації НСКП алгоритму.

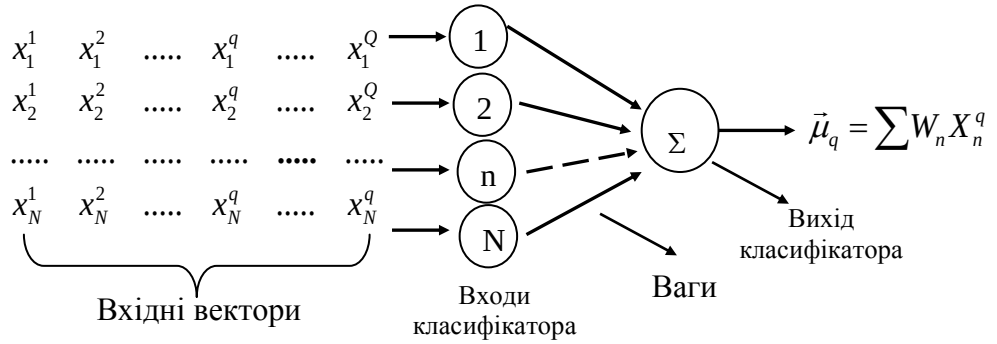


Рисунок 9.3 – Навчання лінійного класифікатора за методом мінімізації середньоквадратичного відхилення

Вибираємо вектор ваг $\vec{W}$ таким чином, щоб мінімізувати середньоквадратичну похибку класифікації. З рис. 9.3 видно, що вибраний належним чином вектор ваг $\vec{W}$ виконує лінійне відображення вхідних векторів $\vec{X}_q$ (зображень об'єктів) у вихідний вектор $\vec{\mu}_q$, який є наближеним значенням еталона $\vec{\mu}$ класу Ω_m. Процедура навчання в цьому випадку формує вектор ваг таким чином, щоб мінімізувати середньоквадратичну похибку класифікації за всіма векторами зображень

$$\vec{X}_q:\ E(w)=\left\|X^T\vec{w}-\vec{\mu}_m\right\|^2=\sum_{q=1,Q}\ \sum_{n=1}^{N}(x_{qn}W_n-\mu_{mn})^2. \qquad (9.48)$$

Величина *E(W)* є критеріальною функцією, градієнт якої використовується для виконання алгоритму навчання як процедури мінімізації.

Мінімізацію починають з довільно вибраного початкового вектора $\vec{W}(1)$ і потім рухаються від деякої поточної точки $\vec{W}(k)$ в напрямку найшвидшого спуску $\Delta E(\vec{W}(k))$ для знаходження наступного наближення до розв'язку $\vec{W}(k+1)$. Градієнт функції (9.48) визначається як

$$\Delta E(\vec{W})=X^T(x\vec{w}-\vec{\mu}_m), \qquad (9.49)$$

і не потребує обчислення оберненої матриці. В методі Ньютона, який використовує матрицю Гессіана других частинних похідних, застосовується приріст корекції, або навчальна поправка, $\delta(k)$ на кожному кроці наближення. Величина цього приросту в кінцевому результаті має зменшитись до нуля для досягнення збіжності для $k\to\infty$. Алгоритм середньоквадратичної помилки в цій модифікації містить такі кроки.

Крок 1. Вибирається довільним чином початкове значення вектора ваг $\vec{W}$ (1), і задається кількість ітерацій К.

Крок 2. Обчислення середньоквадратичної помилки і нових ваг в напрямку найшвидшого спуску:

$$\mathrm{E}(\vec{\mathrm{w}}) = \left\| \mathrm{X}^{\mathrm{T}} \vec{w} - \vec{\mu} \right\|^2,$$

$$\vec{w} = \vec{w} - \delta \cdot \nabla E(\vec{w}).$$

Крок 3: Зміна номера ітерації і поправки приросту корекції

$$k = k + 1,$$

$$\delta = f(\delta),$$

де $f(\delta)$ – функція для обчислення приросту корекції.

Якщо K>l або $\delta \le 0{,}005$, то зупинка, інакше перехід на крок 2.

9.5 Навчання класифікатора за методом потенціальних функцій

Припустимо, що необхідно розділити два класи Ω_1 і Ω_2. Вибіркові зображення подано векторами $\vec{x}_i$ (або точками $\vec{x}_i$) в n-вимірному просторі ознак. Уявимо собі, що в кожній такій точці розміщено електричний заряд. В деякій довільній точці p сукупність всіх зарядів, що створюються кожним окремим потенціалом:

$$\varphi(p) = \frac{1}{4\pi e_0} \sum_i \frac{q_i}{d_i} = \sum \varphi(p, p_i), \qquad (9.50)$$

де q_i – заряд в точці p_i;

d_i – відстань від точки p_i до точки p.

Потенціал описується функцією, симетричною відносно точки, в яку внесено заряд; в цій точці, згідно з (9.50), потенціал сягає нескінченності. Лінії, що з'єднують точки однакового потенціалу, називаються еквіпотенціальними. Відійшовши від суто електричної специфіки цих понять, можна за аналогією вважати, що всяка сукупність точок, яка утворює деякий клас, має вигляд потенціального плато, відділеного від потенціального плато іншого класу глибокою долиною (рис. 9.4).

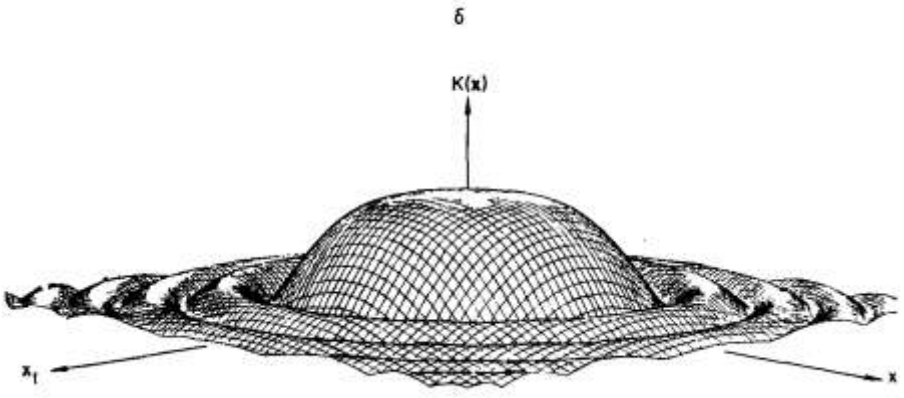


Рисунок 9.4 – Потенціальна функція у двовимірному просторі ознак

Можна вважати, що кластер, утворений вибірковими зображеннями класу Ω_1 утворює «плато», причому вибіркові образи розміщуються на вершинах деякої групи пагорбів. Подібну геометричну інтерпретацію можна застосувати і для образів класу Ω_2. Ці два «плато» розділені долиною, в якій, як вважається, потенціал спадає до нуля. На підставі цих інтуїтивних викладок створено метод потенціальних функцій, який дозволяє визначати вирішувальні функції в задачах класифікації образів. Ці вирішувальні функції можна отримати з потенціальних функцій для векторів $\vec{x}_k$, що подають вибіркові зображення в просторі ознак. Потенціальна функція $K(\vec{x},\vec{x}_k)$ точки $\vec{x}_k$, згідно з теорією апроксимації функцій, може бути подана виразом:

$$K(\vec{x},\vec{x}_k)=\sum_{i=1}^{\infty}\lambda_i y_i(\vec{x})\lambda_i y_i(\vec{x}_k), \tag{9.51}$$

де функції $y_i(\vec{x}), y_i(\vec{x}_k)$ вибираються ортонормованими, а дійсні числа λ_i вибираються таким чином, щоб функція $K(\vec{x},\vec{x}_k)$ була обмеженою за умови підсумовування членів нескінченного ряду (9.51).

Вирішувальні функції, які отримують відповідно до виразу (9.51), називаються потенціальними функціями I типу.

Потенціальні функції II типу будуються з міркувань їх відповідності визначенню потенціалу, тобто симетрії $K(\vec{x},\vec{x}_k)=K(\vec{x}_k,\vec{x})$, і оберненій пропорціональності до відстані. Як приклад цих функцій можна навести такі:

$$K(\vec{x},\vec{x}_k)=\exp(-\alpha\left\|\vec{x}-\vec{x}_k\right\|^2), \tag{9.52}$$

$$K(\vec{x},\vec{x}_k)=\frac{1}{1+\alpha\left\|\vec{x}-\vec{x}_k\right\|^2}, \tag{9.53}$$

$$K(\vec{x},\vec{x}_k)=\frac{\sin\alpha\left\|\vec{x}-\vec{x}_k\right\|^2}{\alpha\left\|\vec{x}-\vec{x}_k\right\|^2}, \tag{9.54}$$

де α – додатна константа,

$D^2=\left\|\vec{x}-\vec{x}_k\right\|^2$ – квадрат відстані D.

Процедура навчання полягає в послідовному поданні системі розпізнавання вибіркових зображень, обчисленню їх потенціальних функцій і знаходженню кумулятивного (сумарного) потенціалу $K(\vec{x})$ як суми потенціалів $K(\vec{x},\vec{x}_k)$ окремих точок.

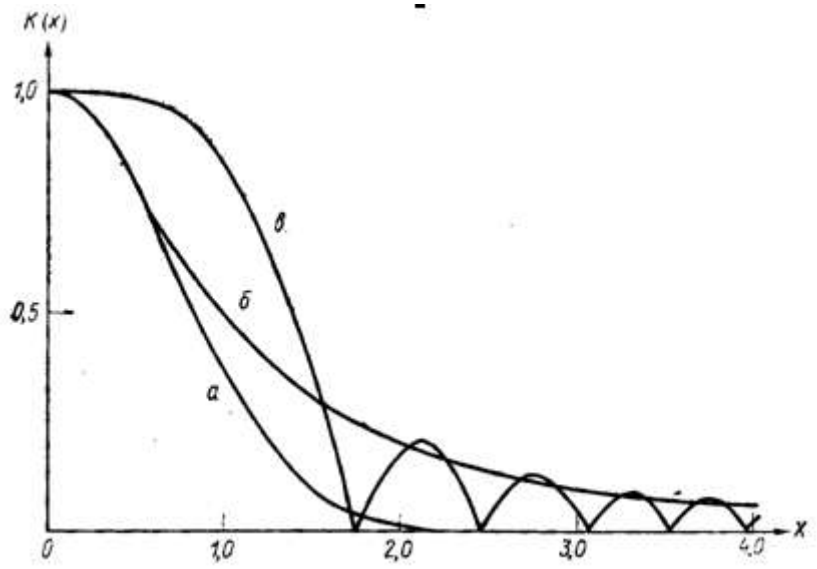


Рисунок 9.5 – Приклади одновимірних потенціальних функцій: а) – графік функції за формулою 9.52; б) – графік функції за формулою 9.53; в) – графік функції за формулою 9.54

На початку етапу навчання значення кумулятивного потенціалу $K_0(\vec{x})$ вибирається для зручності запису таким, що дорівнює нулю. На $(k+1)$-му кроці навчання він коригується таким чином

$$K_{k+1}(\vec{x}) = K_k(\vec{x}) + \rho_{k+1} K(\vec{x}, \vec{x}_{k+1}), \tag{9.55}$$

де

$$\rho_{k+1} = \begin{cases} +1, \textit{якщо } \vec{x}_{k+1} \in \Omega_1 \quad i \quad K_k(\vec{x}_{k+1}) \le 0, \\ -1, \textit{якщо } \vec{x}_{k+1} \in \Omega_2 \quad i \quad K_k(\vec{x}_{k+1}) > 0, \\ 0, \textit{якщо } \vec{x}_{k+1} \in \Omega_1 \quad i \quad K_k(\vec{x}_{k+1}) > 0, \\ 0, \textit{якщо } \vec{x}_{k+1} \in \Omega_2 \quad i \quad K_k(\vec{x}_{k+1}) < 0. \end{cases} \tag{9.56}$$

Вирази (9.55) і (9.56) можна тлумачити таким чином. В тих випадках, коли розподільна границя, визначена кумулятивним потенціалом $K_k(\vec{x})$, не забезпечує правильної класифікації, значення кумулятивного потенціалу збільшується або зменшується на величину $K(\vec{x}, \vec{x}_{k+1})$ залежно від належності вектора $\vec{x}_{k+1}$ до Ω_1 чи Ω_2. Якщо зображення $\vec{x}_{k+1}$ класифікується правильно, то кумулятивний потенціал не змінюється.

Приклад 9.4. Нехай задано образи Ω_1 і Ω_2 своїми вибірковими значеннями, $\Omega_1 : \{\vec{x}_1 = (1,0)^T, \vec{x}_2 = (0,-1)^T\}$, $\Omega_2 : \{\vec{x}_3 = (-1,0)^T, \vec{x}_4 = (0,1)^T\}$. Визначити вирішувальну функцію методом потенціальних функцій, використавши потенціальні функції I типу.

Розв'язування. Для побудови вирішувальної функції використаємо урізаний ряд (9.51) :

$$K(\vec{x},\vec{x}_k)=\sum_{i=1}^{m} y_i(\vec{x})y_i(\vec{x}_k), \qquad (9.57)$$

де $y_i(\vec{x})$ – ортонормовані функції на множині зображень. Оскільки (9.57) є урізаним рядом (9.51) і сума його членів є обмеженою, то коефіцієнти λ_i, пов'язані з вимогою обмеженості потенціальних функцій, у виразі (9.57) опущено.

Для наочності і спрощення обчислень замість ортонормованих функцій скористаємося їх ортогональними аналогами, за які можна вибрати поліноми Ерміта (Розділ 4 першої частини цього посібника):

$$H_0(x)=1, H_1(x)=2x, H_2(x)=4x^2-2,$$
$$H_3(x)=8x^3-12x, H_4(x)=16x^4-48x^2+12.$$

В першому наближенні виберемо *m*=4 і сформуємо ортогональні функції двох змінних з ортогональних функцій однієї змінної:

$$y_1(\vec{x})=y_1(x_1,x_2)=H_0(x_1)H_0(x_2)=1,$$
$$y_2(\vec{x})=y_2(x_1,x_2)=H_1(x_1)H_0(x_2)=2x_1,$$
$$y_3(\vec{x})=y_3(x_1,x_2)=H_0(x_1)H_1(x_2)=2x_2,$$
$$y_4(\vec{x})=y_4(x_1,x_2)=H_1(x_1)H_1(x_2)=4x_1x_2,$$

Скориставшись співвідношенням (9.57), формуємо потенціальну функцію

$$K(\vec{x},\vec{x}_k)=\sum_{i=1}^{n} y_i(\vec{x})y_i(\vec{x}_k)=1+4x_1x_{k1}+4x_2x_{k2}+16x_1x_2x_{k1}x_{k2},$$

де x_{k1} і x_{k2} є координатами вектора $\vec{x}_k$.

Використовуємо алгоритм навчання за методом потенціальних функцій (9.56).

Подамо перше зображення $\vec{x}_1=(1,0)^T, \vec{x}_1\in\Omega_1$:

$$K_1(\vec{x})=K(\vec{x},\vec{x}_k)=1+4x_1\cdot 1+4x_2\cdot 0+16x_1x_2\cdot 1\cdot 0=1+4x_1.$$

Обчислимо $K_1(\vec{x}_2)$, де $\vec{x}_2=(0,-1)^T, \vec{x}_2\in\Omega_1$.

$$K_1(\vec{x}_2)=1+4x_1=1+4\cdot 0=1.$$

Оскільки $K_1(x_2) > 0$ і $\vec{x}_2 \in \Omega_1$, то класифікація виконана правильно, і кумулятивний потенціал не коректується: $K_2(\vec{x}) = K_1(\vec{x}) = 1 + 4x_1$.

Наступний поданий вектор $\vec{x}_3 = (-1,0)^T$ належить класу Ω_2:

$$K_2(\vec{x}_3) = 1 + 4 \cdot (-1) = -3, K_2(\vec{x}_3) < 0, \text{ тому } K_3(\vec{x}) = K_2(\vec{x}) = 1 + 4x_1.$$

Подамо четверте зображення $\vec{x}_4 = (0,1)^T$, $x_4 \in \Omega_2$:

$$K_3(\vec{x}_4) = 1 + 4 \cdot 0 = 1, K_3(\vec{x}_4) > 0,$$

тому проводимо корекцію:

$$K_4(\vec{x}) = K_3(\vec{x}) - K(\vec{x}, \vec{x}_4) = 1 + 4x_1 - (1 + 4x_1 \cdot 0 + 4x_2 \cdot 1 + 16x_1x_2 \cdot 0 \cdot 1) =$$
$$= 1 + 4x_1 - (1 + 4x_2) = 4x_1 - 4x_2$$

Оскільки було допущено одну помилку на цьому етапі, повторюємо цикл навчання знову:

$$\vec{x}_5 = \vec{x}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \in \Omega_1; \quad K_4(\vec{x}_5) = 4 \cdot 1 - 4 \cdot 0 = 4,$$

$$K_5(\vec{x}) = K_4(\vec{x}_5) = 4x_1 - 4x_2;$$

$$\vec{x}_6 = \vec{x}_2 = \begin{pmatrix} 0 \\ -1 \end{pmatrix} \in \Omega_1; \quad K_5(\vec{x}_6) = 4 \cdot 0 - 4 \cdot (-1) = 4,$$

$$K_6(\vec{x}) = K_5(\vec{x}_6) = 4x_1 - 4x_2;$$

$$\vec{x}_7 = \vec{x}_3 = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \in \Omega_2; \quad K_6(\vec{x}_7) = 4 \cdot (-1) - 4 \cdot 0 = -4,$$

$$K_7(\vec{x}) = K_6(\vec{x}_7) = 4x_1 - 4x_2;$$

$$\vec{x}_8 = \vec{x}_4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \in \Omega_2; \quad K_7(\vec{x}_8) = 4 \cdot 0 - 4 \cdot 1 = -4,$$

$$K_8(\vec{x}) = K_7(\vec{x}_8) = 4x_1 - 4x_2.$$

Оскільки в цьому циклі навчання помилок класифікації не було, то це значить, що алгоритм навчання збігся і видав вирішувальну функцію

$$d(\vec{x}) = K_8(\vec{x}) = 4x_1 - 4x_2.$$

Розподільну границю, що відповідає цій функції, зображено на рисунку 9.6.

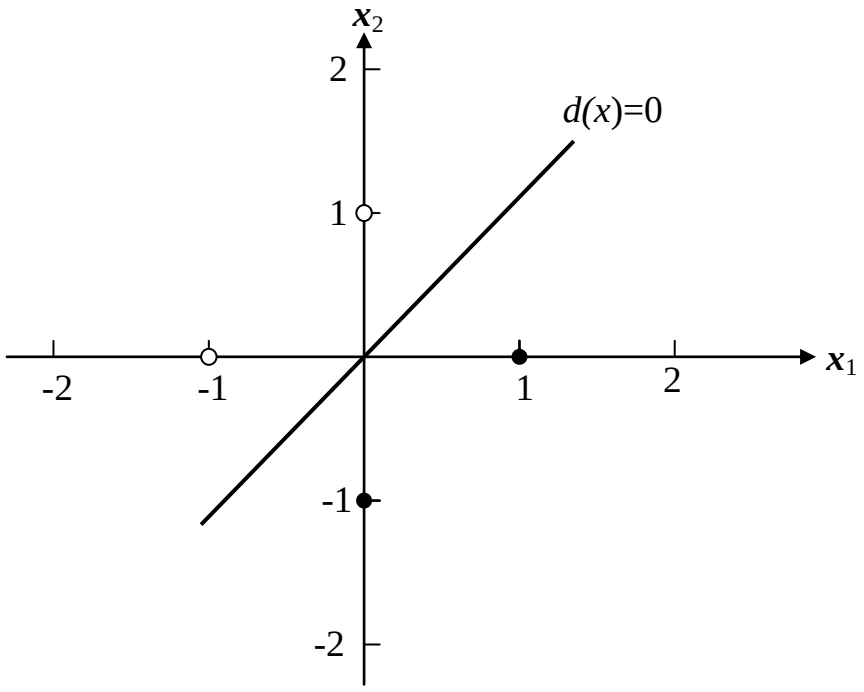


Рисунок 9.6 – Вирішувальна потенціальна функція до прикладу 9.4

9.6 Загальні міркування з проблеми навчання

Описані в попередніх розділах алгоритми навчають систему розпізнавання правильній класифікації шляхом визначення коефіцієнтів вирішувальних функцій на підставі поданих в сеансі навчання вибіркових векторів зображень. Процедура навчання вважається завершеною лише тоді, коли класифікатор в подальшому може правильно розпізнати будь-які вектори, а не тільки ті, які брали участь в навчанні. Ця властивість класифікатора називається здатністю до узагальнення. Виходячи з цієї умови, виникає таке питання: скільки об'єктів потрібно взяти в навчальній виборці, щоб виробити в класифікаторі здатність до узагальнення. Очевидна, на перший погляд, відповідь «чим більше, тим краще» неекономічна з погляду трудомісткості та обчислювальних затрат.

Аналітичне розв'язання цього питання наведено в [14], де показано, що за умови повної відсутності інформації імовірнісного характеру загальна кількість образів в навчальній вибірці для вирішення задачі класифікації на 2 класи має дорівнювати щонайменше подвоєній розмірності образів. Цей висновок узгоджується з поняттям дихотомізаційної потужності $C_N = 2(N+1)$, визначеної в Розділі 3, де $N+1$ дорівнює кількості ваг, що входять до вирішувальної функції. Поняття

дихотомізаційної потужності, по суті, відповідає тому, що за кількості Q навчальних зображень, що мають гарне розміщення в просторі ознак, меншій за C_N, ймовірність правильної дихотомізації є невисокою. Тому, наприклад, якщо розмірність простору ознак $N=5$, то для якісного навчання класифікатора потрібно взяти не менше дванадцяти векторів зображень, тобто $Q \geq 12$. В літературі подається емпірична рекомендація для вибору Q у вигляді:

$$Q = 10 * C_N. \qquad (9.58)$$

Якщо ж класи неможливо розділити за допомогою вибраної розподільної поверхні, для визначення максимальної кількості зображень, які правильно класифікуються цією поверхнею, можна скористатися відомими в літературі рекомендаціями. Описаний в ній для детерміністського випадку алгоритм дає можливість зробити висновок про репрезентативність вибірки. Якщо відомо, що класи розділити неможливо, то отримання алгоритмом більш ніж одного оптимального рішення свідчить про те, що навчальна множина або не є репрезентативною, або складність вибраних вирішувальних функцій неадекватна.

Стосовно отриманих за градієнтною схемою перцептронного і НСКП алгоритмів навчання можна зробити такі порівняння. Алгоритм перцептрона приваблює простотою реалізації і можливістю безпосереднього узагальнення на випадок класифікації багатьох класів. НСКП алгоритм складніший за перцептронний алгоритм, проте він дозволяє уникнути зациклення за рахунок наявності в ньому критерію роздільності класів.

Збіжність цих алгоритмів є однозначним показником їх якості. Наприклад, НСКП алгоритм збігається за меншу кількість ітерацій, ніж перцептронний алгоритм, однак на одну ітерацію в НСКП алгоритмі витрачається більше обчислювальних операцій, ніж в перцептронному. Більш того, як для першого, так і для другого алгоритмів обчислювальні затрати суттєво залежать від геометричних властивостей задачі і початкових значень вагових векторів. Це робить формулювання критеріїв прямого порівняння алгоритмів неможливим.

Контрольні питання та завдання

1. Використайте алгоритм корекції абсолютної величини для класифікації зображень, наведених на рис. 9.2.

2. Розв’яжіть приклад 9.1, застосувавши алгоритм дробової корекції.

3. Задано два класи зображень Ω_1: { $(-1,0)^T$, $(0,-1)^T$, $(1,0)^T$, $(0,1)^T$ } і Ω_2: { $(-2,0)^T$, $(0,-2)^T$, $(2,0)^T$, $(0,2)^T$ }. Побудуйте шляхом навчання класифікатор, використавши перцептронний алгоритм з фіксованим приростом корекції. Зверніть увагу на те, що класи не розділяються лінійно.

4. Задано два класи зображень Ω_1: { $(0,0)^T$, $(2,0)^T$, $(2,2)^T$, $(0,2)^T$ } і Ω_1: { $(4,4)^T$, $(6,4)^T$, $(6,6)^T$, $(4,6)^T$ }. Використайте для навчання класифікатора перцептронний алгоритм корекції абсолютної величини, отримайте вирішувальну функцію і побудуйте її.

5. Застосуйте алгоритм дробової класифікації до класів Ω_1:{$(0,0,0)^T$, $(1,0,0)^T$, $(1,0,1)^T$, $(1,1,0)^T$ } і Ω_2: { $(0,0,1)^T$, $(0,1,1)^T$,$(0,1,0)^T$,$(1,1,1)^T$ }. Взяти за початкове значення вектора ваг $\vec{w}(1) = (-1,-2,-2,0)^T$. Нарисуйте знайдену розподільну границю.

6. Знайдіть вирішувальні функції для розподілу зображень на три класи: Ω_1: { (0,0), (1,1)}; Ω_2 : { (0,5), (-1,5)}; Ω_3: { (4,3), (5,2) }. Використайте для цього алгоритм навчання перцептрона для випадку 3 лінійних вирішувальних функцій. Нарисуйте знайдені функції.

7. Побудуйте градієнтний алгоритм розподілу на два класи, використавши для цього формулу (9.25) та функцію критерію

$$J(\vec{w},\vec{x},b) = \frac{1}{8\|x\|^2}[(\vec{w}^T\vec{x} - b) - |\vec{w}^T\vec{x} - b|]^2, \text{ де } b > 0.$$

8. Використайте алгоритм п. 7 для класифікації образів з прикладу 9.1 за умови $c = b = 1$.

9. Доведіть, що алгоритм з п. 7 для роздільних класів збігається за умови $0 < c < 2$.

10. Розв'яжіть задачу п. 7, скориставшись НСКП алгоритмом.

11. Використайте НСКП алгоритм для визначення розподільної границі двох класів, що складаються з одновимірних зображень Ω_1:{1}, Ω_2:{0} за початкових значень $c = 1$ і $b(1) = 1$.

12. Перевірте за допомогою НСКП алгоритму лінійну роздільність таких класів: Ω_1:{$(-1;-1)^T$,$(0;0)^T$,$(1,1)^T$} і Ω_2:{$(-1;1)^T$,$(1;-1)^T$}.

13. Застосуйте алгоритм методу потенціальних функцій для знаходження розподільної границі для класів Ω_1:{$(0;1)^T$,$(0;-1)^T$} і Ω_2:{$(1;0)^T$,$(-1;0)^T$}. Використайте вирішувальну функцію другого порядку, що має вигляд 9.52.

14. Задачу п. 13 розв'язати з використанням потенціальної функції виду 9.53.

15. Розв'яжіть задачу п. 13, використавши в потенціальній функції I типу (9.51) тільки лінійні члени.

16. Сформулюйте загальну постановку задачі контрольованого навчання.

17. З яких елементів складається модель перцептрона?

18. З яких етапів складається перцептронний алгоритм навчання?

19. Запишіть формулу перцептронної корекції вагового ветора з фіксованим приростом.

20. Запишіть формулу приросту в перцептронному алгоритмі корекції абсолютної величини.

21. Запишіть формулу приросту в перецептронному алгоритмі дробової корекції.

22. Сформулюйте математичне підґрунтя градієнтного методу навчання.

23. Запишіть формулу навчання класифікатора методом градієнтного спуску.

24. В чому полягає ідея класифікації образів методом потенціальних функцій?

25. Запишіть формулу потенціальної функції I типу.

26. З яких міркувань будуються потенціальні функції II типу?

27. В якому випадку процедура навчання вважається завершеною?

28. З яких міркувань вибирається кількість зображень (прецедентів) у навчальній вибірці? Наведіть формулу.

29. Які переваги і недоліки має перцептронний алгоритм навчання порівнянно з алгоритмом НСКП?

30. Від чого залежать обчислювальні затрати перцептронного і НСКП алгоритмів навчання?

РОЗДІЛ 10
НАВЧАННЯ З УЧИТЕЛЕМ. ДЕРЕВА РІШЕНЬ

10.1 Основні означення в задачі навчання дерева рішень

Одним із найпоширеніших методів прийняття рішень є використання моделі дерева рішень. Для побудови такої моделі використовується його навчання на навчальній вибірці прецедентів, вигляд опису яких залежить від характеру предметної області, в якій вирішується задача. Модель навченого дерева рішень відповідає деякій дискретній цільовій функції, а сама процедура навчання задається згідно з визначенням Т. Мітчелла [8]:

«Definition: A computer program is said to learn from experience ***E*** with respect to some class of tasks ***T*** and performance measure ***P,*** if its performance at tasks in ***T,*** as measured by ***P,*** improves with experience ***E***».

В перекладі: «Кажуть, що комп'ютерна програма навчається під час вирішення якоїсь задачі, вибраної з множини класів задач ***T,*** якщо її ефективність, відповідно до вибраного критерію ***P***, поліпшується за накопичення досвіду ***E***».

До найбільш популярних класів задач ***T*** в машинному навчанні відносяться ті задачі, які вже були розглянуті в попередніх розділах:

- класифікація – віднесення об'єкта до однієї з категорій на підставі його ознак;

- регресія – прогнозування кількісної ознаки об'єкта на підставі інших його ознак;

- кластеризація – розбиття множини об'єктів на групи на підставі ознак цих об'єктів так, щоб усередині груп об'єкти були схожі між собою, а поза однієї групи – менш схожі.

Окрім названих, поширеним класом задач є знаходження аномалій в наборах даних, тобто об'єктів або їх груп, сильно несхожих на інші в цьому наборі (це так званий клас методів «Data Mining») та багато інших .

Досвід ***E*** отримується алгоритмом навчання шляхом аналізу впливу даних на результат навчання. Як уже було раніше визначено, залежно від того, які дані використовуюься в навчальні вибірці – марковані чи немарковані, алгоритми машинного навчання можуть бути поділені на навчання з учителем (supervised lerning) і без вчителя (unsupervised learning). У першому випадку для кожного об'єкта навчальної вибірки відома належність до того чи іншого класу, у другому такої інформації немає. Розглянуті в попередніх розділах задачі класифікації і регресії – це завдання навчання з учителем.

Для прикладу, в задачі прогнозування повернення чи неповернення позичальником кредиту досвідом ***E*** є отримана за рахунок попередньої кредитної історії така навчальна вибірка: набір позичальників, кожен з

яких характеризується таким набором ознак як вік, зарплата, тип кредиту та інші, а також цільовою ознакою (класом), яка описується фактом повернення (1) або неповернення (0) кредиту. В цьому випадку ми маємо справу з задачею бінарної класифікації. Якщо відомо, на скільки (за часом) клієнт затягнув з поверненням кредиту і ставиться завдання те ж саме прогнозувати для нових клієнтів, то це відповідає задачі регресії.

Третьою узагальненою складовою у визначенні машинного навчання є метрика (критерій) ***P*** оцінки ефективності алгоритму. Для різних задач і відповідно алгоритмів вибираються різні критерії, і про них можна конкретно говорити, описавши вирішувану проблемну область. Можна тільки відзначити, що найпростішою метрикою якості алгоритму, яка вирішує задачу класифікації є частка правильних відповідей (accuracy), тобто просто частка вірних прогнозів алгоритму на тестовій вибірці.

10.2 Способи подання дерева рішень

Навчені дерева можуть бути подані у графічному вигляді, а також у вигляді наборів правил індуктивного виводу «Якщо – то» (if-then) для поліпшення їх розуміння людьми.

У графічному вигляді дерево рішень, подібно його «прототипу» з живої природи, складається з **кореня** (початкового вузла), **гілок** з ознаками (від них залежить результат – цільова функція), **листків** зі значеннями цільової функції (ними є вирішувальні вершини – результат вибору певних значень ознак), а також **вузлів** – випадкових вершин, в яких визначаються можливі варіанти розвитку подій з певного моменту. «Зростає» дерево до тих пір, поки альтернативні варіанти не почнуть повторюватися.

Дерева рішень класифікують об’єкти, сортуючи їх по дереву від кореня до деякого листка (кінцевого вузла на гілці), який забезпечує класифікацію. Кожен вузол у дереві задає тестування якоїсь ознаки, а кожна гілка, що виходить з цього вузла, відповідає одному з можливих значень цієї ознаки. Класифікація отриманого на вхід прецедента об’єкта починається з кореневого вузла дерева шляхом тестування ознаки, заданої цим вузлом, а потім рухається вниз по гілці дерева, що відповідає значенню ознаки в заданому прикладі. Потім цей процес повторюється для піддерева, що виходить з наступного вузла.

На рис. 10.1 показано приклад дерева рішень, навченого на класифікацію днів відповідно до того, чи підходять вони для гри в теніс чи ні. Визначення класу рішень здійснюється шляхом ієрархічної класифікації по дереву до відповідного листкового (кінцевого) вузла, а потім остаточної класифікації, пов’язаної з цим листком (у цьому випадку є два класи рішень: ТАК або НІ).

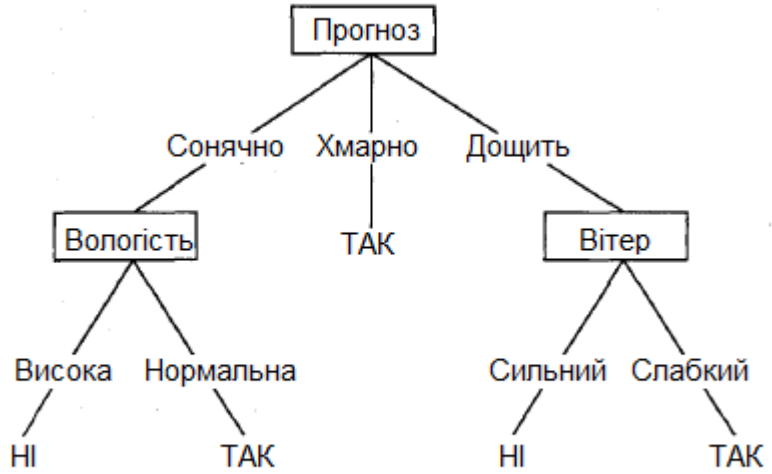


Рисунок 10.1 – Приклад дерева для прийняття рішення «Грати в теніс»

Наприклад, прецедент у вигляді набору ознак: («прогноз = сонячно» і «вологість = висока») буде відсортовано за крайньою лівою гілкою цього дерева рішень і, отже, буде класифіковано як негативний прецедент (тобто дерево передбачає, що вибирається клас НІ для рішення «Грати в теніс»).

В загальному випадку дерева рішень подають розгалуження гілок у вигляді логічної операції диз'юнкції (∨) кон'юнкцій (∧) обмежень на значення ознак об'єктів. Кожен шлях від кореня дерева до листка відповідає кон'юнкції тестів ознак, а саме дерево – диз'юнкції цих кон'юнкцій. Наприклад, дерево рішень, зображене на рисунку 10.1, відповідає логічному виразу:

$$(\text{Прогноз} = \text{Сонячно} \wedge \text{Вологість} = \text{Нормальна} \vee \text{Висока}) \vee (\text{Прогноз} = \text{похмуро}) \vee (\text{Прогноз} = \text{Дощ} \wedge \text{Вітер} = \text{Слабкий} \vee \text{Сильний}) .$$

10.3 Сфери засосування моделі дерева рішень

Незважаючи на те, що розроблено різноманітні методи навчання дерева прийняття рішень з дещо різними можливостями та вимогами, навчання дерева прийняття рішень, як правило, найкраще підходить для проблем із такими характеристиками:

- об'єкти подано параметрами ознака-значення, за такої умови об'єкти описуються фіксованим набором ознак (наприклад, Температура) та їх значеннями (наприклад, Жарко). Найпростіша ситуація для навчання дерева рішень – це коли кожна ознака набуває невеликої кількості несуміжних можливих значень (наприклад, Жарко, Тепло, Холодно). Однак можна застосувати такі розширення базового алгоритму, які

дозволяють обробляти ознаки з реальними величиниами (наприклад, числове подання температури);

- цільова функція має дискретні вихідні значення. Дерево рішень на рис. 10.1 призначає логічну класифікацію (наприклад, ТАК чи НІ) кожному прикладу. Методи дерева рішень легко поширюються на навчальні функції з більш ніж двома можливими вихідними значеннями. Більш суттєве розширення дозволяє навчати цільові функції з реальними величинами, хоча застосування дерев рішень у цьому параметрі є менш поширеним;

- у випадках, коли потрібно використовувати диз'юнктивні описи. Як зазначалося вище, дерева рішень, природно, подають диз'юнктивні вирази;

- у випадках, коли приклади для навчання можуть містити помилки. Методи навчання в дереві рішень надійні щодо помилок: як помилок у класифікації навчальних об’єктів, так і помилок у значеннях ознак цих об’єктів;

- у випадках, коли в даних для навчання можуть бути відсутніми значення деяких ознак. Методи дерева рішень можна використовувати навіть тоді, коли деякі навчальні об’єкти мають невідомі значення (наприклад, якщо Вологість дня відома лише для деяких прикладів навчання).

Існує багато практичних проблем, які відповідають цим характеристикам. Навчання дерева прийняття рішень застосовується до таких проблем, як навчання класифікації медичних пацієнтів за їх захворюваннями, визначення несправності обладнання за відомою причиною та визначення позичальників кредитів за ймовірністю несплати платежів. Такі проблеми, в яких завдання полягає в тому, щоб класифікувати приклади в одну із можливих категорій дискретного набору, часто називають проблемами класифікації.

10.4 Процес конструювання дерева рішень

Як уже було сказано, розглянуте в прикладі завдання класифікації відноситься до стратегії навчання з учителем. В такій стратегії спочатку збирається статистична інформація про об'єкти, для яких вже відомо їх клас, – набір цих об'єктів називається навчальною вибіркою. Потім до цих даних застосовується алгоритм навчання, внаслідок чого виходить навчений класифікатор або модель. Тобто модель = алгоритм + навчальна вибірка.

Перед тим як застосовувати модель на реальних даних, як правило, проводять тестування її продуктивності. Найбільш поширеним методом тестування є так звана перехресна перевірка (cross-validation). Її суть полягає в тому, що навчальна вибірка ділиться на n частин. Класифікатор навчається на (n-1) частинах, а одну частину використовують безпосередньо для перевірки результату. Ця процедура повторюється n

разів, а результатом вважається середнє арифметичне результатів окремих тестів.

Алгоритми конструювання дерев рішень складаються з етапів «побудова» або «створення» дерева (*tree building*) і «скорочення» дерева (*tree pruning*). У процесі створення дерева вирішуються питання вибору критерію розгалуження і зупинки навчання (якщо це передбачено алгоритмом). Під час етапу скорочення дерева вирішується питання відсікання деяких його гілок.

10.4.1 Критерій розгалуження

Процес створення дерева відбувається зверху вниз, тобто є низхідним. Під час процесу алгоритм має знайти такий критерій розгалуження (критерій розбиття), щоб розбити множину на підмножини, які б асоціювалися з цим вузлом перевірки. Кожен вузол перевірки має бути позначений певним атрибутом. Існує правило вибору атрибута: він має розбивати вихідну множину даних таким чином, щоб об'єкти підмножин, які утворилися внаслідок цього розбиття, були представниками одного класу або ж були максимально наближені до такого розбиття. Кількість об'єктів з інших класів, так званих «домішків», в кожному класі має прямувати до мінімуму.

Існують різні критерії розгалуження. Найбільш відомі – міра ентропії та індекс *Gini*.

В деяких методах для вибору атрибута розгалуження використовується так звана міра інформативності підпросторів атрибутів, яка ґрунтується на ентропійному підході і відома під назвою «міра інформаційного виграшу» (*information gain measure*) або міра ентропії.

Ентропія – широко використовуваний в природничих і точних науках термін. В широкому сенсі, в якому слово часто вживається в побуті, ентропія означає міру невпорядкованості системи; чим менше елементи системи підпорядковані якомусь порядку, тим вища ентропія.

Інший критерій розгалуження реалізовано в алгоритмі CART і називається індексом *Gini*. За допомогою цього індексу атрибут вибирається на підставі відстаней між розподілом класів. В основі всіх алгоритмів класифікації лежить поняття відстані. Найпростіше – це подати у вигляді географічної проблеми.

Припустимо, що нам потрібно об'єднати точки на карті – будинки, ресторани, стоянки і т. д. – в міста. Водночас все, що у нас є – це координати окремих точок, але ніякої інформації про адміністративний поділ. Людина візуально може вирішити цю задачу елементарно – будь-яка група точок, відстань між якими менша за відстань до іншої групи, і буде вважатися містом. Після розуміння цього, все, що залишається зробити, – це вибрати міру відстані і знайти ефективний алгоритм.

Якщо дано множину *T*, що містить приклади з *n* класів, індекс *Gini*, тобто *gini*(T), визначається за формулою:

$$gini(T) = 1 - \sum_{j=1}^{n} p_j^2 , \qquad (10.1)$$

де T – поточний вузол,

p_j – імовірність класу j у вузлі T,

n – кількість класів.

10.4.2 Обмеження на розмір дерева

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

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

Який розмір дерева може вважатися оптимальним? Дерево має бути досить складним, щоб враховувати інформацію з досліджуваного набору даних, але водночас воно має бути досить простим. Іншими словами, дерево має використовувати інформацію, що покращує якість моделі, і ігнорувати ту інформацію, яка її не покращує.

Тут існує дві можливі стратегії. Перша полягає в нарощуванні дерева до певного розміру відповідно до параметрів, заданих користувачем. Визначення цих параметрів може ґрунтуватися на досвіді та інтуїції аналітика, а також на деяких «діагностичних повідомленнях» системи, яка конструює дерево рішень.

Друга стратегія полягає у використанні набору процедур, що визначають «відповідний розмір» дерева.

Процедури, які використовують для запобігання створенню надмірно великих дерев, містить: скорочення дерева шляхом відсікання гілок; використання правил зупинки навчання.

Не всі алгоритми під час конструювання дерева працюють за однією схемою. Деякі алгоритми використовують два окремих послідовних етапи: побудова дерева і його скорочення; інші чергують ці етапи в процесі своєї роботи для запобігання нарощуванню внутрішніх вузлів.

10.4.3 Зупинка побудови дерева

Розглянемо правило зупинки. Воно має визначити, чи є розглянутий вузол внутрішнім вузлом, тоді він буде розбиватися далі, або ж він є кінцевим вузлом, тобто вузлом рішення.

Зупинка – це момент у процесі побудови дерева, коли потрібно припинити подальше розгалуження.

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

Другий варіант зупинки навчання – обмеження глибини дерева. У цьому випадку побудова закінчується, якщо досягнуто заданої глибини.

Ще один варіант зупинки – задання мінімальної кількості прикладів, які будуть міститися в кінцевих вузлах дерева. За такого варіанта розгалуження тривають до того моменту, поки всі кінцеві вузли дерева не будуть чистими або будуть містити не більше ніж задане число об'єктів.

10.4.4 Скорочення дерева або відсікання гілок

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

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

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

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

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

Якщо таке усічене дерево буде не інтуїтивним і складним для розуміння, використовують формування правил, які об'єднують в набори для опису класів. Кожен шлях від кореня дерева до його вершини або листка дає одне правило. Умовами правила є перевірки на внутрішніх вузлах дерева.

10.4.5 Переваги дерев рішень

Однією з основних переваг використання дерева рішень для задач класифікації є те, що така модель прийняття того чи іншого рішення із переліку класів можливих рішень **спрощує розуміння** того, чому об’єкт відноситься до вибраного класу. Наприклад, цілком зрозуміло чому в наведеному прикладі дерева рішень «Грати в теніс» приймається рішення

«Так», якщо виконуються умови «Погода=Сонячно» і «Вологість+нормальна».

Дерева рішень, на відміну від статистичних підходів, дозволяють працювати як з числовими, так і категоріальними даними.

Дуже важливою перевагою використання моделі дерева для прийняття рішень є їх можливість застосування у випадках, коли знання про дані важко формалізувати.

Розроблені алгоритми побудови дерев рішень не потребують попереднього вибору ознак (атрибутів) об'єктів. На вхід алгоритму подаються всі визначені ознаки, а алгоритм вибирає необхідні самостійно, що позитивно його відрізняє від алгоритмів навчання нейронних мереж, де ознаки мають бути попередньо визначені дослідником.

Ще однією з важливих переваг алгоритмів навчання моделей дерев рішень є можливість працювати з пропущеними значеннями атрибутів.

Алгоритми навчання моделей класифікації за допомогою дерев рішень набагато швидші за інші методи машинного навчання, наприклад, навчання нейронних мереж.

До основних напрямків застосування моделей у вигляді дерев прийняття рішень є такі задачі:

- ✓ оцінення кредитоспроможності у банківській справі;
- ✓ контроль за якістю продукції в промисловості;
- ✓ діагностика захворювань в медицині;
- ✓ аналіз будови сполучень в молекулярній біології і багато інших.

10.5 Основні алгоритми навчання дерев рішень

ID3 (Iterative Dichotomiser 3) **–** це алгоритм, розроблений Росом Куінланом, який використовується для генерації дерев рішень у машинному навчанні з деякого набору даних. ID3 є попередником більш досконалого алгоритму C4.5 та зазвичай використовується в галузях машинного навчання і обробки природної мови. В основі цього алгоритму лежить поняття інформаційної ентропії, тобто міри невизначеності інформації (оберненої до міри інформаційної корисності величини). Для визначення чергового атрибуту (ознаки) алгоритм підраховує ентропію всіх невикористаних до поточного кроку ознак, що використовуються для опису прецедентів навчальної вибірки і вибирає той атрибут, для якого ентропія мінімальна. Цей атрибут і вважатиметься найбільш інформативною ознакою на цьому кроці класифікації.

Кореневий вузол дерева, що будується алгоритмом, на початку роботи містить всю початкову множину даних навчальної вибірки. На кожній ітерації алгоритм перебирає усі невикористані атрибути множини та обчислює ентропію (або інформаційний приріст) цих атрибутів, після

чого він вибирає атрибут з найменшим значенням ентропії. Цей атрибут характеризує найбільшу вірогідність появи прецедентів вибірки з його значенням. Далі за вибраним атрибутом проводиться розбиття множини прецедентів для отримання підмножин даних. Як приклад можна назвати розбиття множини всього населення (кореневий вузол дерева) за атрибутом «Вік» на вузли-нащадки, що містять підмножини населення за значенням цього атрибуту «Вік» < 50 років, 50 років< «Вік» <70 років і «Вік» > 70 років. Алгоритм продовжує рекурсивно виконуватись на кожній підмножині, враховуючи лише ті атрибути, які раніше не були вибрані.

Рекурсія на підмножині може зупинитися в одному з таких випадків:

- кожен елемент у підмножині належить до одного класу; в цьому випадку вузол перетворюється в листковий вузол і позначається класом прикладів;

- більше немає атрибутів для вибору, але приклади ще не належать до одного класу. У цьому випадку вузол стає листковим вузлом і позначається найпоширенішим класом прикладів у підмножині.

Математичні засади побудови оптимальної стратегії прийняття рішень з використанням алгоритмів такого типу розроблено авторами в [8.3, 8.4].

C4.5. Цей алгоритм – удосконалення попереднього методу, що дозволяє, зокрема, «усікати» гілки дерева, якщо воно занадто сильно «розростається», а також працювати не тільки з атрибутами-категоріями, а й з числовими параметрами.

Вхідними даними для алгоритму C4.5 є навчальна вибірка $\Omega=\{x_1, x_2,\ldots, x_i,\ldots, x_m\}$ маркованих даних, в якій кожен прецедент x_i є n-розмірним зображенням елемента навчальної вибірки у вигляді вектора атрибутів (ознак) $x_i=(x_{i1}, x_{i2},\ldots, x_{in})$. Алгоритм C4.5, так само як і алгоритм ID3, будує дерево рішень, використовуючи критерій. Як умову розбиття прецедентів навчальної вибірки в кожному вузлі вибирається атрибут даних, який здійснює таке розбиття на підкласи, що мінімізує ентропію даних, тобто дає максимальний приріст інформації на цьому кроці.

Дерева рішень, сформовані за допомогою C4.5, можуть бути використані для класифікації, через що цей алгоритм почасти називають статистичним класифікатором. Алгоритм виконується за тим самим принципом, що і його попередник; відмінність полягає в можливості розбиття області значень незалежної числової змінної на кілька інтервалів, кожен з яких буде атрибутом. Відповідно до цього вихідна множина ділиться на підмножини. Зрештою, якщо дерево виходить занадто великим, можливе обернене групування – кількох вузлів в один листок. Водночас, оскільки перед побудовою дерева помилка класифікації вже врахована, вона не збільшується.

Комерційни варіант C5.0 цього алгоритму, розроблений Куінленом, має ряд переваг над алгоритмом C4.5, а саме:

- більшу швидкість роботи;
- отримані внаслідок його роботи дерева мають менший розмір;
- вищу точність роботи за рахунок використання принципу підсилення;
- використовує додаткову опцію winnowing (просіювання) для скорочення кількості малоінформативних атрибутів.

Алгоритм CART розроблений в 1984 році авторами Лео Брейманом, Джеромом Фрідманом, Річардом Олшеном і Чарльзом Стоуном в роботі «CART: Classification and regression trees». Знайшов широке застосування в таких галузях, як електротехніка, інженерія, біологія, медичні дослідження та фінанси. Алгоритм розроблено з метою побудови так званих бінарних дерев рішень – тобто тих дерев, кожен вузол яких при розбитті «дає» тільки двох нащадків. Грубо кажучи, алгоритм діє шляхом поділу на кожному кроці множини прецедентів рівно навпіл – по одній гілці йдуть ті зразки, в яких правило виконується (правий нащадок), за іншою – ті, в яких правило не виконується (лівий нащадок). Таким чином, в процесі «зростання» на кожному вузлі дерева алгоритм проводить перебір всіх атрибутів, і для наступного розбиття вибирає той, який максимізує значення показника, обчислюваного за математичною формулою і залежного від відношення числа прикладів у правому та лівому нащадках до загального числа прикладів.

Алгоритм CART, на відміну від C4.5, не використовує внутрішній, побудований на властивостях навчальної вибірки даних критерій ефективності для вибору дерева. Натомість ефективність дерева і його вибір відбувається на основі оцінки результатів незалежних тестів або за допомогою перехресної перевірки. У випадках, коли тестування або перехресна перевірка не проводиться, алгоритм не здатний визначити, яке дерево в послідовності є найкращим..

Механізм CART може містити автоматичне балансування класів і автоматичну обробку відсутніх значень, а також дозволяє здійснювати ефективне навчання, динамічну побудову функцій і оцінення дерева ймовірностей. Також в своїй роботі автори CART показали, як можна використовувати перехресну перевірку для оцінення ефективності кожного дерева, отриманого внаслідок обрізання, враховуючи, що дерева в різних складках перехресної перевірки можуть не вирівнюватися за кількістю кінцевих вузлів.

Алгоритми побудови дерев рішень розрізняються такими характеристиками:

Вид розгалуження – бінарне (binary), множинне (multi-way).

Критерії розгалуження – ентропія, Gini, інші.

Можливість обробки пропущених значень.

Процедура скорочення гілок або відсікання.

Можливості витягування правил з дерев.

Жоден алгоритм побудови дерева не можна апріорі вважати найкращим або досконалим, підтвердження доцільності використання конкретного алгоритму має бути перевірено і підтверджено експериментом.

Найбільш серйозна вимога, яка зараз висувається до алгоритмів конструювання дерев рішень – це масштабованість до різних розмірів вибірок даних.

Дерева прийняття рішень – один з основних і найбільш популярних методів допомоги у прийнятті рішень. Побудова дерев рішень дозволяє розібратися в структурі даних, створити працюючу модель класифікації даних, зокрема і для великих масивів даних Big Data.

10.6 Математичні засади побудови оптимальної стратегії прийняття рішень з використанням дерева рішень

10.6.1 Інформаційне оцінення ефективності алгоритмів навчання

Однією з найпоширеніших метрик ***P*** оцінки ефективності алгоритму навчання є міра ентропії, яка характеризує інформативність підпросторів вибраних ознак (атрибутів), що використовуються для формування умов розгалуження дерева на основі предикатних виразів у відповідних вузлах дерева. Під предикатом розуміють твердження, істинність якого залежить від значення змінних, що входять в нього. Якщо в нього підставити конкретні значення змінних, то предикат перетворюється в логічне висловлення, для якого можна зробити висновок про його істинність чи хибність. Розглянемо основні математичні означення, які використовуються в процесі використання інформаційної метрики.

Кількість інформації про деяку систему – це число, яке характеризує різноманітність можливих станів відображуваної системи. Під *мірою інформації* розуміють деяку неперервну невід'ємну адитивну функцію, визначену на множині станів.

Найбільшого поширення отримала *адитивна* двійкова міра Хартлі.

Нехай N – кількість повідомлень, n – довжина повідомлення, а m – кількість елементів деякого коду, використовуваного для передачі повідомлень. Очевидно, що $N = m^n$. Число N, а отже, і кількість інформації перебувають в експоненціальній залежності від довжини n повідомлення. Тому N незручно використовувати як міру кількості інформації. Для того, щоб міра мала властивість адитивності, вона має бути пропорційною довжині повідомлення і дозволяти складати кількості інформації низки окремих джерел. Для цього як міру кількості інформації Р. Хартлі запропонував використовувати величину I, обчислювану співвідношенням $I = \log N = n \log m$ (основа логарифма залежить від обраної одиниці кількості інформації).

Кількість інформації, що припадає на один елемент повідомлення (знак, літеру), називається ентропією:

$$H = \frac{I}{n} = \frac{n \log m}{n} = \log m. \qquad (10.2)$$

Якщо для кодування інформації використовувати двійкові символи 0 і 1, то $m = 2$ і $I = \log 2^n = n \log 2$. У цьому випадку зручно як основу логарифма взяти число 2, тоді $I = \log_2 2^n = n$. В такому разі *одиницю кількості інформації на один елемент* повідомлення називають *двійковою одиницею* або *бітом*. За такої умови одиниця невизначеності (двійкова одиниця чи *біт*) являє собою невизначеність вибору з двох рівноймовірних подій (bit – скорочення від англ. *binary digit* – двійкова одиниця). Ясно, що 1 біт – це кількість інформації, яким характеризується один двійковий елемент за рівноймовірних станів 0 і 1. Двійкове повідомлення довжини *n* містить *n* бітів інформації. Одиниця кількості інформації, що дорівнює 8 бітам, називається *байтом*.

Приклад 10.1. Яку кількість інформації можна отримати під час вибору однієї з 8 рівноймовірних літер?

Розв'язання. Передається 8 повідомлень, тобто *N*=8. Кількість отриманої інформації визначається різницею між початковою ентропією H_a (невизначеністю) та остаточною H_p ентропією

$$I = H_a - H_p = \log_2 8 - 0 = \log_2 2^3 = 3.$$

Отримана інформація є фактично довжиною повідомлення за умови двійкового кодування літери. Дійсно, 8 різних літер можна закодувати шляхом використання таких комбінацій двійкових символів: 000, 001, 010, 011, 100, 101, 110, 111.

Приклад 10.2. Текст, в якому використовується 32 літери алфавіту, передається за допомогою телетайпа в двійковому коді (тобто, використовуючи тільки символи 0 і 1). Визначити кількість інформації, що припадає на одну літеру.

Розв'язання. Кількість повідомлень (літер в тексті) $N = 32$. Отже, $I = \log_2 32 = \log_2 2^5 = 5$ біт/літеру.

Дійсно, на практиці кодування текстових повідомлень для їх передачі з допомогою телетайпа здійснюється кодом Бодо, в якому кожна літера кодується 5 двійковими символами.

Формула Хартлі відволікається від семантичних і якісних, індивідуальних властивостей станів системи, вона відтворює тільки її структурне різноманіття. Це основна і позитивна сторона формули. На жаль, міра Р. Хартлі не враховує того факту, що імовірності повідомлень можуть бути різними. Тому вона використовується лише у випадку рівноймовірних подій. Для нерівноймовірних подій невизначеність менша.

Наприклад, невизначеність вибору у випадку двох елементів з апріорними ймовірностями 0,9 і 0,1 менша, ніж у випадку елементів з однаковими ймовірностями (0,5; 0,5). Тому природною є вимога, щоб міра невизначеності була безперервною функцією ймовірностей елементів. Цю вимогу задовольняє *статистична міра*, яка використовує імовірнісний підхід до оцінення кількості інформації. Згідно з Шенноном кожне повідомлення характеризується імовірністю його появи, і чим вона менша, тим більше в повідомленні інформації. Ймовірність конкретних типів повідомлень встановлюють на основі статистичного аналізу.

Нехай повідомлення утворюються послідовною передачею літер деякого алфавіту $x_1,..,x_i,...,x_q$ з імовірністю появи кожної літери $p(x_1)=p_1,...,p(x_i)=p_i,...,p(x_q)=p_q$, виконується умова: $\sum_{i=1}^{q} p_i = 1$.

Множину елементів з відомим розподілом їхніх імовірностей називають ансамблем. Згідно з Шенноном кількість інформації, яка міститься в повідомленні x_i, розраховують за формулою

$$I(x_i) = log_a \frac{1}{p_i}. \qquad (10.3)$$

Якщо з'являється повідомлення про подію, яка часто зустрічається, то таке повідомлення для одержувача є малоінформативним. Настільки ж малоінформативні повідомлення про події, ймовірність появи яких близька до нуля.

Якщо має місце складна подія, що полягає, наприклад, в появі двох незалежних подій x_i і x_j з ймовірностями $p(x_i)$ і $p(x_j)$, відповідно, то її імовірність, як відомо, можна визначити за формулою $p = p(x_i) \cdot p(x_j)$. Тоді кількість інформації в повідомленні, що описує сукупність подій x_i і x_j, дорівнює:

$$I(x_i, x_j) = \log_a \frac{1}{p(x_i)p(x_j)} = \log_a \frac{1}{p(x_i)} + \log_a \frac{1}{p(x_j)} = I(x_i) + I(x_j). \qquad (10.4)$$

Нехай тепер складне повідомлення характеризується алфавітом літер $X=\{x_1, x_2, ..., x_q\}$, їхніми імовірностями $\{p_1, p_2, ..., p_q\}$ і частотою появи кожної літери $\{m_1, m_2, ..., m_q\}$. Вважаючи, що всі повідомлення статистично незалежні, матимемо, що загальна кількість появ літер буде дорівнювати сумі появ кожної з них $m_1 + m_2 + ... + m_q = m$. Загальна кількість інформації для всіх q типів повідомлень з урахуванням виразу (10.3)

$$I_\Sigma = \sum_{i=1}^{q} m_i \log \frac{1}{p_i}. \qquad (10.5)$$

Кількість інформації та невизначеність (ентропію) для всієї сукупності випадкових повідомлень отримують, взявши середню величину за всіма випадками. Середнє значення кількості інформації на одне повідомлення (ентропія) згідно з формулою Шеннона

$$I(X) = \frac{I_\Sigma}{m} = \sum_{i=1}^{q} \frac{m_i}{m} \log \frac{1}{p_i} = \sum_{i=1}^{q} p_i \log \frac{1}{p_i} = -\sum_{i=1}^{q} p_i \log p_i , \qquad (10.6)$$

$$H(X) = \frac{H_\Sigma}{m} = \sum_{i=1}^{q} \frac{m_i}{m} \log \frac{1}{p_i} = \sum_{i=1}^{q} p_i \log \frac{1}{p_i} = -\sum_{i=1}^{q} p_i \log p_i , \qquad (10.7)$$

де m_i / m характеризує ймовірність p_i, якщо m достатньо велике. Вираз $\log \frac{1}{p_i}$ розглядають як часткову ентропію, яка характеризує інформативність окремої літери, а ентропію H – як середнє значення часткових ентропій. У разі малих значень p_i часткова ентропія велика, а з наближенням p_i до одиниці вона наближається до нуля.

Хоча вирази (10.6) і (10.7) для $I(X)$ і $H(X)$ збігаються, поняття «кількість інформації» і «ентропія» принципово різні. Ентропія $H(X)$, що виражає середню невизначеність сукупності подій, може бути обчислена апріорно, тобто до отримання повідомлення про них. Величина ж $I(X)$ визначається апостеріорно, тобто після отримання повідомлень. $H(X)$ є мірою нестачі інформації про стан системи. З надходженням інформації про стан системи ентропія останньої знижується. Збіг виразів для $I(X)$ і $H(X)$ свідчить лише про те, що кількість одержуваної інформації дорівнює чисельно ентропії, яка мала місце відносно джерела повідомлень. Згідно з відомим діалектичним законом єдності і боротьби протилежностей інформація розглядається як міра знищення, тобто зняття ентропії.

Кількість інформації тільки тоді дорівнює ентропії, коли невизначеність ситуації знімається повністю. Найбільшу кількість інформації отримують тоді, коли вірогідність всіх N повідомлень однакова, тобто $p_i = 1/N$ для всіх $i = \overline{1, N}$. Згідно з Хартлі ця максимальна кількість інформації оцінюється як

$$I = -\sum_{i=1}^{N} p_i \log p_i = -\sum_{i=1}^{N} \frac{1}{N} \log_2 \frac{1}{N} = \log_2 N . \qquad (10.8)$$

Таким чином, міра Р. Хартлі – окремий випадок міри К. Шеннона для рівноймовірних елементів. Можна також показати, що міра Шеннона є узагальненням міри Хартлі на випадок нерівноймовірних елементів.

Приклад 10.3. Знайти кількість бітів інформації, яку буде містити повідомлення про результат кидання грального кубика за двійкового кодування.

Розв’язування.

Вірогідність випадання будь-якої грані кубика однакова і дорівнює $p = 1/6$, а число станів $N = 6$. Тоді повідомлення про результат кидання кубика за умови двійкового кодування (основа логарифма 2) буде містити, згідно з (10.8) , $I = \log_2 6 = 2.58$ бітів інформації.

10.6.2 Вибір критеріїв ефективності систем прийняття рішень

Задача побудови оптимальної стратегії для систем прийняття рішень і управлінння, а відповідно і методів та алгоритмів оцінення їх ефективності, в нетривіальній постановці має пов'язуватися з ефективністю роботи як об'єкта управління, так і з умовою досягнення заданого показника якості роботи самих цих систем. Відомо багато робіт, присвячених розв'язанню цієї задачі окремо для власне систем прийняття рішень (наприклад, розпізнавання та ідентифікація образів) [1], і для систем управління. Формалізація процедури побудови ефективної стратегії прийняття рішень можлива тільки в тому випадку, коли попередньо неформальними методами вибрано апріорний алфавіт можливих станів керованих об´єктів, а також вибрано критерії для оцінення ефективності систем прийняття рішень. Тоді формальна постановка задачі побудови оптимальної стратегії прийняття рішень може бути сформульована як задача пошуку оптимального за загальносистемним критерієм дерева рішень.

Формування адекватних критеріїв ефективності можна здійснити на основі рекомендацій, що відображають надбаний досвід системного проектування та дослідження операцій. Правильно вибрані критерії ефективності мають задовольняти низку вимог, основними з яких є:

- відповідність цільовому призначенню системи;
- критичність до параметрів, що визначають якість функціонування системи;
- достатньо проста обчислюваність;
- нормованість;
- універсальність (можливість використання критерію для порівняльного оцінення будь-якої пари систем з деякого класу систем).

Розглянемо проблему побудови моделі оптимального дерева прийняття рішень на прикладі задачі оптимізації стратегії розпізнавання мови.

Особливістю систем автоматичного розпізнавання мови (САРМ), як і інших складних систем, є необхідність їх оцінення за багатьма частковими показниками якості, з яких найбільш часто використовуваними є такі:

1) E – точність розпізнавання;

2) t_r – час розпізнавання;

3) О – обсяг потрібної пам'яті;

4) τ_r – час приведення системи в готовність;

5) С – вартість системи.

Показники 2, 3, 4 з визначеними вагами входять в критерій вартості С, тому їх можна вилучити зі списку частинних критеріїв. Таким чином, основними частинними критеріями для оцінення ефективності САРМ є точність розпізнавання та вартість. Використання їх затруднюється тими обставинами, що різні системи вирішують різні задачі і працюють в різних умовах. Це не дозволяє провести порівняльний аналіз якості САРМ, що вирішують різні задачі розпізнавання. Тому в роботі як загальносистемний критерій використовується узагальнений функціонально-статистичний критерій, модифікований належним чином до системи мовного спілкування шляхом належного вибору потенціальної і реальної САРМ [1]:

$$E = \frac{E_{\text{Р}}}{E_{\text{П}}}\Bigg|_{\text{E} = \text{E}_{\partial}}, \qquad (10.9)$$

де $E_{\text{Р}}$ і $E_{\text{П}}$ – функціонально-статистичні критерії для реальної і потенціальної САРМ, відповідно;

$$E_{\text{Р}} = \frac{\text{I}_{\text{Р}}}{C_{\text{Р}}},\ E_{\text{П}} = \frac{\text{I}_{\text{П}}}{C_{\text{П}}}, \qquad (10.10),\ (10.11)$$

E_{∂} – задана в технічному завданні точність розпізнавання;

$\text{I}_{\text{Р}}, \text{I}_{\text{П}}$ – кількість інформації, яку дістає реальна і потенціальна системи, відповідно;

$C_{\text{Р}}, C_{\text{П}}$ – вартість реальної і потенціальної систем, відповідно.

За модель потенційної системи вибрано модель слухової системи, оскільки вона використовує найбільш ефективну систему ознак і оптимальні способи їх зображення, а також оптимальні процедури розпізнавання. Слухова система є ідеальною метою функціонування декодера інформації з мовного сигналу для заданого словника і рівня навколишніх завад. Кількість інформації, яку дістає потенціальна система розпізнавання, дорівнює:

$$\text{I}_{\text{П}} = \text{H}_{\text{П}}(W) - \text{H}_{\text{П}}(W / W^*), \qquad (10.12)$$

де $\text{H}_{\text{П}}(W)$ – ентропія повідомлення до його передавання, пов'язана з апріорною імовірністю окремого повідомлення $P(w_i)$ таким виразом:

$$\text{H}_{\text{П}}(W) = \sum_{i=1}^{|W|} P(w_i) * \log_2 P(w_i). \qquad (10.13)$$

Апріорна ймовірність $P(w_i)$ визначається або статистичним шляхом, або з припущення однакової ймовірності повідомлень, тоді

$$P(w_i) = \frac{1}{|W|}\ , \qquad (10.14)$$

де $|W|$ – кількість повідомлень в словнику W;

$\mathrm{H}_{\Pi}(W/W^*)$ – ентропія повідомлення після його прийому, яка характеризує втрату інформації на одно повідомлення:

$$\mathrm{H}_{\Pi}(W/W^*) = -\sum_{w_i=W}\sum_{w_j=W} p(w_i/w_j) * p(w_j) * \log_2 \frac{p(w_i) * p(w_i/w_j)}{\sum_{w_j} p(w_i/w_j) * p(w_i)}. \qquad (10.15)$$

Величини $p(w_i/w_j)$, що характеризують подібність слів, можуть бути визначені експериментально для різних рівнів завад за результатами розпізнавання слів і фонем людиною.

На етапі проектування за вартість C_{P} використовується величина, що пов'язана пропорціональною залежністю зі складністю процесу розпізнавання:

$$C_{\mathrm{P}} = C_{\mathrm{X}} + C_{\mathrm{K}}\ , \qquad (10.16)$$

де C_{X} – складність обчислення ознакового опису елементів мови;

C_{K} – складність обчислень класифікації образів.

Критерій (10.9) задовольняє всі вимоги, що висуваються до показників якості роботи систем.

10.6.3 Розробка процедури управління вибором оптимальних параметрів дерева рішень для задачі розпізнавання мови

Формальна постановка задачі побудови оптимальної стратегії розпізнавання може бути подана таким чином:

$$\hat{S}_{Gopt} = \arg u \max E(\hat{S}_{Gi})\Big|_{\left\{\hat{S}_{Gi} \in \hat{S}_G, W_d, r_d, E_d\right\}} \qquad (10.17)$$

де $\tilde{S}_{Gi}$ $\hat{S}_{Gi}$ – одна із стратегій розпізнавання із замкнутої відносно доступної інформації множини стратегій розпізнавання $\tilde{S}_G$ $\hat{S}_G$;

W_d, r_d, E_d – задані умовою задачі розпізнавання словник, рівень завад і точність розпізнавання, відповідно.

Складність вирішення задачі оптимізації, сформульованої у вигляді (10.17), полягає в тому, що обчислювальна складність C_p критерію ефективності (10.9) є функцією двох змінних – обчислювальних затрат на процедуру класифікації C_k та затрат на обчислення ознакового опису C_x. Декомпозицію цих змінних в критерії (10.9) можна здійснити шляхом зображення стратегії розпізнавання в вигляді дерева покрокової процедури класифікації [1].

Використання послідовно-паралельної стратегії розпізнавання у вигляді дерева рішень в оптимізаційній процедурі побудови ефективної стратегії розпізнавання мови не виключає узагальненого підходу, оскільки часто використовувані алгоритми статистичного розпізнавання в *n*-вимірному просторі ознак можна подати у вигляді дерева рішень, в якому є один кореневий вузол, а всі інші вузли – термінальні. Кількість гілок такого дерева (кроків класифікації) дорівнює кількості термінальних вузлів (образів, що розпізнаються).

З метою знаходження принципів побудови процедури пошуку оптимального дерева класифікації необхідно визначити його властивості. Для того, щоб пояснити терміни, які в подальшому будуть використовуватись, наведемо приклад довільного дерева рішень, зображеного на рисунку 10.2.

Зображене дерево $T_r = (\Omega, G)$ містить множину вузлів (класів образів, на які здійснено розбиття множини образів на цьому рівні дерева) $\Omega = \{\Omega_i^h\}$ і множину гілок $G = \{g_{kj}\}$, де h – висота дерева, i – номер класу образів на висоті h, k – індекс вузла-попередника j-го потомка. Індекс k утворюється рекурсивною конкатенацією індексів всіх вузлів-попередників для того, щоб можна було однозначно ідентифікувати весь шлях рішень, який передував розпізнаванню образу, зображеного термінальним символом дерева.

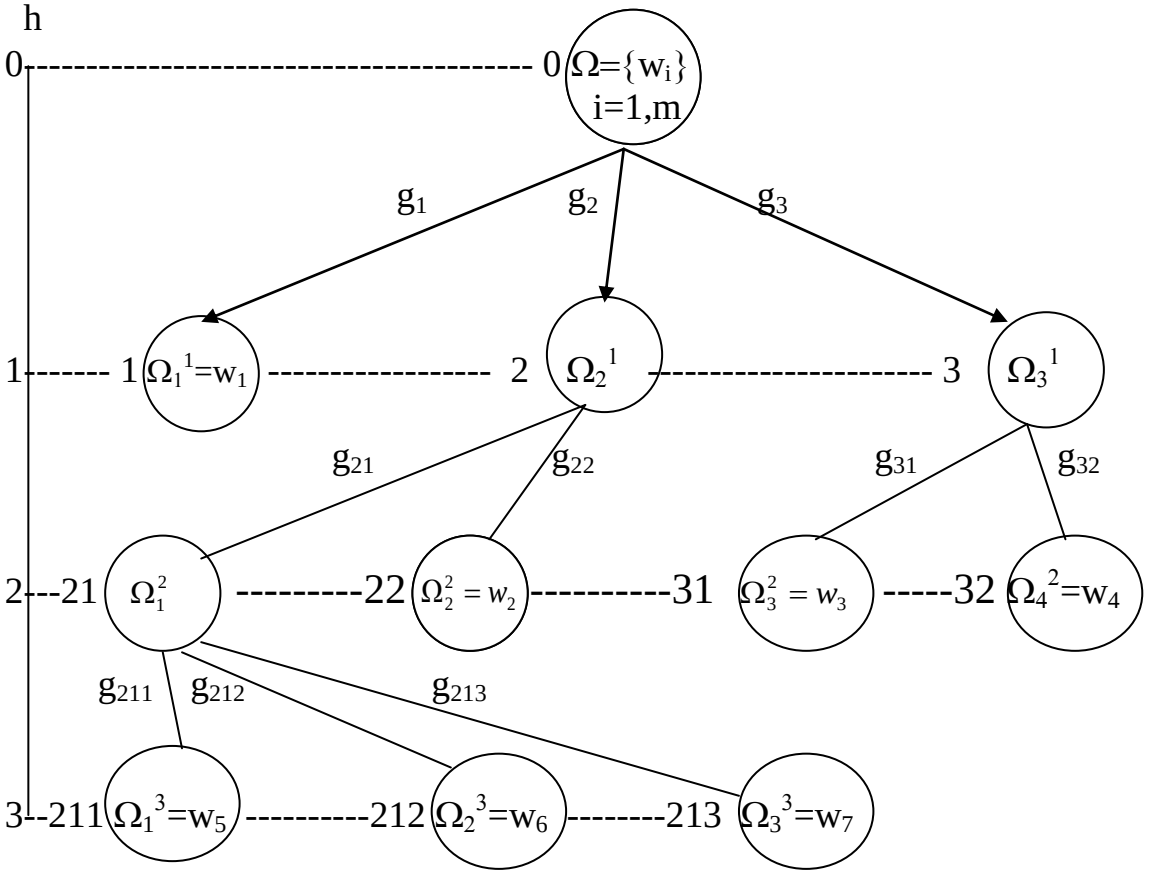


Рисунок 10.2 – Приклад дерева класифікації

Наприклад, для розпізнавання образу w_5, який зображений 1-им термінальним вузлом з індексом 211, необхідно прийняти рішення про класифікацію в вузлах з індексами 0, 2, 21. Дерево класифікації на рис. 10.2 містить сім термінальних вузлів $w_1, w_2, ..., w_7$ з індексами 1, 22, 31, 32, 211, 212, 213, відповідно, три внутрішніх вузли $\Omega_2^1, \Omega_3^1, \Omega_1^2$ з індексами 2, 3, і 21, відповідно, і один кореневий вузол Ω з індексом 0. Кореневий вузол відповідає словнику, який розпізнається.

Оскільки складність C_p системи розпізнавання є адитивною сумою складностей C_i кожного з ієрархічних рівнів розпізнавання, а інформативність I_p є неспадною функцією ймовірності правильного розпізнавання, то оптимальна стратегія є композицією алгоритмів розпізнавання, що максимізують відношення $\frac{\mathrm{I}_i}{C_i}$ на кожному з рівнів. Послідовність композиції алгоритмів в оптимальній стратегії має відповідати послідовності розміщення рівнів дерева класифікації, а ознаки на кожному рівні мають вибиратися з умови їх мінімальної складності:

$$\tilde{S}_{Gopt} = A_1(\tilde{I}(S^1)) \otimes A_2(\tilde{I}(S^2)) \otimes \cdots \otimes A_W(\tilde{I}(W\ )), \qquad (10.18)$$
$$\tilde{I}(S^i) = x^i_{opt}, x^i_{opt} = \arg u \min C_i(x^i_l)\ ,$$

де $\tilde{I}(W)$ – параметричний чи фонетичний опис еталонів слова;

$C_i(x^i_l)$ – обчислювальна складність l-ї ознаки, що використовується для опису елементів i-го рівня;

$A_h(\tilde{I}(S^h))$ – алгоритм попередньої класифікації словника на групи;

$\tilde{I}(S^h))$ – ознаковий опис звукотипів алфавіту S^h, що розпізнаються на цьому рівні ієрархії.

Візьмемо такі позначення для деякого дерева класифікації: C_A – число різних гілок дерева, пропорціональне часу класифікації і може чисельно відображати обчислювальну складність алгоритму класифікації; $\Omega = \{w_1, w_2, ...w_i, ...w_M\}$ – множина образів; t_A – середній час класифікації образу деревом рішень; $P(w_i)$ – апріорна ймовірність образу w_i. Крім цього, задамо деяке кодове дерево T_c, ізоморфне дереву рішень T_r. У цьому випадку число висячих вершин η_i кодового дерева дорівнює числу образів M, що розпізнаються, кількість його ребер m_c дорівнює кількості гілок дерева, $m_c = C_A$, середня довжина кодового дерева $\bar{l}$ відповідає середньому часу класифікації t_A, а ймовірності P_η висячих вершин кодового дерева дорівнюють апріорним ймовірностям $P(w_i)$ термінальних вузлів дерева рішень, що мають одинакові індекси з висячими вершинами кодового дерева.

Для деякого внутрішнього вузла Ω_i^h дерева, яке має K розпізнаваних класів, з якого виходять m вузлів-нащадків, зберігається умова адитивності ентропії:

$$H_K = H_m(P_1, P_2, ..., P_m) + \sum_{i=1}^{m} P_i * H_{ni}(q_{i1}, q_{i2}, ..., q_{in_i}), \qquad (10.19)$$

де n_i – число синів i-ого вузла-нащадка,

P_i – ймовірність цього вузла, $\sum_{i=1}^{m} P_i = 1$;

q_{ij} – ймовірність j-го сина i-го вузла-нащадка, $\sum_{j=1}^{n_i} q_{ij} = 1$, $i = \overline{1,m}$.

Формулу (10.19) неважко довести, якщо розглянути ентропію i-го вузла-нащадка. Ймовірності P_j його синів дорівнюють добутку $P_i * q_{ij}$ як ймовірність складної події, що складається з двох незалежних подій. Тоді ентропія, яка відповідає i-й гілці термінального вузла, дорівнює

$$H_i = P_i q_{i1} * \log_2(P_i q_{i1}) + P_i q_{i2} * \log_2(P_i q_{i2}) + \cdots + P_i q_{in_i} * \log_2(P_i q_{in_i}) =$$

$$P_i * \log_2 P_i \sum_{j=1}^{n_i} q_{ij} + P_i * \sum_{j=1}^{n_i} q_{ij} \log_2 q_{ij} = H(P_i) + P_i * H(q_{ij}).$$

Враховуючи, що термінальний вузол має m вузлів-нащадків, та підсумувавши ентропію по кожному i-ому вузлу, можна отримати (10.19).

На цьому співвідношенні грунтується така теорема:

Теорема 10.1. Якщо на будь-якому рівні h дерева T_r справедливі умови

$$\Omega_i^h \bigcap \Omega_j^h = \varnothing, \ \forall i \neq j \quad \text{i},$$

$$H^{(h)} = -\sum_{i=1}^{m_h} P_i^{(h)} * \log_2 P_i^{(h)} \geq H_0 \qquad (10.20)$$

для кожного внутрішнього вузла h-го рівня, то середній час $\bar{t}_A$ класифікації невідомого образу від кореневого вузла до термінального задовольняє таке співвідношення

$$\bar{t}_A \leq H_w / H_0 \ . \qquad (10.21)$$

В (10.20) і (10.21) прийнято такі позначення: H_0 – деяке порогове значення ентропії для вузлів дерева рішень; $P_i^{(h)}$ – ймовірність, пов'язана з вузлом Ω_i^h, $P_i^h = \sum_{j \subset \Omega_i^h} P_j$; $H_w = -\sum_{i=1}^{M} P(w_i) * \log_2 P(w_i)$ – ентропія множини розпізнаваних образів; $P(w_i)$ – апріорна ймовірність класу w_i; m_h – число гілок вузла Ω_i^h на рівні h.

Теорема 10.1 доводиться методом математичної індукції, беручи до уваги те, що кожний внутрішній вузол дерева зі своїми нащадками сам є деревом рішень (піддеревом).

Рішення проблеми належного вибору коефіцієнта розгалуження дозволяє значно звузити коло пошуку в оптимізаційній процедурі пошуку оптимального дерева рішень. Припустимо, що середня висота термінальних вузлів дорівнює $\bar{h}$, а середній коефіцієнт розгалуження $\bar{B}_r$. Тоді число різних шляхів M від кореня дерева до термінальних вузлів

$$M = \bar{B}_r^{\bar{h}},$$

звідки

$$B_r^{\bar{h}} = \sqrt[\bar{h}]{M}\ . \qquad (10.22)$$

З виразу (10.22) можна визначити, що значення H_0 в (10.20) наближено дорівнює

$$H_0 \approx \log_2(B_r).$$

Оскільки число вузлів, що виходять з деякого внутрішнього вузла, дорівнює наближено B_r, то з (10.21) можна записати, що середній час класифікації образу в дереві

$$\bar{t}_A = B_r^{\bar{h}} * \frac{H_w}{\log_2 \bar{B}_r}. \qquad (10.23)$$

Мінімізація (10.23) показує, що $\bar{t}_A$ мінімальний за $\bar{B}_r = 2.7182...$

Теорема 10.2. Якщо помилка e^h класифікації образу в кожному вузлі дерева на рівні h задовольняє умову $e^h \le e_0$, де e_0 – деяке граничне значення помилки, то загальна помилка класифікації E_d образу в дереві визначиться виразом

$$E_d = e_0 * \frac{H_w}{H_0}\ . \qquad (10.24)$$

Доведення: верхня границя помилки класифікації в дереві визначається як

$$E_0 = e_0 * \bar{t}_A,$$

звідки, з урахуванням (10.24), маємо $E_d \le E_0 = e_0 * \frac{H_w}{H_0}$, що і потрібно було довести. Оскільки

$$H_w \approx \log_2 M, \text{ а } H_0 = \log_2 \bar{B}_r,$$

то

$$E_d \le e_0 * \frac{\log_2 M}{\log_2 \bar{B}_r}. \qquad (10.25)$$

Мінімізація сумісно помилки класифікації (10.25) і часу класифікації (10.23) дає границі коефіцієнта розгалуження $\bar{B}_r$, що вибирається під час побудови оптимального дерева рішень:

$$2 \le B_r \le 5\ . \qquad (10.26)$$

Таким чином, встановлені властивості дерева рішень дозволяють звузити діапазон пошуку під час вирішення оптимізаційної задачі (10.17).

Вирішення задачі побудови ефективної стратегії розпізнавання у вигляді дерева класифікації можна здійснити за допомогою оптимізаційної процедури «керованого пошуку вперед з поверненням». В цій процедурі

критерій (10.9) керує пошуком такої структури дерева рішень серед всіх можливих, в якій на кожному кроці пошуку вибирається та конфігурація вузлів, яка має найвище значення критерію. Для заданого вузла Ω_i^h дерева процедура пошуку виконується в вигляді такої послідовності кроків:

1. На основі вибраної ознаки $x^h \in X$ здійснюється одне з можливих розбиттів $\pi^h \in \Pi$ вузла Ω_i^h на підмножину вузлів-нащадків $\{\Omega_j\}, j = \overline{1,m}$. Ознака x^h вибирається за «матрицею розрізнюваності» таким чином, щоб коефіцієнт розгалуження лежав в межах, визначених в (10.26).
2. Обчислюється значення критерію (10.9) для отриманої конфігурації вузла.
3. Повторюючи пункти 1 і 2, будують інші можливі розбиття і для них обчислюють значення критерію .
4. Визначають конфігурацію, для якої критерій має максимальне значення, і тим самим знаходять оптимальний набір ознак для цього вузла дерева і оптимальний крок алгоритму класифікації.

Як «матрицю розрізнюваності» використовують таблицю попарної розрізнюваності образів w_i і w_j за всіма ознаками апріорного алфавіту на основі вибраної в просторі ознак відстані d_{ij}.

На рис. 10.3 наведено деякі результати реалізації описаної поцедури.

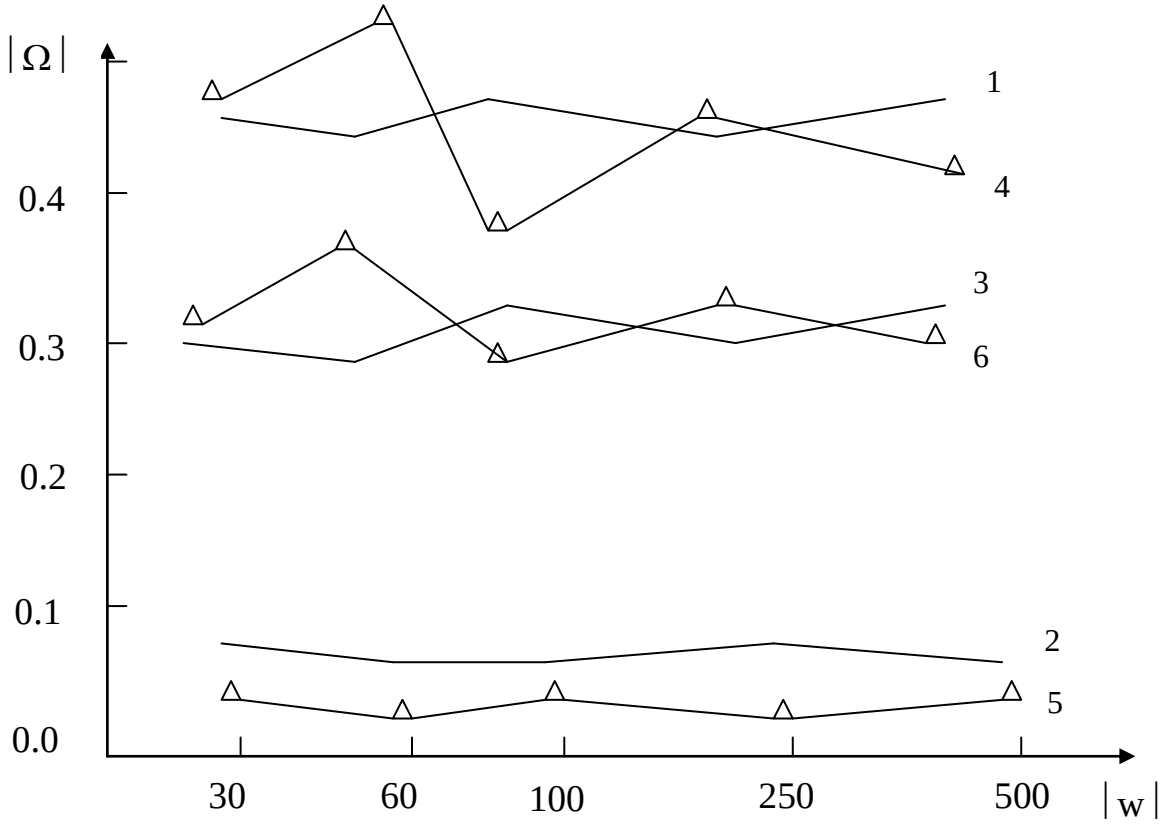


Рисунок 10.3 – максимальний, мінімальний і середній обсяги класів слів, що розрізняються за кількістю пауз (криві 1, 2, 3, відповідно) і кількістю складів (криві 4, 5, 6)

Вони показують, що простір пошуку кандидатів на ропізнавання зменшується на перших кроках класифікації приблизно в два – три рази на основі використання вказаних контекстно- і дикторонезалежних ознак.

Контрольні питання і завдання

1. Дайте означення дерева рішень.
2. Як формулюється процедура навчання моделі дерева рішень згідно з означенням Т. Мітчелла?
3. Для рішення яких найбільш популярних класів задач можна застосувати дерево рішень?
4. Назвіть приклади застосування моделі дерева рішень для прийняття рішень.
5. Які критерії застосовують для визначення ефективності алгоритму побудови дерева рішень?
6. Якими способами, окрім графічного, можна подати дерево рішень?
7. Назвіть найбільш поширені галузі застосування моделі дерева для прийняття рішень.
8. Які складові використовуються для побудови моделі дерева рішень?
9. Які методи застосовуються для тестування моделі дерева рішень?
10. Які етапи виконуються під час конструювання дерева рішень і які задачі вирішуються на кожному з цих етапів?
11. Які найбільш відомі критерії розгалуження використовуються під час побудови дерева?
12. Що таке індекс Gini і як він формулюється?
13. Які способи обмеження розміру дерева і зупинки процедури його побудови використовуються?
14. Назвіть основні відомі алгоритми навчання дерев рішень.
15. Назвіть одну з найпоширеніших метрик оцінення ефективності алгоритму навчання.
16. Дайте означення терміна «кількість інформації».
17. Що розуміють під поняттям «міра інформації»?
18. Як називається одиниця кількості інформації за використання двійкової системи числення?
19. Як називається одиниця кількості інформації, яка дорівнює 8 бітам?
20. Визначіть кількість інформації, яка припадає на одну літеру в алфавіті, що містить 16 символів.
21. Запишіть формулу Хартлі для визначення кількості інформації в повідомленні.
22. Що називається ентропією?
23. Запишіть формулу Шеннона для визначення кількості інформації і ентропії станів системи.

24. Сформулюйте основні вимоги, що висуваються до критеріїв оцінення ефективності прийняття рішень.

25. Які частинні показники якості використовуються під час оцінювання ефективності систем розпізнавання?

26. Що розуміють під потенціальною і реальною системами?

27. Як формулюється постановка задачі побудови оптимальної стратегії розпізнавання?

28. Який вигляд має оптимізаційна процедура побудови ефективної стратегії розпізнавання мови?

29. В яких межах змінюється коефіцієнт розгалуження в кожному вузлі для забезпечення одночасної мінімізації помилки прийняття рішень і кількості кроків в прийнятті рішень?

30. Опишіть послідовність кроків оптимізаційної процедури «керованого пошуку вперед з поверненням», яка використовується для вирішення задачі побудови ефективної стратегії розпізнавання образів (чи прийняття рішень) у вигляді дерева класифікації.

31. В якій послідовності використовуються ознаки образів в вузлах дерева класифікації для забезпечення оптимального результату розпізнавання?

РОЗДІЛ 11
НАВЧАННЯ З УЧИТЕЛЕМ. РЕГРЕСІЙНИЙ АНАЛІЗ

11.1 Матричне подання експериментальних даних

Багато об'єктів дослідження характеризуються множиною параметрів, і за результатами спостереження за їх функціонуванням формуються матриці вимірювань (експериментальних даних – ЕД)

$$\mathbf{X}=\begin{vmatrix} x_{11} & x_{12} & \dots & x_{1m} \\ x_{21} & x_{22} & \dots & x_{2m} \\ . & . & . & . \\ x_{n1} & x_{n2} & \dots & x_{nm} \end{vmatrix} \qquad (11.1)$$

Рядки такої матриці відповідають результатам реєстрації всіх вимірюваних параметрів об'єкта в одному експерименті, а стовпці містять результати спостережень за одним параметром (фактором, варіантом) у всіх експериментах. Позначимо кількість параметрів через m $(m>1)$, а кількість спостережень – через n.

У матриці **X** елемент x_{ij} відповідає значенню *j*-го варіанта в *i*-му спостереженні. Матриця в загальному випадку може містити порожні елементи, наприклад, через пропуски в реєстрації значень параметрів. У багатовимірному аналізі бажано усунути пропущені значення. Для цього існують спеціальні прийоми, зокрема, викреслювання відповідних рядків матриці або занесення середніх значень замість відсутніх. Надалі будемо вважати, що матриця не містить порожніх елементів, а параметри об'єкта характеризуються неперервними випадковими величинами.

Методи обробки матриці ЕД основані на такому припущенні: якщо об'єкт піддати новому обстеженню і отримати, взагалі кажучи, іншу матрицю даних, то після її обробки за допомогою тих самих методів будуть отримані результати, близькі до результатів обробки першої матриці. Це припущення грунтується на статистичній гіпотезі формування матриці ЕД. Матриця породжується випадковим чином відповідно до певної імовірнісної закономірності, а саме: в m-вимірному просторі параметрів існує деякий (нехай і невідомий) розподіл ймовірностей, і кожен рядок матриці з'являється згідно з цим розподілом незалежно від появи інших рядків.

Кожен стовпець матриці являє собою випадкову вибірку значень одного параметра об'єкта. Вказане припущення означає, по-перше, що оцінки моментів і параметрів розподілу, обчислені за вибіркою, будуть близькі до істинних значень, по-друге, значення неперервних функцій,

побудованих за цими оцінками, будуть близькі до значень функцій, побудованих за істинним значенням параметрів.

Таким чином, об'єктом дослідження в багатовимірному аналізі є багатовимірна випадкова величина, подана вибіркою певного обсягу.

Параметри, що характеризують об'єкт дослідження, мають різний фізичний зміст, і матриця даних істотно змінюється, якщо змінюються шкали, в яких вимірюються ті чи інші параметри. Матрицю даних ще до проведення аналізу доцільно привести до стандартного вигляду, тобто стандартизувати значення варіант (нагадаємо, що середнє значення стандартизованої варіанти дорівнює нулю, дисперсія – одиниці). Навіть коли всі варіанти вимірюються в одній шкалі, це перетворення дозволяє спростити подальші перетворення. Стандартизовану матрицю будемо позначати через ***U***. Для переходу від вихідної до стандартизованої матриці обчислюються оцінки математичного сподівання $\mu_1(x_j) = \frac{1}{n}\sum_{i-1}^{n} x_{ij}$ і дисперсії $\mu_2(x_j) = \sigma_{x_j}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_{ij} - \mu_1(x_j))^2$ кожної варіанти, $j = \overline{1,m}$ та елементи стандартизованої матриці $u_{ij} = (x_{ij} - \mu_1(x_j)) / \sigma(x_j)$, $i = \overline{1,n}$, $j = \overline{1,m}$.

Елементи матриці ***U*** є безрозмірними величинами. Матриця ***U*** є об'єктом обробки в таких задачах, як кореляційний аналіз, побудова регресійних моделей, ідентифікація моделей статичних систем, виділення інформативних ознак розпізнаваних образів тощо.

В багатьох напрямках прикладної діяльності людини породжується велика кількість експериментальних даних, аналіз яких потребує розв’язання задачі їх узагальнення. Наприклад, для цього часто потрібно віднайти теоретичну залежність, яка б дозволила їх описати і зберігати в стиснутому аналітичному вигляді. Таку задачу можна розв’язати, застосувавши один із двох підходів до узагальнення експериментальних даних: інтерполяція або апроксимація. Інтерполяція знаходить модель аналітичної лінії (поверхні для багатовимірного випадку), яка точно збігається з експериментальними даними. Інтерполяцію застосовують для детермінованого випадку для даних без шумів. В більшості практичних задач експериментальні дані описуються випадковими величинами, тому для них застосовують апроксимацію, яка знаходить модель у вигляді лінії чи поверхні, що наближають середні значення експериментальні даних. В цьому випадку говорять про регресійний аналіз даних.

11.2 Регрессійний аналіз

11.2.1 Постановка задачі

Однією з типових задач обробки багатовимірних ЕД є визначення кількісної залежності показників якості об'єкта від значень його параметрів і характеристик зовнішнього середовища. Прикладом такої постановки

завдання є встановлення залежності між часом обробки запитів до бази даних і інтенсивністю вхідного потоку. Час обробки залежить від багатьох факторів, зокрема від розміщення шуканої інформації на зовнішніх носіях, складності запиту. Отже, час обробки конкретного запиту можна вважати випадковою величиною. Але, разом з тим, за умови збільшення інтенсивності потоку запитів потрібно очікувати зростання його середнього значення, тобто вважати, що час обробки та інтенсивність потоку запитів пов'язані кореляційною залежністю.

Постановка задачі регресійного аналізу формулюється таким чином.

Є сукупність результатів спостережень виду (8.1). У цій сукупності один стовпець відповідає показнику, для якого необхідно встановити функціональну залежність з параметрами об'єкта і середовища, поданими іншими стовпцями. Будемо позначати показник через y^* і вважати, що йому відповідає перший стовпець матриці спостережень. Решта $m-1$ $(m>1)$ стовпців відповідають параметрам (факторам) $x_2, x_3, ..., x_m$.

Потрібно: встановити кількісний взаємозв'язок між показником і факторами. У такому випадку завдання регресійного аналізу розуміється як завдання виявлення такої функціональної залежності $y^* = f(x_2, x_3, ..., x_m)$, яка найкращим чином описує наявні експериментальні дані.

Допущення:

- кількість спостережень достатня для прояву статистичних закономірностей щодо факторів і їх взаємозв'язків;
- оброблювані ЕД містять деякі помилки (перешкоди), обумовлені похибками вимірювань, впливом неврахованих випадкових чинників;
- матриця результатів спостережень є єдиною інформацією про досліджуваний об'єкт, наявною в розпорядженні перед початком дослідження.

Функція $f(x_2, x_3, ..., x_m)$, що описує залежність показника від параметрів, називається рівнянням (функцією) регресії. Термін «регресія» (regression (лат.) – відступ, повернення до чогось) пов'язаний зі специфікою однієї з конкретних задач, вирішених на стадії становлення методу, і в нині не відбиває всієї суті методу, але продовжує застосовуватися.

Вирішення задачі регресійного аналізу доцільно розбити на декілька етапів:

- попередня обробка ЕД;
- вибір виду рівнянь регресії;
- обчислення коефіцієнтів рівняння регресії;
- перевірка адекватності побудованої функції результатами спостережень.

Попередня обробка містить стандартизацію матриці ЕД, розрахунок коефіцієнтів кореляції, перевірку їх значущості і виключення з розгляду

незначущих параметрів (ці перетворення були розглянуті в рамках кореляційного аналізу). Внаслідок перетворень будуть отримані стандартизована матриця спостережень $\boldsymbol{U}$ (через y будемо позначати стандартизовану величину y^*) і кореляційна матриця ρ.

Стандартизованій матриці U можна поставити у відповідність одну з таких геометричних інтерпретацій:

- в m-вимірному просторі осі відповідають окремим параметрам і показникам. Кожен рядок матриці подає вектор в цьому просторі, а вся матриця – сукупність n векторів в просторі параметрів;
- в n-вимірному просторі осі відповідають результатам окремих спостережень. Кожен стовпець матриці – вектор в просторі спостережень. Всі вектори в цьому просторі мають однакову довжину, яка дорівнює $\sqrt{n}$. Тоді кут між двома векторами характеризує взаємозв'язок відповідних величин. І чим менший кут, тим тісніший зв'язок (тим більший коефіцієнт кореляції).

У кореляційній матриці особливу роль відіграють елементи лівого стовпця – вони характеризують наявність або відсутність лінійної залежності між відповідним параметром u_i $(i = 2,3,...,m)$ і показником об'єкта y. Перевірка значущості дозволяє виявити такі параметри, які потрібно виключити з розгляду під час формування лінійної функціональної залежності, і тим самим спростити подальшу обробку.

11.2.2 Вибір виду рівняння регресії

Задача визначення функціональної залежності, що найкращим чином описує ЕД, пов'язана з подоланням ряду принципових труднощів. У загальному випадку для стандартизованих даних функціональну залежність показника від параметрів можна подати у вигляді

$$y = f(u_1, u_2, ..., u_p) - \varepsilon, \qquad (11.2)$$

де f – заздалегідь невідома функція, яка підлягає визначенню;

ε – помилка апроксимації ЕД.

Зазначене рівняння прийнято називати вибірковим рівнянням регресії y на u. Це рівняння характеризує залежність між варіацією показника і варіаціями факторів. А міра кореляції вимірює частку варіації показника, пов'язану з варіацією факторів. Інакше кажучи, кореляцію показника і факторів не можна трактувати як зв'язок їхніх рівнів, а регресійний аналіз не пояснює ролі факторів у створенні показника.

Ще одна особливість стосується оцінки ступеня впливу кожного фактора на показник. Регресійне рівняння не забезпечує оцінку окремого впливу кожного фактора на показник, така оцінка можлива лише у випадку, коли всі інші фактори не пов'язані з досліджуваним. Якщо

досліджуваний фактор пов'язаний з іншими, що впливають на показник, то буде отримано змішану характеристику впливу фактора. Ця характеристика містить як безпосередній вплив фактора, так і опосередкований вплив, який він вчинив через зв'язок з іншими факторами та їх впливом на показник.

У регресійне рівняння не рекомендується включати фактори, слабко пов'язані з показником, але тісно пов'язані з іншими факторами. Не включають у рівняння і фактори, функціонально пов'язані один з одним (для них коефіцієнт кореляції дорівнює 1). Включення таких факторів призводить до виродження системи рівнянь для оцінок коефіцієнтів регресії і до невизначеності рішення.

Функція *f* має підбиратися так, щоб помилка ε в деякому розумінні була мінімальною. Існує нескінченна множина функцій, що описують ЕД абсолютно точно ($\varepsilon = 0$), тобто таких функцій, які для всіх значень параметрів $u_{j,2}, u_{j,3}, ..., u_{j,m}$ набувають точно відповідних значень показника y_i, $i = 1, 2, ..., n$. Разом з тим, для всіх інших значень параметрів, відсутніх в результатах спостережень, значення показника можуть набувати будь-яких значень. Зрозуміло, що такі функції не відповідають дійсному зв'язку між параметрами і показником.

З метою вибору функціонального зв'язку заздалегідь висувають гіпотезу про те, до якого класу може належати функція *f*, а потім підбирають «кращу» функцію в цьому класі. Потрібно, щоб вибраний клас функцій мав деяку «гладкість», тобто «невеликі» зміни значень аргументів мають викликати «невеликі» зміни значень функції (ЕД містять деякі помилки вимірювань, а сама поведінка об'єкта піддається впливу перешкод, що маскують справжній зв'язок між параметрами і показником).

Простим, зручним для практичного застосування і таким, що відповідає зазначеним умовам, є клас поліноміальних функцій

$$y = a_0 + \sum_{j=2}^{m} a_j u_j + \sum_{j=2}^{m-1} \sum_{k=j+1}^{m} a_{jk} u_j u_k + \sum_{j=2}^{m} a_{jj} u_j^2 + ... + \varepsilon . \qquad (11.3)$$

Для такого класу завдання вибору функції зводиться до задачі вибору значень коефіцієнтів $a_0, a_j, a_{jk}, ..., a_{jj} ...$ Однак універсальність поліноміального подання забезпечується тільки за можливості необмеженого збільшення степеня полінома, що не завжди допустимо на практиці, тому доводиться застосовувати й інші види функцій.

Окремим випадком, широко застосовуваним на практиці, є поліном першого ступеня або рівняння лінійної регресії

$$y = a_0 + \sum_{j=2}^{m} a_j u_j + \varepsilon \ . \qquad (11.4)$$

Це рівняння в регресійному аналізі потрібно трактувати як векторне, бо мова йде про матрицю даних.

Зазвичай прагнуть забезпечити таку кількість спостережень, яка б перевищувала кількість оцінюваних коефіцієнтів моделі. Для лінійної регресії у випадку $n > m$ кількість рівнянь перевищує кількість коефіцієнтів полінома, що підлягають визначенню. Але і в цьому випадку не можна підібрати коефіцієнти таким чином, щоб помилка в кожному скалярному рівнянні ставала нульовою, оскільки до невідомих відносяться a_j і ε_i, їх кількість $n+m-1$, тобто завжди більше кількості рівнянь n. Аналогічні міркування справедливі і для поліномів вище першого степеня.

Для вибору виду функціональної залежності можна рекомендувати такий підхід:

- в просторі параметрів графічно відображають точки із значеннями показника. У разі великої кількості параметрів можна будувати точки відносно до кожного з них, отримуючи двовимірні розподіли значень;
- за розташуванням точок і на основі аналізу суті взаємозв'язку показника і параметрів об'єкта роблять висновок про приблизний вигляд регресії або її можливих варіантів;
- після розрахунку параметрів оцінюють якість апроксимації, тобто оцінюють ступінь близькості розрахункових і фактичних значень;
- якщо розрахункові і фактичні значення близькі у всій області задання, то задачу регресійного аналізу можна вважати вирішеною. В іншому випадку можна спробувати вибрати інший вид полінома або іншу аналітичну функцію, наприклад, періодичну.

11.3 Лінійні регресійні моделі

11.3.1 Математичний опис лінійної регресії

З метою досліджень часто буває зручно подати досліджуваний об'єкт у вигляді ящика, що має входи і виходи, не розглядаючи детально його внутрішню структуру. Звичайно, перетворення в ящику (на об'єкті) відбуваються (сигнали проходять по зв'язках і елементах, змінюють свою форму і т. п.), але за такого подання вони відбуваються приховано від спостерігача.

За ступенем інформованості дослідника про об'єкт існує поділ об'єктів на три типи «ящиків»:

- «білий ящик»: про об'єкт відомо все;
- «сірий ящик»: відомо структуру об'єкта, не відомо кількісні значення параметрів;
- «чорний ящик»: про об'єкт не відомо нічого.

Чорний ящик умовно зображають так, як показано на рис. 11.1. Значення на входах і виходах чорного ящика можна спостерігати і вимірювати. Вміст ящика невідомий.

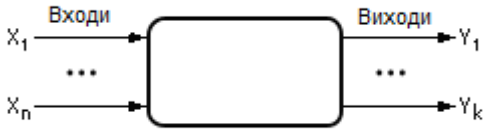


Рисунок 11.1 – Позначення чорного ящика на схемах

Завдання полягає в тому, щоб, знаючи множину значень на входах і виходах, побудувати модель, тобто визначити функцію ящика, за якою вхід перетвориться в вихід. Така задача називається задачею регресійного аналізу.

Залежно від того, доступні входи досліднику для управління або тільки для спостереження, можна говорити про активний або пасивний експеримент з ящиком.

Нехай, наприклад, перед нами стоїть завдання визначити, як залежить випуск продукції від кількості споживаної електроенергії. Результати спостережень відобразимо на графіку (рис. 11.2). Всього на графіку *n* експериментальних точок, які відповідають *n* спостереженням.

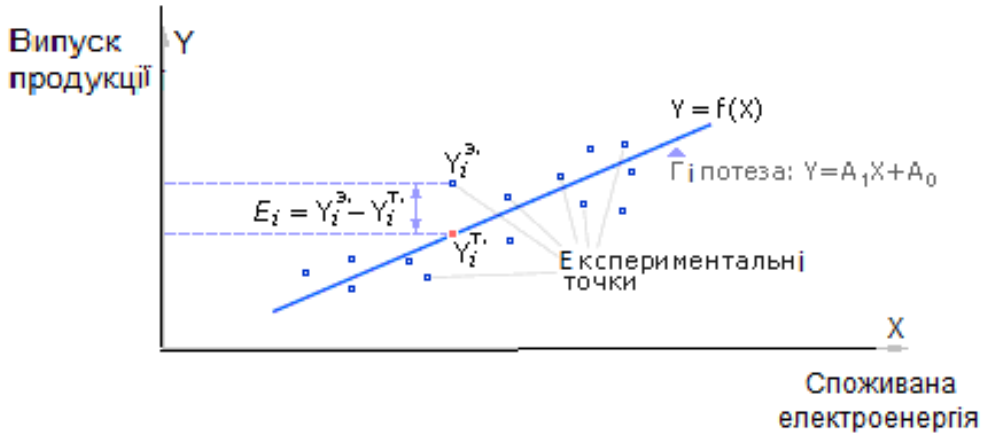


Рисунок 11.2 – Графічний вигляд подання результатів спостереження над чорним ящиком

Для початку припустимо, що ми маємо справу з чорним ящиком, який має один вхід і один вихід. Припустимо для простоти, що залежність між входом і виходом лінійна або майже лінійна. Тоді ця модель буде називатися лінійною одновимірною регресійною моделлю.

Розв'язання задачі регресійного аналізу доцільно розбити на декілька етапів:

- попередня обробка ЕД;
- вибір виду рівнянь регресії;
- обчислення коефіцієнтів рівняння регресії;

- перевірка адекватності побудованої функції результатами спостережень.

1. Дослідник вносить гіпотезу про структуру ящика

Розглядаючи експериментально отримані дані, припустимо, що вони підпорядковуються лінійній гіпотезі, тобто вихід *Y* залежить від входу *X* лінійно, тому гіпотеза має вигляд: $Y = A_0 + A_1 X$ (рис. 11.2).

2. Визначення невідомих коефіцієнтів A_0 і A_1 моделі

Лінійна одновимірна модель (рис. 11.3).

Рисунок 11.3 – Одновимірна модель чорного ящика

Для кожної з *n* знятих експериментально точок обчислимо помилку E_i між експериментальним значенням $Y_i^{експ}$ і теоретичним значенням $Y_i^{теор}$, що лежить на гіпотетичній прямій $A_1X + A_0$ (див. рис. 11.2):

$$E_i = Y_i^{експ} - Y_i^{теор}, \qquad (11.5)$$

$$E_i = Y_i^{експ} - (A_0 + A_1 X_i),\ i = \overline{1,n}.$$

Помилки E_i для всіх *n* точок потрібно додати. Щоб позитивні помилки не компенсували в сумі негативні, кожну з помилок підносять до квадрата і додають їх значення в сумарну помилку *E* вже одного знака:

$$E_i = (Y_i^{експ} - (A_0 + A_1 X_i))^2,\ i = \overline{1,n},$$

$$E(A_0, A_1) = \sum_{i=1}^{n} E_i^2 .$$

Мета методу – мінімізація сумарної помилки E за рахунок підбору коефіцієнтів A_0, A_1. Іншими словами, це означає, що необхідно знайти такі коефіцієнти A_0, A_1 лінійної функції $Y = A_0 + A_1 X$, щоб її графік проходив якомога ближче одночасно до всіх експериментальних точок. Тому цей метод називається методом найменших квадратів.

$$F(A_0, A_1) = \sum_{i=1}^{n} E_i^2 = \sum_{i=1}^{n} (Y_i - A_0 - A_1 X_i)^2 \Rightarrow \min_{A_0, A_1}. \qquad (11.6)$$

Сумарна помилка E є функцією двох змінних A_0 і A_1, тобто $E(A_0, A_1)$, змінюючи які, можна впливати на величину сумарної помилки (рис. 11.4).

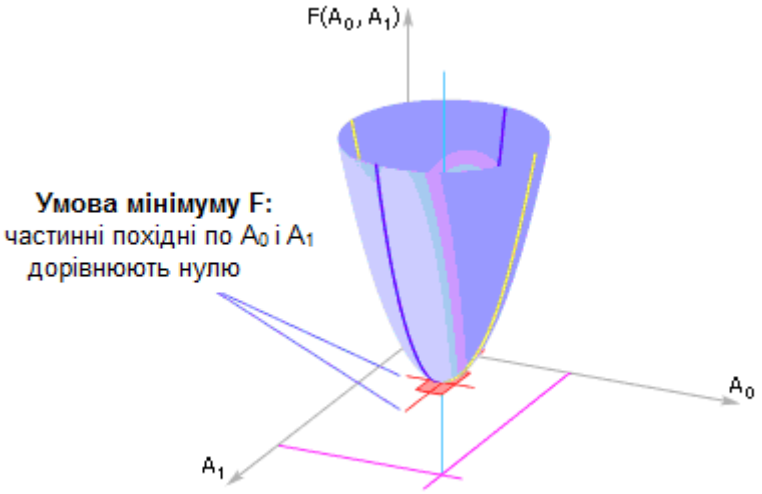


Рисунок 11.4 – Приблизний вид функції помилки

Щоб сумарну помилку мінімізувати, знайдемо частинні похідні від функції E за кожною змінною і прирівняємо їх до нуля (умова екстремуму):

$$\frac{\partial F}{\partial A_0} = -2\sum_{i=1}^{n} Y_i - A_0 - A_1 X_i) = 0,$$

$$\frac{\partial F}{\partial A_1} = -2\sum_{i=1}^{n} Y_i - A_0 - A_1 X_i) = 0.$$

Після розкриття дужок отримаємо систему з двох лінійних рівнянь:

$$\sum_{i=1}^{n} A_0 + A_1 \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} Y_i,$$

$$A_0 \sum_{i=1}^{n} X_i + A_1 \sum_{i=1}^{n} X_i^2 = \sum_{i=1}^{n} X_i Y_i.$$

Для знаходження коефіцієнтів A_0 і A_1 методом Крамера подамо систему в матричній формі:

$$\cdot \begin{pmatrix} n & \sum_{i=1}^{n} X_i \\ \sum_{i=1}^{n} X_i & \sum_{i=1}^{n} X_i^2 \end{pmatrix} \cdot \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} Y_i \\ \sum_{i=1}^{n} Y_i X_i \end{pmatrix}.$$

Розв'язок має вигляд:

$$A_0 = \frac{\sum_{i=1}^{n} Y_i \sum_{i=1}^{n} X_i^2 - \sum_{i=1}^{n} Y_i X_i \sum_{i=1}^{n} X_i}{n\sum_{i=1}^{n} X_i^2 - \left(\sum_{i=1}^{n} X_i\right)^2}, \quad A_1 = \frac{n\sum_{i=1}^{n} Y_i X_i - \sum_{i=1}^{n} Y_i \sum_{i=1}^{n} X_i}{n\sum_{i=1}^{n} X_i^2 - \left(\sum_{i=1}^{n} X_i\right)^2}.$$

3. Перевірка

Щоб визначити, приймається гіпотеза чи ні, потрібно, по-перше, розрахувати помилку між точками заданої експериментальної і отриманої теоретичної залежності (11.5) і сумарну помилку:

$$E(A_0, A_1) = \sum_{i=1}^{n} E_i^2, \; i = \overline{1,n}, \tag{11.7}$$

і, по-друге, необхідно знайти значення σ за формулою

$$\sigma = \sqrt{\frac{E}{n}},$$

де E – сумарна помилка,

n – загальне число експериментальних точок.

Якщо в смугу, обмежену лініями $Y^{теор} - S$ і $Y^{теор} + S$ (рис. 11.5), потрапляє 68,26 % і більше експериментальних точок $Y_i^{експ}$, то висунута нами гіпотеза приймається. В іншому випадку вибирають більш складну гіпотезу або перевіряють вихідні дані. Якщо потрібна велика впевненість в результаті, то використовують додаткову умову: в смугу, обмежену лініями $Y^{теор} - S$ і $Y^{теор} + S$, мають потрапити 95,44 % і більше експериментальних точок $Y_i^{експ}$.

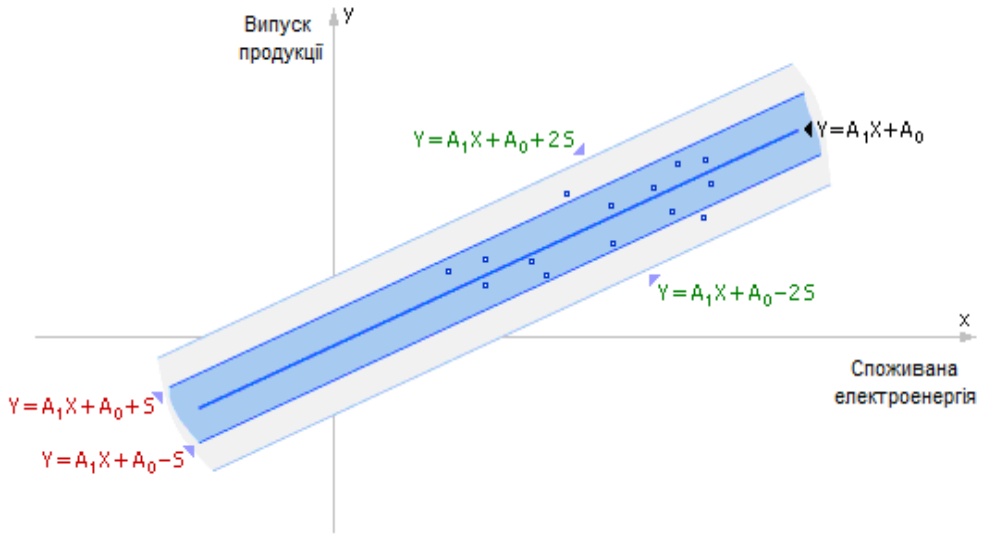


Рисунок 11.5 – Дослідження допустимості прийняття гіпотези

Відстань S пов'язана з σ таким співвідношенням:

$$S = \sigma / \sin(\beta) = \sigma / \sin(90° - arctg(A_1)) = \sigma / \cos(arctg(A_1)),$$

що й проілюстровано на рис. 11.6.

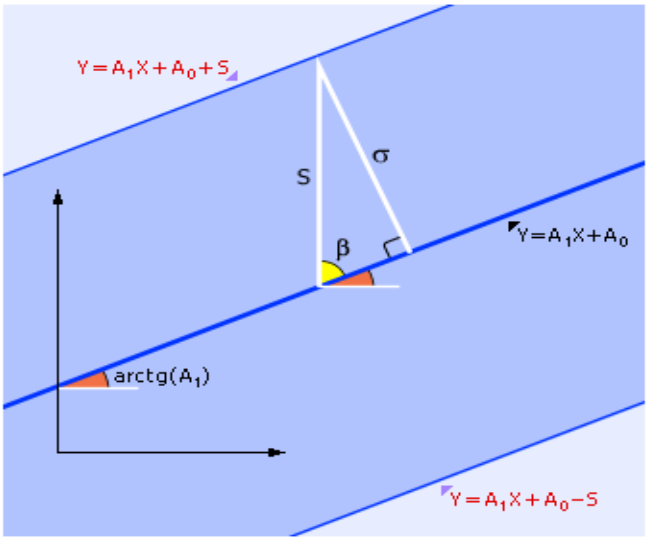


Рисунок 11.6 – Зв'язок значень σ і S

11.3.2 Приклад аналізу з використанням лінійної регресії

Приклад 11.1. Спираючись на наявні експериментальні дані, побудуємо модель (визначимо функцію чорного ящика), за якої вхід перетворюється на вихід

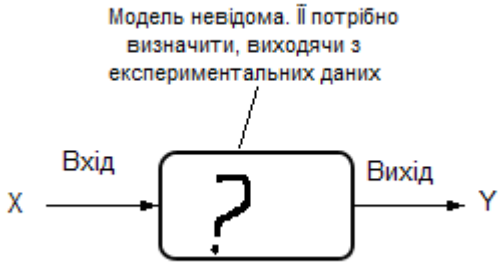


Рисунок 11.7 – Схема одновимірної регресійної моделі

Дані, отримані як результат експериментальних вимірювань, зображено на рис. 11.8. Вони містять $n = 8$ експериментальних точок.

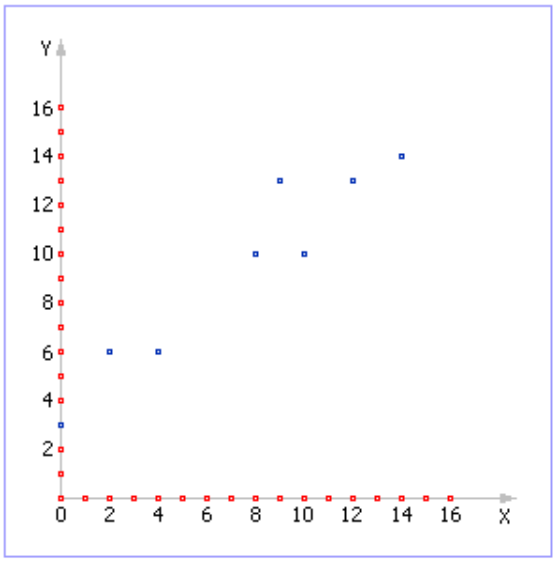


Рисунок 11.8 – Графік експериментальних даних

Розв'язування прикладу 11.1

1. Внесемо отримані дані в таблицю 11.1.

Таблиця 11.1 –Експериментальні дані

Номер вимірювання i	Параметр X_i	Параметр Y_i
1	0	3
2	2	6
3	4	6
4	8	10
5	9	13
6	10	10
7	12	13
8	14	14

2. Розглядаючи експериментальні дані, припустимо, що вони підлягають лінійним законам; висуваємо гіпотезу: $Y = A_0 + A_1 X$.

3. Запишемо рівняння помилки і сумарної помилки:

$$E_i = Y_i^{експ} - Y_i^{теор},\ i = \overline{1,n},$$

$$E_i = Y_i^{експ} - (A_0 + A_1 X_i),\ i = \overline{1,n},$$

$$F(A_0, A_1) = \sum_{i=1}^{n} E_i^2 = \sum_{i=1}^{n} (Y_i - A_0 - A_1 X_i)^2 \Rightarrow \min_{A_0, A_1}.$$

4. Для знаходження екстремуму прирівняємо частинні похідні функції F за змінними A_0 і A_1 до нуля (умова екстремуму):

$$\frac{\partial F}{\partial A_0} = -2\sum_{i=1}^{n} Y_i - A_0 - A_1 X_i) = 0,$$

$$\frac{\partial F}{\partial A_1} = -2\sum_{i=1}^{n} Y_i - A_0 - A_1 X_i) = 0.$$

Після розкриття дужок отримаємо систему з двох лінійних рівнянь:

$$\sum_{i=1}^{n} A_0 + A_1 \sum_{i=1}^{n} X_i = \sum_{i=1}^{n} Y_i,$$
$$A_0 \sum_{i=1}^{n} X_i + A_1 \sum_{i=1}^{n} X_i^2 = \sum_{i=1}^{n} X_i Y_i.$$

Для зручності обчислень складемо таблицю 11.2.

Таблиця 11.2 – Таблиця проміжних обчислень

Номер вимірювання i	Параметр X_i	Параметр Y_i	X_i^2	$X_i Y_i$
1	0	3	0	0
2	2	6	4	12
3	4	6	16	24
4	8	10	64	80
5	9	13	81	117
6	10	10	100	100
7	12	13	144	156
8	14	14	196	196
Сума:	59	75	605	685

5. Для знаходження коефіцієнтів A_0 і A_1 методом Крамера подамо систему в матричній формі:

$$\begin{pmatrix} n & \sum_{i=1}^{n} X_i \\ \sum_{i=1}^{n} X_i & \sum_{i=1}^{n} X_i^2 \end{pmatrix} \cdot \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} = \begin{pmatrix} \sum_{i=1}^{n} Y_i \\ \sum_{i=1}^{n} Y_i X_i \end{pmatrix}.$$

Підставляючи конкретні значення з табл. 11.2, отримаємо:

$$\begin{pmatrix} 8 & 59 \\ 59 & 605 \end{pmatrix} \cdot \begin{pmatrix} A_0 \\ A_1 \end{pmatrix} = \begin{pmatrix} 75 \\ 685 \end{pmatrix}.$$

Знаходимо значення A_0 і A_1:

$$A_0 = \frac{75 \cdot 605 - 685 \cdot 59}{8 \cdot 605 - 59^2} = \frac{4960}{1359} \approx 3.6497$$

$$A_1 = \frac{8 \cdot 685 - 75 \cdot 59}{8 \cdot 605 - 59^2} = \frac{1055}{1359} \approx 0.7763$$

Отже, знайдені значення A_0=3,65 і A_1= 1055/1359 = 0,78 забезпечують проходження графіка $Y=A_0+A_1X$ якомога ближче одночасно до всіх експериментальних точок.

Таким чином, ми отримали таке лінійне рівняння: Y=3.65+0.78X.

6. Тепер необхідно перевірити, чи маємо ми право прийняти отриману гіпотезу Y= 3.65+0.78X як правильну, чи вона має бути відхилена. Для цього необхідно розрахувати помилку E_i між точками заданої експериментальної і отриманої теоретичної залежностей (див. табл. 11.3), сумарну помилку F і значення σ за формулами (11.5), (11,7) і (11.8), відповідно:

$$\sigma = \sqrt{\frac{E}{n}}. \qquad (11.8)$$

Таблиця 11.3 – Обчислення помилок між точками заданої експериментальної і отриманої теоретичної залежностей

Номер вимірювання i	Параметр X_i	Параметр Y_i	$E_i = Y_i - 3.65 - 0.78X_i$	E_i^2
1	0	3	-0.65	0.4225
2	2	6	0.79	0.6241
3	4	6	-0.77	0.5929
4	8	10	0.11	0.0121
5	9	13	2.33	5.4289
6	10	10	-1.45	2.1025
7	12	13	-0.01	0.0001
8	14	14	-0.57	0.3249

Сумарна помилка становить:
E=0.4225+0.6241+0.5929+0.0121+5.4289+2.1025+0.0001+0.3249=9.5080.

Значення σ = *sqrt* (9.5080/8) = 1.09.

Знайдемо значення $S = \sigma/cos(arctg(A_1))$ = 1.09/*cos*(*arctg*(0.78)) = 1.38.

Якщо в смугу, обмежену лініями Y=3.65+0.78X-1.38 і Y=3.65+0.78X+1.38 потрапить 68.26 % або більше з усіх

експериментальних точок, то можна зробити висновок про те, що наша гіпотеза про лінійність правильна.

Остаточні розрахунки (табл. 11.4) показують, що 6 точок з 8 (тобто 75 %) потрапляють в смугу, обмежену лініями Y=3.65+0.78X-1.38 і Y=3.65+0.78X+1.38, з чого робимо висновок: залежність між входом і виходом моделі лінійна, тобто висунута нами гіпотеза правильна.

Таблиця 11.4 – Перевірка попадання точок в довірчий інтервал

i	X_i	Y_i	$Y = 3.65 + 0.78X_i - 1.38$	$Y = 3.65 + 0.78X_i + 1.38$	Потрапляє чи ні?
1	0	3	2.27	5.03	так
2	2	6	3.83	6.59	так
3	4	6	5.39	8.15	так
4	8	10	8.51	11.27	так
5	9	13	9.29	12.05	ні
6	10	10	10.07	12.83	ні
7	12	13	11.63	14.39	так
8	14	14	13.19	15.95	так

На рис. 11.9 наведено графічну ілюстрацію проведених розрахунків.

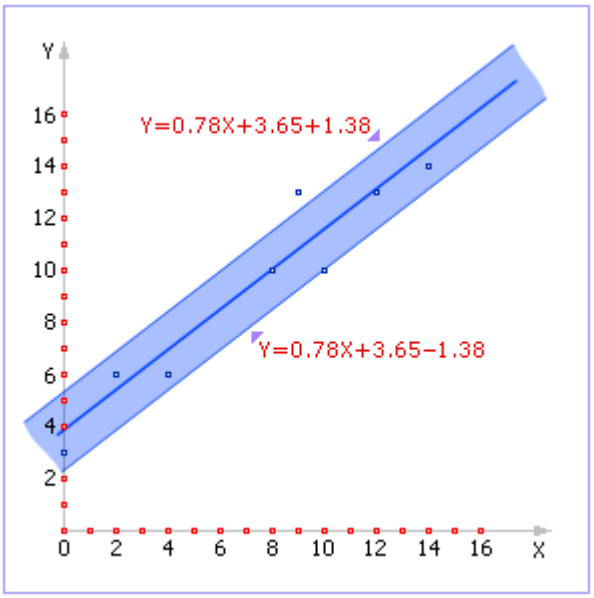


Рисунок 11.9 – Знайдена лінійна регресійна залежність з довірчим інтервалом [-S, +S]

11.4 Поліноміальна регресія

Лінійна регресія потребує, щоб відношення вихідної (залежної) величини від вхідних незалежних даних було лінійним. Для більш складного розподілу даних потрібно використовувати моделі вищого порядку, наприклад, функції поліноміального класу виду (11.3). Нехай,

наприклад, задано набір експериментальних даних, що описують статичну характеристику ланки (табл. 11.5).

Таблиця 11.5 – Залежність вихідного сигналу від вхідного в статиці

X	0	1	2	3	4	5	6	7	8	9
Y	1,2	2,1	5,6	9,8	16,2	28,1	37,3	54,1	68,2	91,1

Потрібно апроксимувати її поліномом.
Нанесемо експериментальні точки на графік (рис. 11.10).

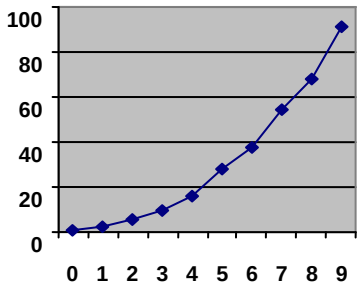


Рисунок 11.10 – Статична характеристика досліджуваного об’єкта

За виглядом графіка зробимо припущення, що експериментальні дані описуються квадратичною функцією

$$\varphi(x) = a_0 + a_1 x + a_2 x^2. \qquad (11.9)$$

Відповідно до методу найменших квадратів знайдемо коефіцієнти a_0, a_1, a_2, які забезпечують мінімальне СКВ:

$$\sum_{i=1}^{n} (y_i - (a_0 + a_1 x + a_2 x^2))^2 = \min.$$

Знаходимо частинні похідні лівої частини за коефіцієнтами a_0, a_1, a_2 і прирівнюємо їх до нуля:

$$\begin{cases} \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i + a_2 x_i^2)) = 0; \\ \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i + a_2 x_i^2)) x_i = 0; \\ \sum_{i=1}^{n} (y_i - (a_0 + a_1 x_i + a_2 x_i^2)) x_i^2 = 0. \end{cases} \qquad (11.10)$$

Розкриваємо дужки і перетворюємо рівняння:

$$\begin{cases} na_0 + (\sum_{i=1}^{n} x_i)a_1 + (\sum_{i=1}^{n} x_i^2)a_2 = \sum_{i=1}^{n} y_i; \\ (\sum_{i=1}^{n} x_i)a_0 + (\sum_{i=1}^{n} x_i^2)a_1 + (\sum_{i=1}^{n} x_i^3)a_2 = \sum_{i=1}^{n} x_i y_i; \\ (\sum_{i=1}^{n} x_i^2)a_0 + (\sum_{i=1}^{n} x_i^3)a_1 + (\sum_{i=1}^{n} x_i^4)a_2 = \sum_{i=1}^{n} x_i^2 y_i. \end{cases} \quad (11.11)$$

Ця система є системою лінійних алгебраїчних рівнянь відносно невідомих a_0, a_1, a_2 і розв’язується одним із відомих чисельних методів. Така система називається системою нормальних рівнянь. В загальному випадку для полінома m-го степеня система буде мати вигляд:

$$\begin{cases} na_0 + \left(\sum x_i\right)a_1 + \left(\sum x_i^2\right)a_2 + \cdots + \left(\sum x_i^m\right)a_m = \sum x_i^* y_i; \\ \left(\sum x_i\right)a_0 + \left(\sum x_i^2\right)a_1 + \left(\sum x_i^3\right)a_2 + \cdots + \left(\sum x_i^{m+1}\right)a_m = \sum x_i' y_i; \\ \text{-------------------------------------} \\ \left(\sum x_i^m\right)a_0 + \left(\sum x_i^{m+1}\right)a_1 + \left(\sum x_i^{m+2}\right)a_2 + \cdots + \left(\sum x_i^{2m}\right)a_m = \sum x_i^m y_i. \end{cases} \quad (11.12)$$

Таким чином, в обчислювальному плані задача апроксимації полягає в формуванні розширеної матриці рівнянь і її розв'язання за допомогою відомих чисельних методів. Вхідними даними для складання алгоритму апроксимації за методом найменших квадратів слугують масиви експериментальних даних *x*[*n*] та *y*[*n*], де (*n*+*1*) – кількість елементів в масиві; *m* – степінь апроксимувального полінома.

В спрощеному вигляді алгоритм має складатись з чотирьох кроків: введення експериментальних даних; обчислення необхідних сум, визначення коефіцієнтів системи (11.12); розв'язання отриманої системи і друк результатів (числових значень коефіцієнтів поліному a_i). Більш детально алгоритм зображено на рис. 11.11. Під час обчислення коефіцієнтів p_{ij} біля невідомих (блок 6 алгоритму) використовується помічена в системі (11.12) закономірність

$$p_{ij} = \sum_{k=0}^{n} x_k^{(i+j)}.$$

Вільні члени рівнянь обчислюються за формулою (блок 10 алгоритму)

$$p_{ij} = \sum_{k=0}^{n} x_k^i y_k.$$

Кількість обчислень скорочено за рахунок того, що повністю обчислюються тільки 1 рядок, останній і передостанній стовпці матриці системи (блоки 4 і 13 алгоритму). Дії в блоках 1, 6 і 10 подано в спрощеному вигляді. Їх потрібно реалізувати у вигляді циклічних процесів. В алгоритмі прийнято такі позначення:

i, j – індекси, що позначають номер рядка і стовпця, відповідно;

e – індекс, яким позначається номер елемента в масиві *x* або *y*;

p_y – коефіцієнти, що заміняють відповідні суми в системі (11.12).

Для розв'язання систем лінійних алгебраїчних рівнянь зручно використовувати програмні середовища, орієнтовані на вирішення інженерних задач, наприклад, Matlab або Mathcad, в яких існують готові вбудовані функції.

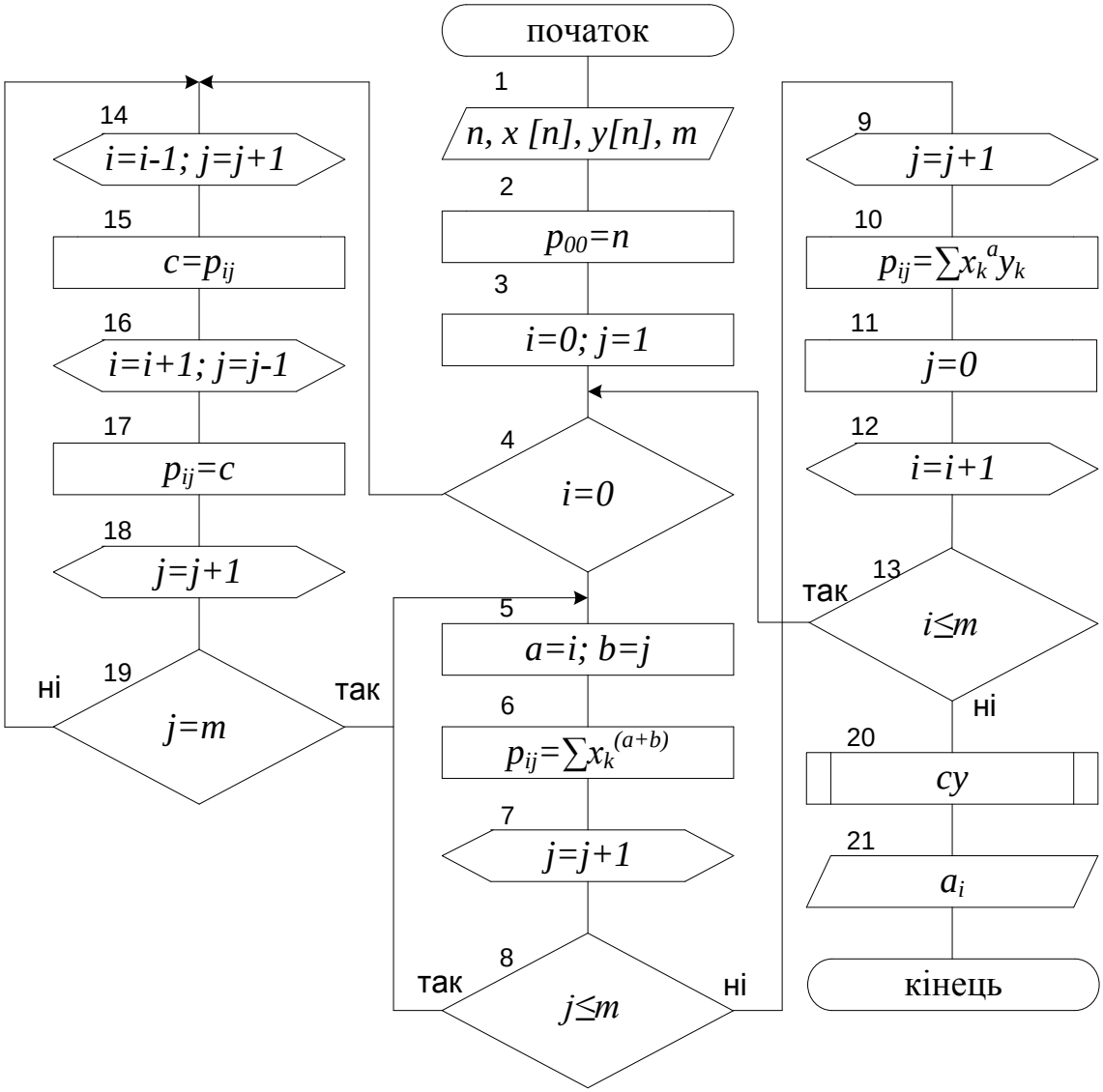


Рисунок 11.11 – Блок-схема алгоритму апроксимації експериментальних даних за методом найменших квадратів

Контрольні питання і завдання

1. Запишіть приклад матриці експериментальних даних (ЕД).
2. Що означають рядки і стовпці матриці ЕД?
3. Яка імовірнісна закономірність передбачається в формуванні матриці ЕД?
4. Як здійснюється перехід від вихідної до стандартизованої матриці ЕД?
5. Що впливає на подання досліджуваного об’єкта моделями «білої», «сірої» і «чорної» скриньки?
6. В яких випадках застосовується інтерполяційна модель даних, а в яких – апроксимаційна?
7. Сформулюйте постановку задачі регресійного аналізу.
8. Що називається рівнянням регресії?
9. Назвіть етапи вирішення задачі регресійного аналізу.
10. Запишіть загальний вигляд рівняння регресії.
11. Яким чином визначається вигляд функції регресії?
12. Який критерій використовують для визначення коефіцієнтів рівняння регресії?
13. Що являє собою в обчислювальному плані задача апроксимації під час побудови регресійної моделі?
14. Запишіть рівняння лінійної і поліноміальної регресії.
15. Яким чином здійснюється оцінювання адекватності побудованої регресійної моделі?
16. Як обчислюється довірчий інтервал для побудованої лінії регресії?
17. Яка кількість експериментальних точок має потрапляти в довірчий інтервал правильно побудованої регресійної моделі?

РОЗДІЛ 12
НАВЧАННЯ З УЧИТЕЛЕМ. УЗАГАЛЬНЕНІ МЕТОДИ РЕГРЕСІЙНОГО АНАЛІЗУ

12.1 Лінійна багатофакторна модель регресії

В Розділі 11 розглянуто найпростіший варіант регресії, яка визначає залежність однієї вихідної величини від однієї незалежної вхідної величини (рис. 11.7). В регресійному аналізі незалежну вхідну величину, вплив якої на вихідну залежну визначається, часто називають фактором, тому розглянуту регресію називають однофакторною. На практиці найчастіше потрібно визначати вплив декількох факторів на вихідну залежну величину, що можна подати у вигляді чорного ящика з декількома входами і одним виходом, як це показано на рис. 12.1. Така регресія називається багатофакторною або множинною. Як і для лінійної однофакторної регресії маємо на увазі, що фактори мають випадковий характер, тобто їх значення можуть змінюватися від експерименту до експерименту.

Рисунок 12.1 – Структурна схема багатовходового чорного ящика

Для цього випадку рівняння лінійної регресії можна записати у вигляді:

$$Y = A_0 + A_1 X_1 + A_2 X_2 + ... + A_m X_m . \qquad (12.1)$$

Для проведених n експериментів, в кожному з яких було визначено значення вихідної величини y_i залежно від i-их експериментальних значень факторів x_{ji}; $j = \overline{1,m}$; $i = \overline{1,n}$ з огляду на вираз (12.1) можна записати таку систему рівнянь для теоретичних значень величини y_i^{th}:

$$\begin{cases} y_1^{th} = A_0 \cdot 1 + A_1 \cdot x_{11} + A_2 \cdot x_{21} + ... + A_m \cdot x_{m1}, \\ y_2^{th} = A_0 \cdot 1 + A_1 \cdot x_{12} + A_2 \cdot x_{22} + ... + A_m \cdot x_{m2}, \\ ..... \\ y_i^{th} = A_0 \cdot 1 + A_1 \cdot x_{1i} + A_2 \cdot x_{2i} + ... + A_m \cdot x_{mi}, \\ ..... \\ y_n^{th} = A_0 \cdot 1 + A_1 \cdot x_{1n} + A_2 \cdot x_{2n} + ... + A_m \cdot x_{mn}. \end{cases} \qquad (12.2)$$

З виразу (12.2) матрицю експериментальних даних (ЕД) для вхідних впливів $X_1, X_2,...,X_j,...,X_m$, доповнену одиницями в першому стовпці, можна записати таким чином:

$$X = \begin{vmatrix} 1 & x_{11} & x_{21} & \dots & x_{m1} \\ 1 & x_{12} & x_{22} & \dots & x_{m2} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{1i} & x_{2i} & \dots & x_{mi} \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x_{1n} & x_{2n} & \dots & x_{mn} \end{vmatrix}. \qquad (12.3)$$

Похибку обчислення теоретичного значення функції регресії y_i^{th} відносно експериментального значення y_i для кожної експериментальної точки можна знайти з виразу:

$$E_i = y_i - y_i^{th}, \quad i = \overline{1,n}, \qquad (12.4)$$

або

$$E_i = y_i - (A_0 \cdot 1 + A_1 \cdot x_{1i} + A_2 \cdot x_{2i} + ... + A_m \cdot x_{mi}), \quad i = \overline{1,n}. \qquad (12.5)$$

Для мінімізації загальної похибки моделі регресії використаємо функцію F критерію мінімуму середньоквадратичної похибки $\sum_{i=1}^{n} E_i^2$:

$$F(A_0, A_1, ..., A_m) = \sum_{i=1}^{n} (Y_i - ((A_0 \cdot 1 + A_1 \cdot x_{1i} + ... + A_m \cdot x_{mi}))^2 \Rightarrow \min_{A_0, A_1, ..., A_m}. \qquad (12.6)$$

Для знаходження екстремуму прирівняємо частинні похідні функції F за змінними A_0 , A_1 , …, A_m до нуля (умова екстремуму):

$$\frac{\partial F}{\partial A_0} = -2\sum_{i=1}^{n} (y_i - A_0 - A_1 \cdot x_{1i} - A_2 \cdot x_{2i} - ... - A_m \cdot x_{mi}) = 0,$$

$$\frac{\partial F}{\partial A_1} = -2\sum_{i=1}^{n} (y_i - A_0 - A_1 \cdot x_{1i} - A_2 \cdot x_{2i} - ... - A_m \cdot x_{mi}) x_{1i} = 0,$$

……..

$$\frac{\partial F}{\partial A_m} = -2\sum_{i=1}^{n} (y_i - A_0 - A_1 \cdot x_{1i} - A_2 \cdot x_{2i} - ... - A_m \cdot x_{mi}) x_{mi} = 0.$$

Після елементарних перетворень отримаємо систему лінійних алгебраїчних рівнянь *m+1* порядку, яку для зручності обчислень запишемо в такому вигляді:

$$\begin{vmatrix} n & \sum_{i=1}^{n} x_{1i} & \sum_{i=1}^{n} x_{2i} & \dots & \sum_{i=1}^{n} x_{mi} \\ \sum_{i=1}^{n} x_{1i} & \sum_{i=1}^{n} x_{1i}x_{1i} & \sum_{i=1}^{n} x_{1i}x_{2i} & \dots & \sum_{i=1}^{n} x_{1i}x_{mi} \\ \sum_{i=1}^{n} x_{2i} & \sum_{i=1}^{n} x_{2i}x_{1i} & \sum_{i=1}^{n} x_{2i}x_{2i} & \dots & \sum_{i=1}^{n} x_{2i}x_{mi} \\ \dots & \dots & \dots & \dots & \\ \sum_{i=1}^{n} x_{mi} & \sum_{i=1}^{n} x_{mi}x_{1i} & \sum_{i=1}^{n} x_{mi}x_{2i} & \dots & \sum_{i=1}^{n} x_{mi}x_{1m}. \end{vmatrix} \cdot \begin{vmatrix} A_0 \\ A_1 \\ A_2 \\ \dots \\ A_m \end{vmatrix} = \begin{vmatrix} \sum_{i=1}^{n} y_i \\ \sum_{i=1}^{n} y_i x_{1i} \\ \sum_{i=1}^{n} y_i x_{2i} \\ .... \\ \sum_{i=1}^{n} y_i x_{mi} \end{vmatrix} . \quad (12.7)$$

В матричному вигляді вираз (12.7) можна записати так:

$$X^T \cdot X \cdot A^T = X^T \cdot Y, \quad (12.8)$$

де символ T означає операцію транспонування.

Модель лінійної множинної регресії виду (12.8) корисна тим, що до неї можна звести шляхом підстановок і переозначень більшість нелінійних моделей.

12.2 Нелінійні багатофакторні моделі регресії

12.2.1 Поліноміальна багатофакторна модель

Як приклад поліноміальної множинної регресійної моделі розглянемо модель системи з двома входами і одним виходом (рис. 12.2).

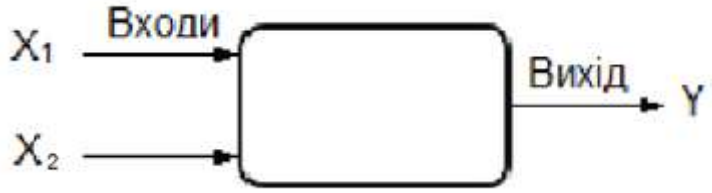


Рисунок 12.2 – Структура чорного ящика для двовходової моделі

Нехай залежність вихідної величини від вхідних нагадує квадратичну (поліном другого степеня), тоді цю залежність можна записати у вигляді функції двох змінних:

$$Y = A_0 + A_1 X_1 + A_2 X_2 + A_3 X_1 X_2 + A_4 X_1^2 + A_5 X_2^2. \quad (12.9)$$

Перевизначимо змінні у виразі (12.9) у такий спосіб:

$$X_1^* = X_1, X_2^* = X_2, X_3^* = X_1 X_2, X_4^* = X_1^2, X_5^* = X_2^2.$$

Підставиши ці значення змінних у вираз (12.9), отримаємо знову вираз лінійної множинної регресії:

$$Y = A_0 + A_1 X_1^* + A_2 X_2^* + A_3 X_3^* + A_4 X_4^* + A_5 X_5^*. \qquad (12.10)$$

Модель чорного ящика для лінійної множинної регресії тепер виглядає таким чином (рис. 12.3):

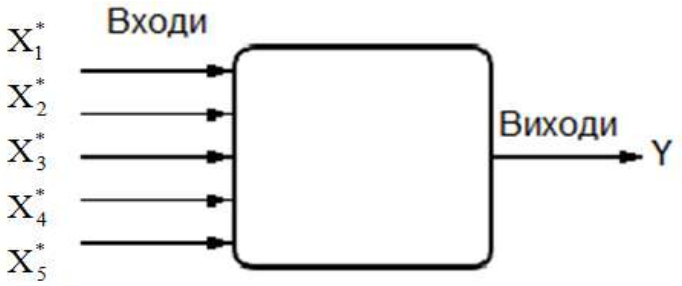


Рисунок 12.3 – Структура чорного ящика для моделі множинної лінійної регресії

В загальному випадку для *n* входів (X_1, X_2,...,X_n) квадратична регресійна функція за аналогією з (12.9) запишеться у вигляді:

$$Y = W_0 + \sum_{j=1}^{n} W_j X_j + \sum_{j=1}^{n-1} \sum_{k=j+1}^{n} W_{jk} X_j X_k + \sum_{j=1}^{n} W_{jj} X_j^2 . \qquad (12.11)$$

Кількість членів у функції (12.11) дорівнює:

$$1 + n + \frac{n(n-1)}{2} + n = \frac{(n+1)(n+2)}{2} . \qquad (12.12)$$

Перейменувавши відповідним чином змінні і коефіцієнти в виразі (12.11), отримаємо рівняння квадратичної регресії для випадку *n*-входів, подібне рівнянню (12.10).

Функції множинної регресії, що задаються поліномом *r*-го степеня, можна записати рекурентним співвідношенням:

$$Y(X^r) = (\sum_{p_1=1}^{n} \sum_{p_2=p_1}^{n} \dots \sum_{p_r=p_{r-1}} W_{p_1 p_2 \dots p_r} X_{p_1} X_{p_2} \dots X_{p_r}) + Y(X^{r-1}), \qquad (12.13)$$

де $Y(X^0) = W_0$.

Вирази (12.11) і (12.13) показують, що поліноміальну багатофакторну (множинну) регресію будь-якого степеня можна звести за допомогою відповідних замін і підстановок до лінійної багатофакторної регресії.

12.2.2 Експоненціальна багатофакторна регресійна модель

У випадку, коли залежність вихідної величини від вхідних впливів (факторів) має експоненціальний характер (рис. 12.4), можна виконати перетворення таким чином, щоб звести її до вигляду лінійної багатофакторної моделі.

Рисунок 12.4 – Зображення моделі багатовходового чорного ящика на схемах

$$Y = e^{B_0+B_1X_1+B_2X_2+...+B_mX_m} . \quad (12.14)$$

Прологарифмуємо ліву і праву частину рівняння (12.14):

$$\ln(Y) = B_0 + B_1 \cdot X_1 + B_2 \cdot X_2 + \ldots + B_m \cdot X_m. \quad (12.15)$$

Виконаємо заміну $Y^* = \ln(Y)$ і підставимо в (12.15), отримаємо рівняння множинної лінійної регресії:

$$Y^* = B_0 + B_1 \cdot X_1 + B_2 \cdot X_2 + \ldots + B_m \cdot X_m. \quad (12.16)$$

12.2.3 Мультиплікативна багатофакторна модель регресії

В тому випадку, коли вплив незалежних факторів на вихідну величину має мультиплікативний характер, модель регресії знову ж таки можна звести до вигляду лінійної:

$$Y = A_0 \cdot X_1^{A_1} \cdot X_2^{A_2} \cdot ... \cdot X_m^{A_m} . \quad (12.17)$$

Прологарифмуємо ліву і праву частини виразу (12.17), отримаємо:

$$\ln(Y) = \ln(A_0) + A_1 \cdot \ln(X_1) + A_2 \cdot \ln(X_2) + \ldots + A_m \cdot \ln(X_m). \quad (12.18)$$

Виконаємо такі заміни:

$$Y^* = \ln(Y),\ A_0^* = \ln(A_0),\ X_1^* = \ln(X_1),\ X_2^* = \ln(X_2),\ \ldots,\ X_m^* = \ln(X_m).$$

.

Підставимо отримані значення у вираз (12.18):

$$Y^* = A_0^* + A_1 \cdot X_1^* + ... + A_m \cdot X_m^* . \quad (12.19)$$

Отримали вираз лінійної багатофакторної регресії.

12.3 Узагальнені методи ідентифікації регресійних моделей

Оскільки в попередньому розділі було показано, що всі види залежності між вихідною величиною і множиною впливів на неї можна звести до лінійної регресії, то визначення аналітичної функції регресії в загальному випадку можна звести до розв'язання відносно невідомих коефіцієнтів $A_0, A_1, A_2, \ldots, A_m$ матричного рівняння виду (12.8), отриманого на основі критерію мінімуму середньоквадратичної помилки,:

$$A = (X^T X)^{-1} X^T Y. \qquad (12.20)$$

Якщо коваріаційна матриця $\sum=(X^T X)$ має неповний ранг, то обернену до неї марицю $\sum^{-1}=(X^T X)^{-1}$ отримати неможливо. Для виконання обернення матриці в цьому випадку доводиться відкидати лінійно залежні фактори (вхідні впливи) або застосовувати методи регуляризації чи методи головних компонент.

На практиці частіше зустрічається проблема мультиколінеарності – коли матриця $\sum$ має повний ранг, але бизька до деякої матриці неповного рангу. В такому випадку кажуть, що матриця $\sum$ погано обумовлена. Геометрично це означає, що вибіркові значення вхідних величин (факторів) зосереджені поблизу лінійного підпростору меншої розмірності $k<n$. Ознакою мультиколінеарності є наявність в матриці власних значень, близьких до нуля. Розв'язання матричного рівняння (12.20) в цьому випадку стає нестійким – малі значення похибок вимірювання в експерименті вхідних впливів і вихідної величини в навчальній вибірці можуть суттєво вплинути на вектор розв'язку $\vec{A}$. Відповідно похибки вимірювання вхідних величин X в тестовій вибірці вплинуть на якість функції регресії $Y(X, A)$. Таким чином, мультиколінеарність тягне за собою не тільки нестійкість і необхідність перенавчання, а також неможливість правильної інтерпретації коефіцієнтів регресії, оскільки за їх величиною неможливо визначати ступінь важливості різних факторів. Тому нині розроблено багато методів, які дозволяють усунути проблему мультиколінеарності шляхом регуляризації матриці експериментальних даних X.

До таких методів можна віднести кореляційний аналіз і метод аналізу головних компонент (PCA), які буде розглянуто в наступних розділах.

12.4 Суть кореляційного аналізу

Величини, що характеризують різні властивості об'єктів, можуть бути незалежними або взаємопов'язаними. Розрізняють два види залежностей між величинами (факторами): функціональну та статистичну.

У разі функціональної залежності двох величин значенню однієї з них обов'язково відповідає одне або декілька точно визначених значень іншої величини. Функціональний зв'язок двох чинників можливий лише за умови, що друга величина залежить тільки від першої і не залежить ні від яких інших величин. Функціональний зв'язок однієї величини з множиною інших можливий в тому випадку, коли ця величина залежить тільки від цієї множини факторів. У реальних ситуаціях існує нескінченно велика кількість властивостей самого об'єкта і зовнішнього середовища, що впливають один на одного, тому такого роду зв'язки не існують, інакше кажучи, функціональні зв'язки є математичними абстракціями. Їх застосування допустимо тоді, коли відповідна величина переважно залежить від відповідних факторів.

Під час дослідження деякого об’єкта за допомогою інформаційно-вимірювальної системи багато вимірюваних параметрів потрібно вважати випадковими, що виключає прояв однозначної відповідності значень. Вплив загальних факторів, наявність об'єктивних закономірностей у поведінці об'єктів призводять лише до прояву статистичної залежності. Статистичною називають залежність, за якої зміна однієї з величин спричиняє зміну розподілу іншої, і ці інші величини набувають деяких значень з певними ймовірностями. Функціональну залежність у такому випадку потрібно вважати окремим випадком статистичної: значенню одного фактора відповідають значення інших факторів з імовірністю, що дорівнює одиниці. Однак на практиці такий розгляд функціонального зв'язку застосування не знайшов.

Більш важливим окремим випадком статистичної залежності є кореляційна залежність, що характеризує взаємозв'язок значень одних випадкових величин із середнім значенням інших, хоча в кожному окремому випадку будь-яка взаємопов'язана величина може набувати різних значень.

Якщо ж у взаємозалежних величин варіацію має тільки одна змінна, а інша є детермінованою, то такий зв'язок називають не кореляційним, а регресійним.

Кореляційний аналіз – *це сукупність методів виявлення залежності (кореляції) між двома або більше випадковими параметрами або процесами.*

Під кореляцією будемо розуміти статистичну залежність між двома випадковими величинами, що не має, взагалі кажучи, строгого функціонального характеру.

Зауважимо, що кореляційний аналіз не дозволяє визначити вид функціонального зв'язку між випадковими величинами, а тільки наявність або відсутність передбачуваної залежності, наприклад, лінійної, параболічної, експоненціальної і т. д.

Визначення виду функціонального зв'язку між величинами розглядається в регресійному аналізі, елементи якого і практичне використання розглянуто в попередніх підрозділах.

Назва «кореляційний аналіз» походить від латинського слова correlatio – узгодження, зв'язок, співвідношення, взаємозв'язок. Термін вперше введено Френсісом Гальтоном (англ. Francis Galton) у 1888 р.

Зазвичай досліджують парну кореляцію, тобто залежність між двома випадковими величинами (процесами), хоча можливі і більш складні ситуації, коли необхідно виявити наявність або відсутність зв'язків між трьома або більше випадковими величинами.

Ми обмежимося дослідженням парної кореляції.

Як відомо, зв'язок між двома випадковими величинами можна описати за допомогою двовимірної функції розподілу. Однак такий опис часто дуже складний, а для практичних цілей можна задовольнитися визначенням залежностей середніх значень.

Приклади величин, для яких вивчають наявність залежності, подано в таблиці 12.1.

Таблиця 12.1 – Приклади пар величини, між якими може існувати кореляція

a	b
Середній бал успішності навчальної групи з математики	Середній бал виконання вправи з комп'ютерного моделювання
Розсіювання точки падіння снаряда по дальності	Розсіювання точки падіння снаряда за бічним відхиленням
Вага студента	Успішність з фізвиховання

Необхідно перевірити: чи є зв'язок між величинами a і b?

Перевірка наявності (або відсутності) зв'язку, тобто кореляції, між випадковими величинами виконується таким чином.

Проводиться два експерименти, кожен з відповідною моделлю. У кожному експерименті n спостережень (нагадуємо, що комп'ютерний експеримент складається із спостережень, а спостереження - з реалізацій (прогонів) моделі, число яких розраховується з урахуванням необхідної точності та достовірності одержуваних результатів моделювання). Внаслідок експериментів знаходять дві множини значень вимірюваних параметрів a і b: a_i і b_i, $i=\overline{1,n}$.

З цих множин формуються пари: $(a_1,b_1),(a_2,b_2),...,(a_i,b_i),....,(a_n,b_n)$.

Кожна пара інтерпретується як координати випадкової точки в системі координат a, b.

Первинне дослідження можна провести графічно. Можливі такі варіанти розміщення точок на графіках (рис. 12.5).

З графіків можна візуально визначати за розташуванням даних, наскільки тісно вони корельовані.

Кажуть, що дві змінні позитивно корельовані, якщо у разі збільшення значень однієї змінної збільшуються значення іншої змінної (рис. 12.5, б).

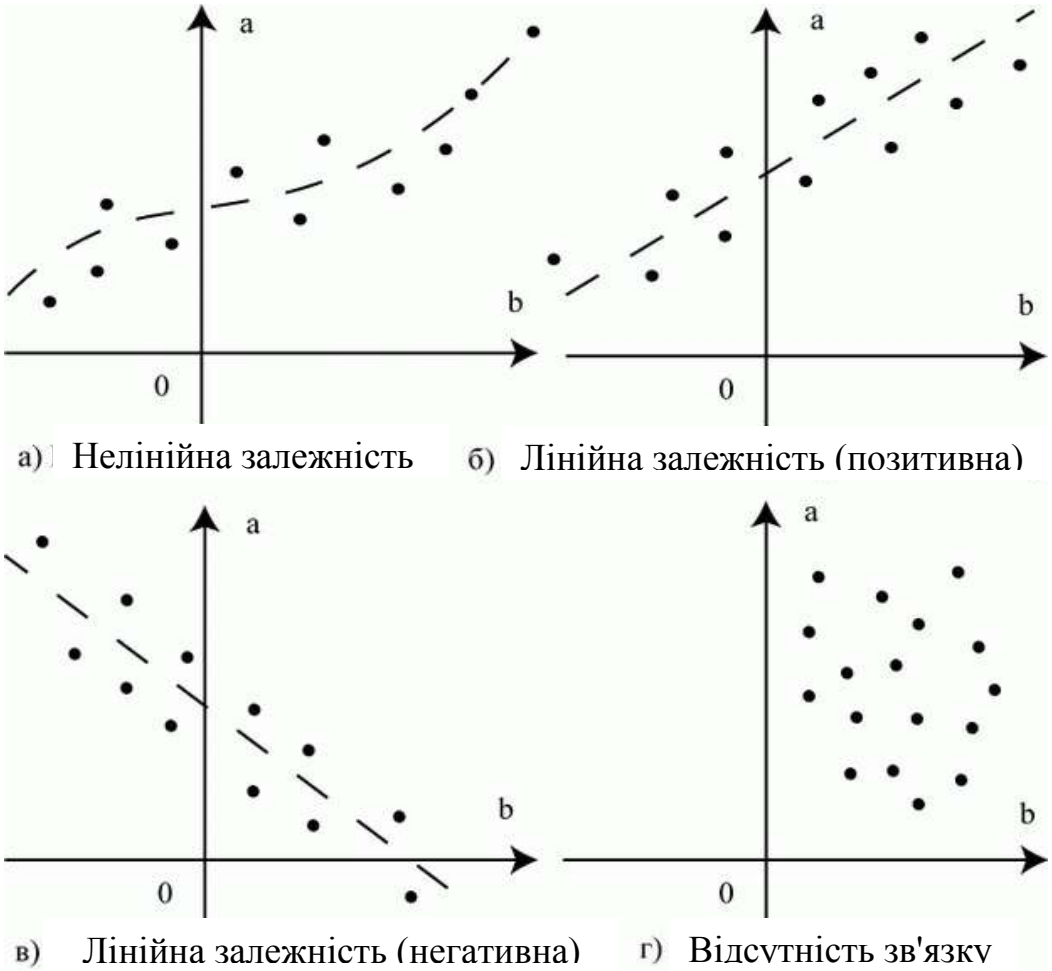
Дві змінні негативно корельовані, якщо у разі збільшення однієї змінної інша змінна зменшується (рис. 12.5, в).

Відсутність кореляції – спільної поведінки змінних – виявляється хаотичним нагромадженням точок, що виключає проведення якої-небудь апроксимувальної лінії (рис. 12.5, г.).

Але таке якісне дослідження недостатньо. Необхідно мати кількісну оцінку ступеня кореляції між величинами a і b.

Зазвичай вихідною для кореляційного аналізу є матриця ЕД (12.1).

Кореляційна залежність визначається різними параметрами, серед яких найбільшого поширення набули показники, що характеризують взаємозв'язок двох випадкових величин (парні показники): кореляційний момент, коефіцієнт кореляції.



а) Нелінійна залежність б) Лінійна залежність (позитивна)

в) Лінійна залежність (негативна) г) Відсутність зв'язку

Рисунок 12.5 – Графічне дослідження кореляції

Оцінка кореляційного моменту (коефіцієнта коваріації) двох варіант x_i та x_j обчислюється за вихідною матрицею $\mathbf{X}$

$$\xi_{jk} = \frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \mu_1(x_j))(x_{ik} - \mu_1(x_k)) \qquad (12.21)$$

Цей показник незручний для практичного застосування, оскільки має розмірність, що дорівнює добутку розмірностей варіант, і за його величиною важко судити про залежність параметрів.

Коефіцієнт коваріації r_{jk} нормованих випадкових величин називають коефіцієнтом кореляції, його оцінка

$$\tilde{r}_{jk} = \frac{1}{n}\sum_{i=1}^{n} u_{ij}u_{ik} \ . \qquad (12.22)$$

Значення коефіцієнта кореляції лежить в межах від -1 до 1. Якщо випадкові величини u_j і u_k незалежні, то коефіцієнт r_{jk} обов'язково дорівнює нулю, зворотне твердження неправильне. Коефіцієнт r_{jk} характеризує значимість лінійного зв'язку між параметрами:

- у разі $r_{jk} = 1$ значення u_j і u_k повністю збігаються, тобто значення параметрів набувають однакових значень. Інакше кажучи, має місце функціональна залежність: знаючи значення одного параметра, можна однозначно вказати значення іншого параметра;
- у разі $r_{jk} = -1$ величини u_j і u_k набувають протилежних значень. І в цьому випадку має місце функціональна залежність;
- у разі $r_{jk} = 0$ величини u_j і u_k практично не пов'язані одна з одною лінійним співвідношенням. Це не означає відсутності якихось інших (наприклад, нелінійних) зв'язків між параметрами;
- у разі $|r_{jk}| > 0$ і $|r_{jk}| < 1$ однозначного лінійного зв'язку величин u_j і u_k немає. І чим менша абсолютна величина коефіцієнта кореляції, тим меншою мірою за значеннями одного параметра можна передбачити значення іншого.

Використовуючи поняття коефіцієнта кореляції, матриці ЕД можна поставити у відповідність квадратну матрицю оцінок коефіцієнтів кореляції (кореляційну матрицю)

$$\boldsymbol{\rho} = \begin{vmatrix} \rho_{11} & \rho_{12} & \cdots & \rho_{1m} \\ \rho_{21} & \rho_{22} & \cdots & \rho_{2m} \\ . & . & . & . \\ \rho_{m1} & \rho_{m2} & \cdots & \rho_{mm} \end{vmatrix} \qquad (12.23)$$

Оцінка вірогідності коефіцієнта кореляції має бути визначена з необхідними точністю і достовірністю, які залежать від кількості реалізацій моделі. Знайдемо цей зв'язок.

Попереднє визначення $\tilde{r}$ здійснюється за даними пробного експерименту в кількості $n = 500...1000$ реалізацій моделі.

На підставі викладеного й у силу випадкового характеру досліджуваних величин ми можемо стверджувати лише таке: справжнє значення коефіцієнта кореляції r лежить в межах із заданою вірогідністю α,

$$\tilde{r} - \varepsilon \le r \le \tilde{r} + \varepsilon,$$

де ε – абсолютна величина помилки:

$$\varepsilon = t_\alpha \frac{1-\tilde{r}^2}{\sqrt{n}}, \quad t_\alpha = \Phi^{-1}(\frac{\alpha}{2}),$$

де Φ – функція Лапласа,

t_α – аргумент функції Лапласа.

На закінчення відзначимо, що якщо спільний розподіл випадкових величин x_j і x_k не є нормальним, то оцінка коефіцієнта кореляції може виступати як орієнтовна оцінка ступеня тісноти зв'язку цих величин.

12.5 Регуляризація матриці вхідних даних методом аналізу головних компонент

Метод PCA ґрунтується на тих міркуваннях, що часто ознаки досить сильно залежать одна від одної. Знаючи залежності і їх силу, можна виразити кілька ознак через одну, злиту воєдино, і працювати вже з більш простою моделлю образу. Метод розроблено К. Пірсоном (англ. Karl Pearson) в 1901 р. Застосовується в багатьох сферах, таких як розпізнавання образів, комп'ютерний зір, стиснення даних та ін. Обчислення головних компонент зводиться до обчислення власних векторів і власних значень коваріаційної матриці вихідних даних або до сингулярного розкладання матриці даних. Іноді метод головних компонент називають перетворенням Карунена-Лоева (англ. Karhunen-Loeve) або перетворенням Хотеллінга (англ. Hotelling transform). Завдання аналізу головних компонент може бути реалізовано на використанні таких чотирьох базових версій:

- апроксимація даних лінійними різноманіттями меншої розмірності;

- знаходження підпросторів меншої розмірності, в ортогональній проекції на які розкид даних (тобто середньоквадратичного відхилення від середнього значення) максимальний;

- знаходження підпросторів меншої розмірності, в ортогональній проекції на які середньоквадратична відстань між точками максимальна;

- побудова для заданої багатовимірної випадкової величини такого ортогонального перетворення координат, внаслідок якого кореляції між окремими координатами згорнуться в нуль.

Перші три версії оперують скінченними множинами даних. Вони еквівалентні і не використовують жодної гіпотези про статистичну природу (походження) даних. Четверта версія оперує випадковими величинами, в яких скінченні множини з'являються як вибірки з цього розподілу, а рішення трьох перших завдань – як наближення до «істинного» перетворення Карунена-Лоева. Виникає додаткове і не цілком тривіальне питання про точність цього наближення.

Подальше викладення суті методу PCA орієнтується на другу базову версію – знаходження підпросторів меншої розмірності, в ортогональній проекції на які розкид даних (тобто середньоквадратичне відхилення від середнього значення) максимальний. Цей підхід можна коротко описати таким чином. Нехай нам дано центрований набір векторів даних, в якому середнє арифметичне значення (математичне сподівання) кожної координати дорівнює нулю. Завдання – знайти таке ортогональное перетворення в нову систему координат, для якого були б правильні такі умови:

- вибіркова дисперсія даних уздовж першої координати максимальна (цю координату називають першою головною компонентою, рис. 12.6);

Рисунок 12.6 – Приклад побудови першої головної компоненти

- вибіркова дисперсія даних уздовж другої координати максимальна за умови ортогональності першій координаті (друга головна компонента);

- вибіркова дисперсія даних уздовж значень *k*-ої координати максимальна за умови ортогональності перших координат до *k*-1.

Обчислення головних компонент зводиться до обчислення власних векторів і власних значень коваріаційної матриці вихідних даних.

Контрольні питання та завдання

1. Чим відрізняється багатофакторна (множинна) модель регресії від однофакторної?
2. Як виглядає схем чорного ящика для багатофакторної моделі?
3. Запишіть рівняння моделі багатофакторної лінійної регресії.
4. Запишіть матрицю експериментальних даних для множинної лінійної регресії.
5. Який критерій використовується для мінімізації похибки моделі множинної лінійної регресії? Запишіть вираз для нього.
6. Запишіть матричну форму моделі багатофакторної лінійної регресії.
7. Запишіть вираз квадратичної багатофакторної моделі.
8. Як можна перейти від поліноміальної моделі багатофакторної регресії до лінійної моделі?
9. Запишіть вираз функції множинної регресії, що задається поліномом *r*–го степеня.
10. Покажіть шлях переходу від експоненціальної багатофакторної регресійної моделі до лінійної.
11. Як перейти від мультиплікативної багатофакторної моделі регресії до лінійної?
12. Запишіть узагальнене матричне рівняння для обчислення коефіцієнтів регресійної функції.
13. Які існують проблеми в обчислення коваріаційної матриці під час обчислення коефіцієнтів регресії?
14. Які наслідки «тягне» за собою мультиколінеарність коваріаційної матриці під час ідентифікації регресійної моделі?
15. Чим відрізняється функціональна залежність між двома вимірюваними параметрами від кореляційної?
16. Що таке кореляція?
17. Що називається кореляційним аналізом?
18. Назвіть приклади величин, між якими може існувати кореляція.
19. Які види кореляції між досліджуваними параметрами можуть існувати?
20. Запишіть формулу для обчислення коефіцієнта кореляції.
21. В яких межах може змінюватися коефіцієнт кореляції?
22. В чому полягає суть регуляризації матриці вхідних даних методом PCA?
23. До чого зводиться обчислення головних компонент коваріаційної матриці в методі PCA?

РОЗДІЛ 13
НАВЧАННЯ З УЧИТЕЛЕМ. РІДЖ -ЛАССО РЕГРЕСІЯ

13.1 Загальні передумови для застосування Рідж і Лассо регресії

Як було показано в Розділі 12, вираз для обчислення коефіцієнтів рівняння регресії, отриманий на основі мінімізації помилки за методом найменших квадратів, має вигляд:

$$A = (X^T X)^{-1} X^T Y. \qquad (13.1)$$

Нерідко доводиться стикатися з ситуацією, коли матриця $(X^T X)^{-1}$ є близькою до виродження. Тоді говорять про наявність мультиколінеарності. У таких ситуаціях МНК-оцінки формально існують, але мають «погані» статистичні властивості. Невелика зміна вихідних статистичних даних (додавання або вилучення невеликої порції спостережень) призводить до істотної зміни оцінок коефіцієнтів регресійної моделі, аж до зміни їх знаків.

В рівнянні (13.1) число обумовленості τ коваріаційної матриці Σ, $\Sigma = X^T X$ визначається за формулою:

$$\tau(X^T X) = \left\| X^T X \right\| \cdot \left\| X^T X \right\|^{-1} = \frac{\lambda_{\max}}{\lambda_{min}},$$

де $\lambda_{\max}$, $\lambda_{\min}$ – найбільше і найменше власні значення коваріаційної матриці Σ.

Якщо $\tau(\Sigma) >> 1$, то система рівнянь називається погано обумовленою.

Методи регресії Рідж і Лассо здійснюють регуляризацію параметрів і дозволяють подолати деякі недоліки методу найменших квадратів. Під регуляризацією розуміють додавання деякої додаткової інформації до умови з метою вирішити некоректно поставлену задачу. Ця інформація часто має вигляд штрафу за складність моделі. Вказані методи регресійного аналізу в багатьох випадках дозволяють підвищити точність прогнозу і покращити інтерпретованість моделі. Якщо залежність між змінними близька до лінійної, і число факторів *m* значно менше обсягу вибірки (*m*<<*n*), найімовірніше простий метод найменших квадратів буде давати гарні результати, і тоді в застосуванні методів Рідж і Лассо немає особливої потреби. Однак, якщо число вхідних факторів (ознак, впливів) *m* не набагато менше розміру вибірки *n*, зростає дисперсія прогнозу, а його точність спадає. Якщо ж *m*>*n*, метод найменших квадратів не дає однозначної відповіді і дисперсія прогнозу стає нескінченною. Методи

регуляризації часто дозволяють домогтися зменшення дисперсії прогнозу за рахунок незначного збільшення його зміщення. Як результат – точність прогнозу зростає.

Дуже часто зустрічаються випадки, коли в модель внесли завелику кількість m факторів, багато з яких можуть не впливати на значення відгуку. Якщо виключити такі змінні з вибірки, модель буде легше інтерпретувати. Методи регресії Рідж і Лассо становлять в цьому сенсі альтернативу відомій техніці зменшення числа факторів, наприклад, методу аналізу основних компонент PCA, розглянутому раніше. Внаслідок застосування цих методів коефіцієнт (вага) при деяких факторах лінійної моделі наближається до нуля (або дорівнює нулю).

Як уже відзначалось в попередніх розділах, для побудови оптимальної регресійної моделі і оцінення її адекватності використовують метод найменших квадратів (МНК, в англомовних джерелах RSS – residual sum of squares,), який ґрунтується на критерії мінімуму середньоквадратичного відхилення прогнозних точок моделі від експериментальних:

$$RSS = \sum_{i=1}^{n} (y_i - y_i^{th})^2 , \qquad (13.2)$$

де y_i – експериментальне значення,

y_i^{th} – прогнозне (теоретичне) значення залежної величини.

Крім того, для кращого орієнтування як отриманий регресійний прогноз значень вихідної величини від вхідних факторів, що впливають на неї, може використовуватися нормований коефіцієнт середньо-квадратичного відхилення (коефіцієнт детермінації) у вигляді:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - y_i^{th})^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \in [0,1], \qquad (13.3)$$

де $\bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i$ – вибіркове середнє.

Чим ближчий коефіцієнт (13.3) до одиниці, тим краще регресійна модель пояснює експериментальні дані.

Як було обґрунтовано в Розділі 11, в задачах регресійного аналізу в багатьох випадках потрібно застосовувати нормалізацію експериментальних даних з використанням таких величин:

$$\bar{x}_j = \frac{1}{n} \sum_{i=1}^{n} x_{ij} , \ \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i , \ \sigma_{x_j}^2 = \frac{1}{n-1} \sum_{i=1}^{n} (x_{ij} - \bar{x}_j)^2 , \ \sigma_y^2 = \frac{1}{n-1} \sum_{i=1}^{n} (y_i - \bar{y})^2 , \qquad (13.4)$$

де $j = \overline{1,m}$.

Дотримуючись виразів (13.4), шляхом центрування і нормування отримують стандартизовані елементи матриці експериментальних даних:

$$v_i = \frac{y_i - \bar{y}}{\sigma_y^2},\ u_{ij} = \frac{x_{ij} - \bar{x}_j}{\sigma_{x_j}^2},\ i = \overline{1,n},\ j = \overline{1,m}. \tag{13.5}$$

Якщо ввести матричні позначення

$$\mathrm{V} = [v_i]_n, \quad U = \lfloor u_{ij} \rfloor_{n \times m}, \tag{13.6}$$

то вираз (13.1) обчислення коефіцієнтів регресії для стандартизованих даних запишеться як:

$$A = (U^T U)^{-1} U^T V. \tag{13.7}$$

13.2 Ridge регресія

Гребенева регресія або рідж-регресія (англ. Ridge regression) – один з методів зниження розмірності. Застосовується для боротьби з надмірністю даних, коли незалежні змінні корелюють одна з одною, внаслідок чого виявляється нестійкість оцінок коефіцієнтів багатофакторної лінійної регресії.

Критерій якості регресійної моделі в Ridge регресії дуже схожий на критерій мінімуму середньоквадратичного відхилення, за винятком додавання «гребеня».

$$\sum_{i=1}^{n} (y_i - A_0 - \sum_{j=1}^{m} A_j x_{ij})^2 + \beta \sum_{j=1}^{m} A_j^2 = RSS + \beta \sum_{j=1}^{m} A_j^2, \tag{13.8}$$

де β – коефіцієнт регуляризації, параметр, який в практичних застосуваннях є додатним числом, що зазвичай лежить в межах [0,1].

Рівняння (13.8) визначає два різних критерії. Як і у випадку з методом найменших квадратів, Ridge регресія шукає оцінки коефіцієнтів, які добре підходять для даних, мінімізуючи RSS. Однак другий член, який називається штрафом за скорочення, тим менший, чим ближче до нуля $A_1,...,A_m$, тому він приводить до прямування оцінок коефіцієнтів A_j до нуля.

Для стандартизованих експериментальних даних Рідж-оцінка невідомих коефіцієнтів регресії запишеться рівнянням:

$$A_R = (U^T U + \beta I)^{-1} U^T V. \tag{13.9}$$

У матриць U^TU і $(U^TU+\beta I)$ власні вектори збігаються, а власні значенням різняться на β. Тому число збільшення обумовленості для матриці $U^TU+\beta I$ дорівнює:

$$\tau(U^TU+\beta I)=\frac{\lambda_{\max}+\beta}{\lambda_{min}+\beta}. \qquad (13.10)$$

Вираз (13.10) показує, що за збільшення β число обумовленості τ коваріаційної матриці U^TU зменшується, тобто із зростанням β зростає стійкість розв'язку регресійної задачі.

Додавання «гребеня» у вигляді діагональної матриці βI до U^TU перетворює «погано обумовлену» матрицю в «добре обумовлену» матрицю $U^TU+\lambda I$. Така заміна збільшує власні значення матриці U^TU, але не змінює її власні вектори. Це дозволяє уникнути труднощів, пов'язаних з оберненням вироджених матриць. Проте на відміну від оцінки коефіцієнтів A МНК методом Рідж-оцінка коефіцієнтів A_R є зміщеною.

На відміну від методу найменших квадратів, який генерує лише один набір оцінок коефіцієнтів, Ridge регресія буде виробляти новий набір оцінок коефіцієнтів для кожного значення β [6]. Зауважимо, що в (13.8) штраф застосовується до $A_1,...,A_m$, але не до A_0. Таким чином, Ridge-оцінка є МНК-оцінкою з обмеженням норми L_2 можливих рішень (сферичне обмеження на параметри).

13.3 Lasso регресія

У Ridge регресії є один недолік. На відміну від методів вибору кращої підмножини факторів, таких, наприклад, як метод PCA чи метод сингулярного розкладу матриці SVD (входів регресійної моделі), які зазвичай визначає моделі, що містять тільки підмножини змінних, Ridge регресія буде містити всі m факторів у кінцевій моделі. Штраф β стискає всі коефіцієнти до нуля, але він не буде встановлювати жодного з них точно в нуль (крім випадку, коли $\beta = \infty$). Це не зменшує точність прогнозу, але може затруднити інтерпретацію моделі у випадку, коли число факторів m досить велике.

На відміну від Ridge регресії, Lasso регресія має дещо інше обмеження [7].

$$\sum_{i=1}^{n}(y_i-A_0-\sum_{j=1}^{m}A_jx_{ij})^2+\beta\sum_{j=1}^{m}\left\|A_j\right\|=RSS+\beta\sum_{j=1}^{m}\left\|A_j\right\|. \qquad (13.11)$$

Коефіцієнт β множиться на лінійну норму L_2 вектора $A_1,...,A_m$, тоді як в Ridge регресії використовується квадратична норма (рис. 13.1).

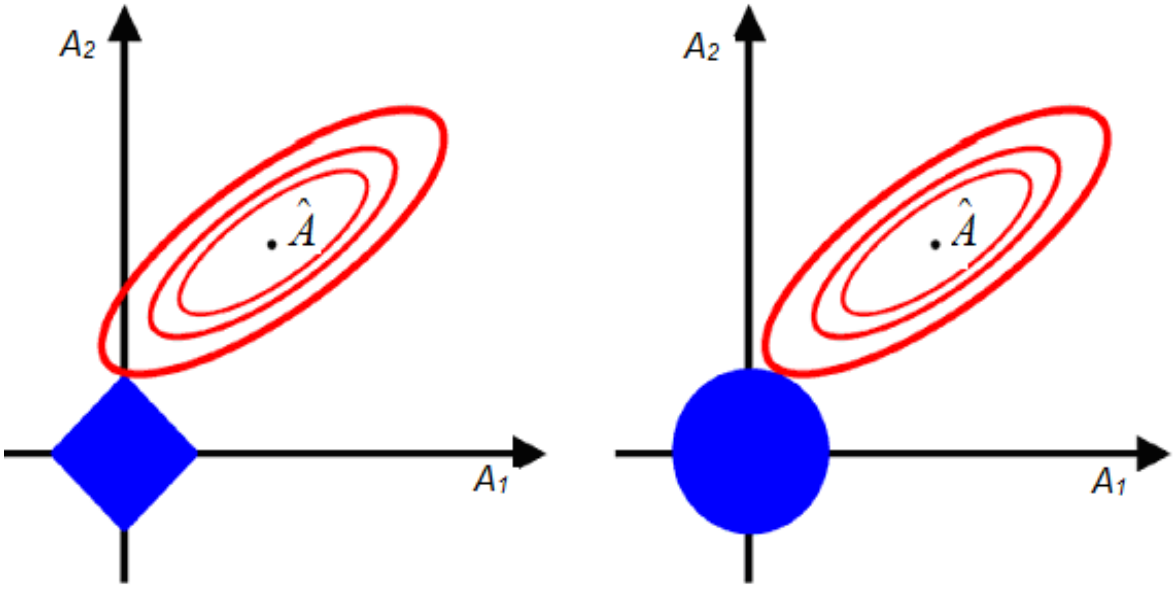


Рисунок 13.1 – Обмеження на норму ваг: зліва – лінійна L_1-норма (Lasso регресія), справа – квадратична L_2-норма (Ridge регресія)

З геометричного погляду задача мінімізації помилки за умови дотримання накладених умов на штрафи β полягає в тому, щоб знайти точку дотику лінії, що відображає функцію помилки, з фігурою, що відображає обмеження на β.

З рисунка 13.1 інтуїтивно зрозуміло, що в разі Лассо-регресії ця точка з великою ймовірністю буде перебувати на кутах ромба, тобто лежати на осі, тоді як у випадку Рідж-регресії таке відбувається дуже рідко. Якщо точка перетину лежить на осі, один з коефіцієнтів буде дорівнює нулю, а значить, значення відповідної незалежної змінної не буде враховуватися. Позитивним (в плані інтерпретації моделі) результатом такої заміни норми є той факт, що Lasso, на відміну від Ridge регресії, не тільки здійснює регуляризацію, але і прирівнює деякі з коефіцієнтів до нуля за досить великого значення β [8]. Тобто додатково здійснює вибір підмножини змінних, що дозволяє легше інтерпретувати модель.

13.4 Вибір значення β для Ridge і Lasso

Спочатку вся вибірка випадково розділяється на Q блоків. Один з блоків розглядається як контрольна вибірка, а решта Q-1 в сукупності становлять навчальну вибірку. На практиці Q зазвичай вибирають таким, що дорівнює 5 або 10. Далі береться вектор $\vec{\beta} = (\beta_1, \beta_2, ..., \beta_t, .., \beta_m)$ з деяким кроком, і для кожного зі значень β_t за навчальною вибіркою будується регресійна модель. Для кожної моделі обчислюється помилка прогнозу, тобто сума квадратів залишків регресії RSS

$$RSS(q,\beta_t)=\sum_{i=1}^{n}(y_i-\sum_{j=0}^{m}\hat{A}_j(q,\beta_t)x_{ij})^2, \quad (13.12)$$

де $q=\overline{1,Q}$ – номер блока, вибраного як контрольна вибірка. Далі обчислюється середнє значення цієї помилки по всіх блоках:

$$MSE(\beta_t)=\frac{1}{Q}\sum_{q=1}^{Q}RSS(q,\beta_t). \quad (13.13)$$

Як відповідне β вибирається таке β_t, за якого $MSE(\beta_t)$ буде мінімальним [9].

Ridge регресія – удосконалення лінійної регресії з підвищеною стійкістю до помилок, що накладає обмеження на коефіцієнти регресії для отримання більш наближеного до реальності результату. Окрім того, цей результат набагато простіше інтерпретувати. Регресія цього типу застосовується як метод для боротьби з надмірністю даних, коли незалежні змінні корелюють один з одним (мультиколінеарність).

Lasso регресія подібна до Ridge, за винятком того, що коефіцієнти регресії можуть дорівнювати нулю (частина ознак в цьому випадку виключається з моделі).

Обидва методи успішно вирішують проблеми мультиколінеарності, перенавчання та зменшують розкид коефіцієнтів. Ridge регресія використовує всі ознаки, намагаючись «вичавити максимум» з усієї наявної інформації.

Lasso проводить відбір ознак, що є кращим для випадку, коли серед ознак є шумові або вимірювання ознак пов'язано з відчутними витратами. Отримані методом крос-валідації значення β дають найкращі результати в процесі отримання прогнозних значень моделі.

Лістинг програми мовою Python для реалізації Ridge-регресії

```
# імпорт бібліотек
from sklearn.datasets import make_regression
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split

# генерація даних для X і y
X, y = make_regression(n_samples=10000, noise=100, random_state=0)

# розподіл даних на train і test
```

```
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3,
random_state=3)

ridge_regression = Ridge(alpha=0.1)  # alpha — величина регуляризации

# навчання
ridge_regression.fit(train_X, train_y)

# прогнозування результату
print(ridge_regression.predict(test_X))

# виведення точності прогонозу
print(ridge_regression.score(test_X, test_y))
```

Точність прогнозу для даного датасету и параметрів:

```
>>> 0.8171822749108134
```

Лістинг програми мовою Python для реалізації Lasso-регресії

```
# імпорт бібліотек

from sklearn.datasets import make_regression
from sklearn.linear_model import Lasso
from sklearn.model_selection import train_test_split

# генеруєм дані для X і y
X, y = make_regression(n_samples=10000, noise=100, random_state=0)

# розподіл даних на train і test
train_X, test_X, train_y, test_y = train_test_split(X, y, test_size=0.3,
random_state=3)

lasso_regression = Lasso(alpha=0.1) # alpha — величина регуляризации

# навчання
lasso_regression.fit(train_X, train_y)

# прогнозування результату
print(lasso_regression.predict(test_X))

# виведення точності прогнозу
print(lasso_regression.score(test_X, test_y))
```

Точність прогнозу для даного датасету і параметрів:

```
>>> 0.8173906804156383
```

Контрольні питання та завдання

1. В якому випадку кажуть, що МНК-оцінки коефіцієнтів регресії мають погані статистичні властивості?
2. Що таке мультиколінеарність матриці?
3. Що таке число обумовленості коваріаційної матриці?
4. В яких випадках матриця вважається погано обумовленою?
5. Що називається регуляризацією параметрів?
6. В яких випадках використовують Ridge і Lasso регресію?
7. Запишіть формули для критерію МНК і коефіцієнта детермінації.
8. Як здійснюється стандартизація матриці експериментальних даних?
9. Запишіть формулу критерію якості регресійної моделі в Ridge регресії.
10. Запишіть рівняння для Ridge-оцінки коефіцієнтів регресії.
11. Запишіть формулу числа, що характеризує обумовленість коваріаційної матриці в рівнянні Ridge-регресії.
12. Як впливає Ridge-регресія на коефіцієнти прогнозу?
13. Запишіть формулу критерію якості регресійної моделі в Lasso-регресії.
14. Охарактеризуйте відмінності між Ridge і Lasso регресіями.

РОЗДІЛ 14
НЕЙРОМЕРЕЖЕВІ ТЕХНОЛОГІЇ

14.1 Основні засади побудови і роботи штучних нейромереж

14.1.1 Коротка характеристика штучних нейронних мереж

У наші дні зростає необхідність в системах, які здатні не тільки виконувати одного разу запрограмовану послідовність дій над заздалегідь визначеними даними, як це робить комп'ютер, але й здатні самі аналізувати інформацію, що надходить заново, знаходити в ній закономірності, виробляти прогнозування тощо. У цьому напрямку застосувань найкращим чином зарекомендували себе так звані нейронні мережі – самонавчальні системи, що імітують діяльність людського мозку.

Порівняно з традиційними комп'ютерами з архітектурою фон Неймана, мозок людини має багато інших якостей:

- низьке енергоспоживання;
- толерантність до помилок;
- адаптивність;
- здатність до навчання й узагальнення;
- розподілене подання інформації і паралельні обчислення.

Механізм природного мислення базується на збереженні інформації у вигляді образів.

Базовим елементом мозку людини є нейрони – специфічні клітини, здатні запам'ятовувати і застосовувати попередній досвід до наступних дій, що докорінно відрізняє їх від решта клітин тіла.

Кора головного мозку людини є протяжною, утвореною нейронами поверхнею товщиною від 2 до 3 мм із площею близько 2200 см^2. Кора головного мозку містить близько 10^{11} нейронів. Кожен нейрон зв'язаний з великою кількістю (10^3 – 10^4) інших нейронів. Загалом мозок людини містить приблизно від 10^{14} до 10^{15} взаємозв'язків. Потужність людського розуму залежить як від кількості нейронів, так і від різноманіття зв'язків між ними.

Окремий нейрон є складним, має свої складові, підсистеми та механізми керування і передає інформацію через велику кількість електрохімічних зв'язків. Налічують близько сотні різних класів нейронів. Разом нейрони та з'єднання між ними формують недвійковий, нестійкий та несинхронний процес, що різниться від процесу обчислень традиційних комп'ютерів.

Штучні нейромережі моделюють лише найголовніші елементи природного мозку, але спонукають науковців та розробників до пошуку нових шляхів розв'язування проблеми. **Штучна нейронна мережа (ШНМ)** *– це математична або програмна чи апаратна модель, побудована за принципом функціонування біологічних нейронних мереж клітин живого організму.*

Штучні нейромережі є моделями нейронної структури мозку, який здатен сприймати, обробляти, зберігати та продукувати інформацію. Особливістю мозку також є навчання та самонавчання на власному досвіді. Адаптивні системи на основі штучних нейронних мереж дозволяють з успіхом вирішувати проблеми розпізнавання образів, виконання прогнозів, оптимізації, асоціативної пам'яті і керування. Біологічні нейрони передають один одному хімічні сигнали за допомогою електричних імпульсів. Внаслідок такої активності у людини з’являється мислення та відчуття. Штучний інтелект і когнітивне моделювання намагаються імітувати деякі властивості біологічних нейронних мереж.

Штучна нейронна мережа за своєю будовою являє собою систему з’єднаних і взаємодіючих між собою штучних нейронів. Кожен штучний нейрон ШНМ має справу тільки з сигналами, які він періодично отримує, і сигналами, які він періодично посилає іншим штучним нейронам.

ШНМ працює таким чином: на входи нейронів надходять сигнали, які підсумовуються. При цьому враховується синаптична вага, тобто значимість кожного з входів. Вихідні сигнали одних нейронів надходять на входи інших нейронів. Вага кожного такого зв’язку може бути позитивною (збуджувальні зв’язки) або негативною (гальмівні зв’язки). Вони визначають обчислення нейронної мережі, а значить її пам’ять та поведінку.

Для того, щоб ШНМ виконувала складні завдання, її необхідно навчити. На рис. 14.1 наведено приклад навчання тришарової штучної мережі, яка навчається прогнозувати тиск, що утворюється на поверхні деякого фізичного тіла залежно від величини температури, величини і швидкості деформації, які на нього впливають.

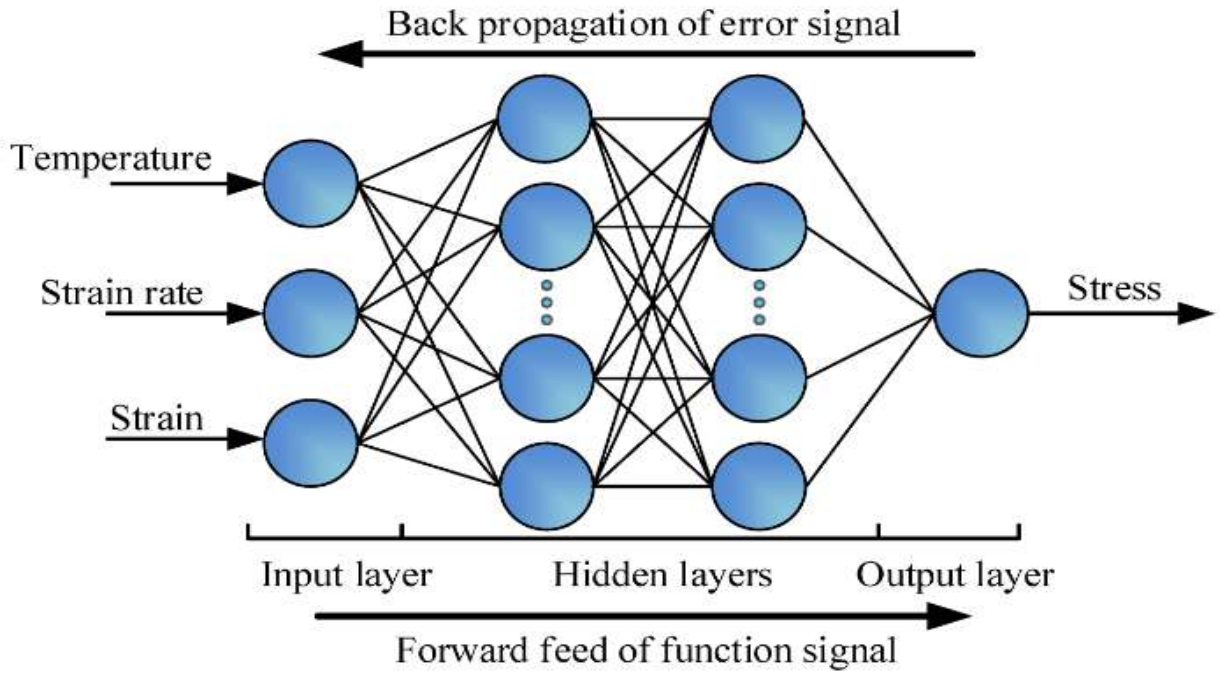


Рисунок 14.1 – Схема функціонування ШНМ

Штучні нейронні мережі дозволяють реалізувати створення паралельних мереж, їх навчання та вирішення інтелектуальних завдань, не використовуючи традиційного програмування.

14.1.2 Біологічний нейрон

Нервова система та мозок людини складаються з нейронів, з'єднаних між собою нервовими волокнами. Нервові волокна здатні передавати електричні імпульси між нейронами. Всі процеси передачі подразнень від нашої шкіри, вух та очей до мозку, процеси мислення та управління діями – все це реалізовано в живому організмі як передача електричних імпульсів між нейронами. Будову біологічного нейрона зображено на рис. 14.2. Кожен нейрон складається з *соми* (тіла клітини) та відростків нервових волокон двох типів – вхідних (*дендритів*), які сприймають імпульси, і вихідного (*аксона*), через який нейрон може передавати імпульс до інших нейронів. Аксон контактує з дендритами інших нейронів через спеціальні утворення – синапси, які впливають на силу імпульсу.

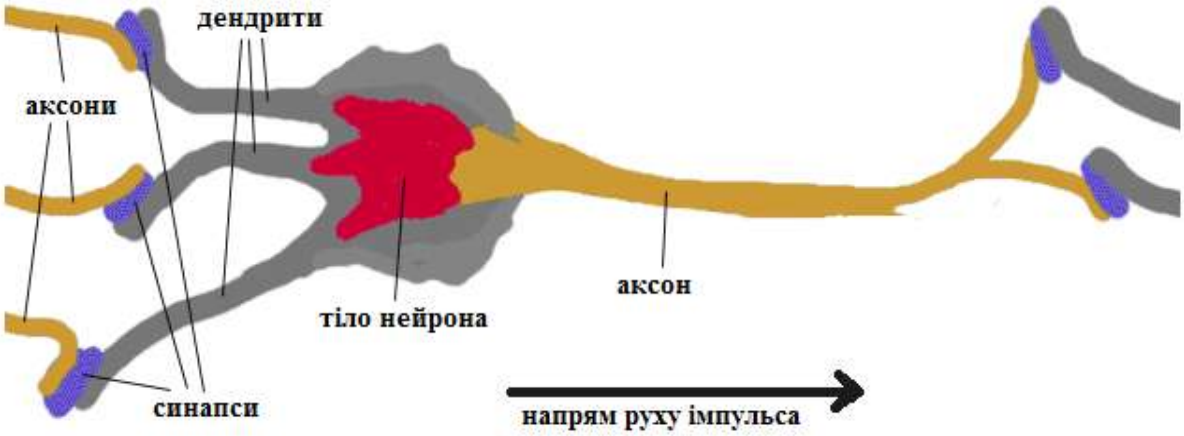


Рисунок 14.2 – Схема будови біологічного нейрона

Нейрон отримує сигнали (імпульси) від інших нейронів через дендрити (приймачі) і передає сигнали, згенеровані тілом клітки, вздовж аксона (передавач), який наприкінці розгалужується на волокна (strands). На закінченнях волокон знаходяться синапси (synapses). Під час проходження імпульсу від аксона до дендрита його сила змінюється у певну кількість разів, і цю зміну називають вагою синапсу або вагою міжнейронного зв’язку. Імпульси, що одночасно надходять до нейрона від дендритів, підсумовуються. Якщо сумарний імпульс перевищує певний поріг, нейрон збуджується, формує власний імпульс і передає далі по аксону. Ваги синапсів в процесі роботи нейронної мережі можуть змінюватися з часом, і таким чином змінювати й поведінку відповідних

нейронів. Синапс є функціональним вузлом між двома нейронами (волокно аксона одного нейрона і дендрит іншого). Коли імпульс досягає синаптичного закінчення, там продукуються хімічні речовини – нейротрансмітери. Нейротрансмітери проходять через синаптичну щілину і залежно від типу синапсу, збуджують або гальмують здатність нейрона-приймача генерувати електричні імпульси. Результативність синапсу налаштовується сигналами, які проходять скрізь нього, тому синапси навчаються відповідно до активності процесів, у яких вони беруть участь. Нейрони взаємодіють за допомогою короткої серії імпульсів, повідомлення передається за допомогою частотно-імпульсної модуляції.

На основі наведеної схеми біологічного нейрона математичну модель процесу його роботи можна подати у вигляді рис. 14.3.

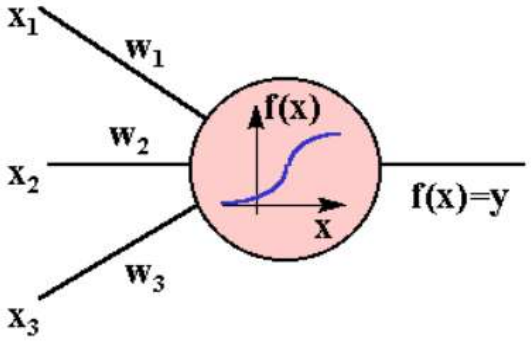


Рисунок 14.3 –Математична модель біологічного нейрона

Математична модель нейрона з трьома входами (дендритами), зображена на рис. 14.3, відповідає схемі будови біологічного нейрона на рис. 14.2. Ваги w_1, w_2, w_3 відображають дію синапсів дендритів, які збільшують (підсилюють) або зменшують (гальмують) силу вхідних імпульсів в деяке число разів. Імпульси сили x_1, x_2, x_3 після проходження синапсів та дендритів до нейрона надходять у вигляді зважених добутків w_1x_1, w_2x_2, w_3x_3. Отриманий сумарний імпульс $\mathbf{X}=w_1x_1+w_2x_2+w_3x_3$ нейрон перетворює згідно з деякою передатною функцією $f(\mathbf{X})$ і генерує вихідний імпульс з силою $y = f(\mathbf{X}) = f(w_1x_1+w_2x_2+w_3x_3)$.

Таким чином, математично нейрон повністю описується вагами своїх зв'язків w_k та передатною функцією $f(x)$. Отримавши вектор чисел x_k на входах, нейрон видає деяке число y на виході.

Оскільки нейрофізіологія надає дослідникам розширене розуміння дії нейронів, а технологія обчислень постійно вдосконалюється, то розробники нейромереж мають широкий простір для вдосконалення моделей нейрона і, як наслідок, моделей біологічного мозку.

14.1.3 Штучний нейрон

Штучний нейрон (ШН) для нейронних мереж є базовим модулем і моделює основні функції природного нейрона (рис. 14.4).

Під час функціонування на нейрон одночасно надходить багато вхідних сигналів. Кожен вхід нейрона має власну синаптичну вагу, яка впливає на функцію суматора елемента обробки. Ваги являють собою міру сили вхідних зв'язків і моделюють різноманітні синаптичні сили біологічних нейронів. Ваги суттєвого входу підсилюються, а вага несуттєвого входу примусово зменшується, визначаючи інтенсивність вхідного сигналу. Ваги можуть змінюватись відповідно до навчальних прикладів, топології мережі та навчальних правил.

Вхідні сигнали x_n зважуються ваговими коефіцієнтами зв'язків w_n, підсумовуються, і за функціональним законом передатної ланки генерують вихідний результувальний сигнал, що подається на вихід.

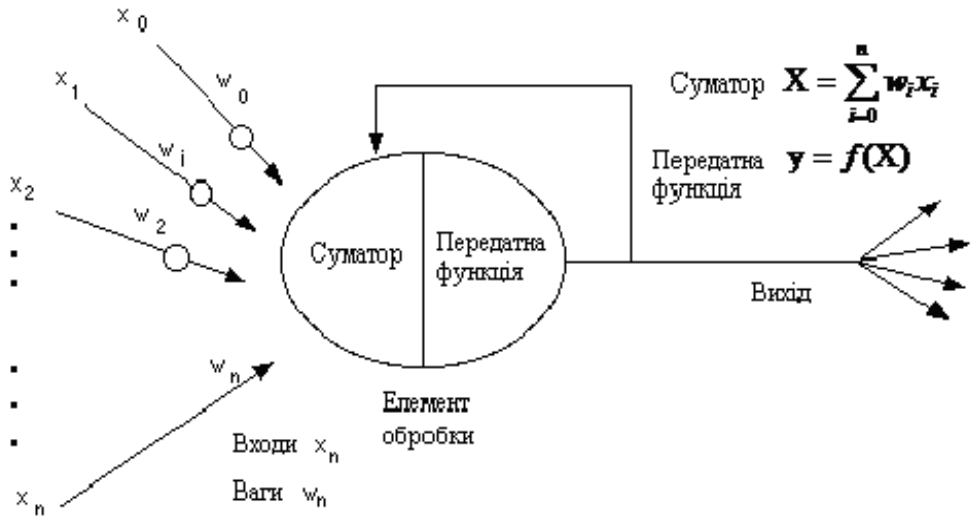


Рисунок 14.4 – Узагальнена схема базового штучного нейрона

У програмних реалізаціях штучні нейрони часто відомі під назвами «елементи обробки» або «процесори», і їм надають більше функціональних можливостей, ніж тих, які відображені в схемі базового штучного нейрона на рис. 14.4. На рисунку 14.5 наведено більш докладну схему штучного нейрона.

Функція суматора може мати різноманітні форми, такі як вибір мінімуму, максимуму, середнього арифметичного, добутку або обчислення за іншим алгоритмом. В багатьох програмних реалізаціях використовують власні реалізації суматора, які програмуються мовами вищого рівня, таких як C, C++, Python, Java.

Перед подачею на передатну функцію вхідні сигнали та вагові коефіцієнти можуть комбінуватися різними способами. Алгоритми для комбінування входів нейронів визначаються відповідно до мережної архітектури та парадигми.

У деяких нейромережах суматор виконує додаткову обробку, так звану функцію активації, яка зміщує вихід функції суматора в часі. Цю

функцію краще використовувати як компоненту мережі в цілому, а не як окремий нейрон. Часто ця функція взагалі не застосовується

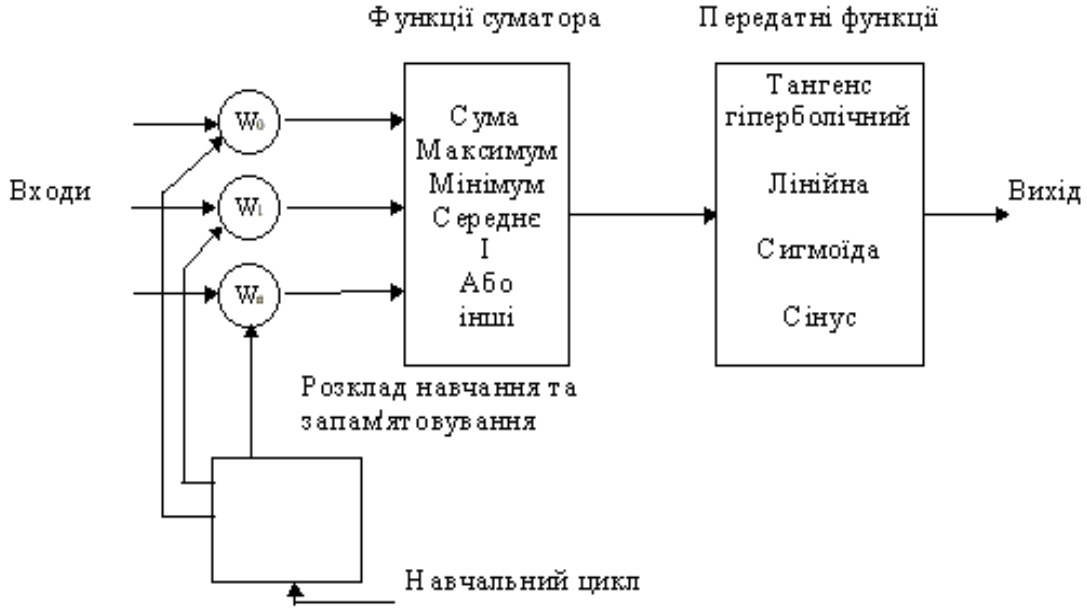


Рисунок 14.5 – Модель штучного нейрона як «елемента обробки»

Результат функції суматора перетворюється у вихідний сигнал через передатну функцію. У передатній функції для визначення виходу нейрона загальна сума порівнюється з певним порогом (зазвичай, це діапазон [0, 1] або [-1,1], або інший) за допомогою певного алгоритму.

Зазвичай застосовують нелінійні передатні функції, оскільки лінійні функції обмежені, і вихід є пропорційним до входу. Застосування лінійних передатних функцій було проблемою у ранніх моделях мереж, і їх обмеженість та недоцільність була доведена в книзі Мінскі та Пейперта «Перцептрони».

У наявних нейромережах для передатної функції використовують сигмоїду, синус, гіперболічний тангенс тощо. На рисунку 14.6 зображено типові передатні функції штучного нейрона .

На рис. 14.6, а) зображено **жорстку (порогову) передатну функцію**, яка використовується в штучному нейроні. ШН з таким виходом може видавати або 0 і 1, або 1 і -1, або інші числові комбінації. Він використовується здебільшого в задачах логічної класифікації і дозвоює синтезувати будь-які логічні схеми класифікаторів на основі передатної функції з такою нелінійністю. Функція досить просто обчислюється, завдяки чому нейрони з такою нелінійністю не потребують значних обчислювальних витрат. Порогова функція надмірно спрощена і ШН з такою передатною функцією не дозволяють моделювати схеми з неперервними сигналами. Ця функція не має першої похідної, що ускладнює застосування градієнтних методів для навчання нейромереж, які використовують ШН такого типу.

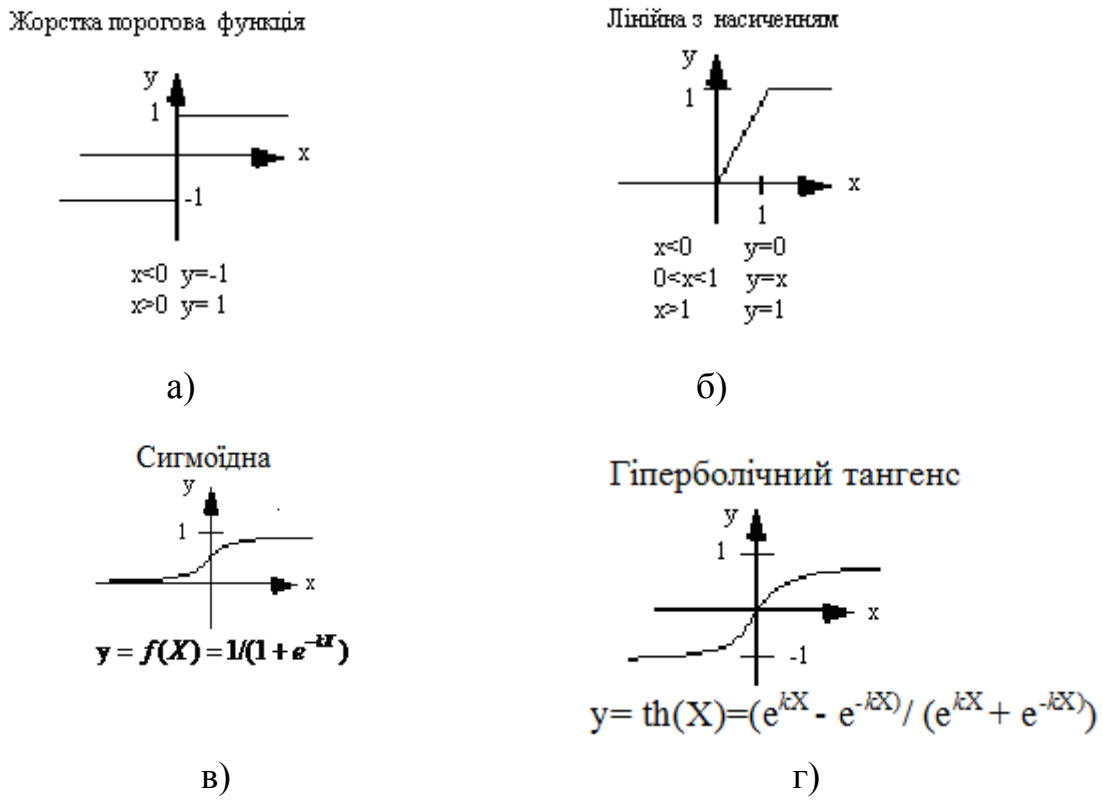


Рисунок 14.6 – Типові передатні функції нейрона

Лінійна передатна функція з насиченням, подана на рис. 14.6, б), пропорційна вхідному сигналу всередині заданого діапазону (на рисунку це діапазон [0, 1]) і діє як жорсткий обмежувач поза межами цього діапазону (на рисунку обмеження за порогом з рівнем y=1). Це лінійна функція, яка відсікається до мінімальних та максимальних значень, роблячи її нелінійною (рис. 14.6, б). Ця функція обчислюється досить легко, але її перша похідна має розриви першого роду в точках (x,y) =θ і (x,y)=1, що ускладнює алгоритм навчання мереж, які використовують ШН такого типу.

На рис. 14.6, в) зображено сигмоїдну передатну функцію (інакше її називають S-подібною або ще логістичною функцією). Вона описується формулою:

$$y = f(X) = 1/(1 + e^{-kX}), \qquad (14.1)$$

і має діапазон $0 < f(X) < 1$.

У виразі (14.1) k – додатна константа, що впливає на розтяг функції: збільшення k стискає функцію, вироджуючись у горизонтальну лінію на рівні 0,5 за k=0. За умови $k \to \infty$ сигмоїда наближається до функції Хевісайда (одиничної сходинки). Цей коефіцієнт використовують як параметр підсилення. Для малих вхідних сигналів він дає можливість забезпечити крутий кут нахилу, який буде досить швидко змінювати функцію виходу для значного підсилення сигналу. Для великих вхідних

сигналів можна зменшувати кут нахилу і, відповідно, підсилення. Завдяки цьому мережа може приймати великі сигнали і водночас залишатися чутливою до слабких змін сигналу.

Популярність сигмоїдної функції зумовлюють такі її властивості:

- здатність підсилювати слабкі сигнали сильніше, ніж великі, і опиратися «насиченню» від потужних сигналів;
- монотонність і диференційованість на всій осі абсцис;
- простий вираз для похідної:

$$f'(\mathrm{X}) = \frac{ke^{-kX}}{(1+e^{-kX})^2} = \frac{e^{-kX}k\dfrac{1}{(1+e^{-kX})}}{(1+e^{-kX})} = kf(\mathrm{X})\frac{e^{-kX}}{(1+e^{-kX})} = kf(\mathrm{X})(1-f(\mathrm{X})), \qquad (14.2)$$

що дозволяє спростити реалізацію алгоритмів навчання нейронної мережі градієнтними методами (наприклад, метод зворотного поширення помилки).

На рис. 14.6, г) зображено передатну функцію ШН у вигляді **гіперболічного тангенса**:

$$y = th(X) = (e^{kX} - e^{-kX})/(e^{kX} + e^{-kX}), \qquad (14.3)$$

де $k > 0$ – коефіцієнт ширини S-кривої по осі абсцис (зазвичай k=1).

Ця передатна функція також часто застосовується для мереж із неперервними сигналами. Вона є симетричною відносно точки (0,0), що є її перевагою порівняно з сигмоїдною передатною функцією. Перша похідна гіперболічного тангенса є незперервною і записується через саму функцію.

Для різних нейромереж можуть вибиратись інші передатні функції, серед яких дуже часто застосовуються дзвоноподібні радіально-базисні функції (RBF):

$$\mathrm{y} = \exp(-k(X - X_0)^2, \qquad (14.4)$$

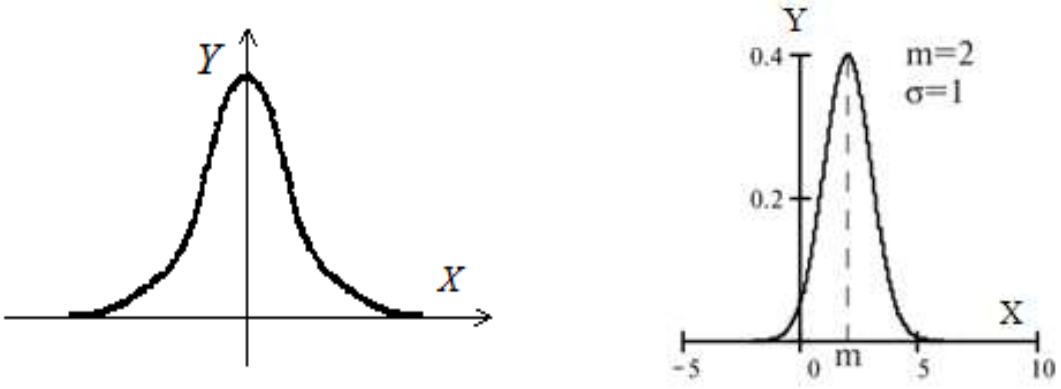
Особливістю їх є те, що вони залежать лише від відстані від центрального вектора, радіально симетричні відносно цього вектора, звідси і отримали назву радіальної базисної функції. Вигляд таких функцій може задаватися різними виразами. Нормою відстані для таких функцій вважається Евклідова відстань. Зазвичай радіально-базисна функція вважається гаусовою, яка подається виразом:

$$\mathrm{y} = f(\mathrm{X}) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(\mathrm{X}-m)^2}{2\sigma^2}\right), \qquad (14.5)$$

де m – математичне сподівання вхідного сигналу, середньоквадратичне відхилення вхідного сигналу.

Радіально-базисна функція застосовується у тих випадках, коли реакція ШН має бути максимальною для певного значення входу X.

Загальний вигляд RBF зображено на рис. 14.7, а), а гаусової функції – на рис.14.7, б).



Функція Гаусса

а) б)

Рисунок 14.7 – Вигляд радіально-базисних передатних функцій нейрона

Після обробки сигналу нейрон на виході має результат передатної функції, який надходить на входи інших нейронів або до зовнішнього з'єднання, як це передбачається структурою нейромережі.

Вибір виду передатної функції ШН визначається такими факторами:

1. Специфікою задачі;

2. Зручністю реалізації на комп'ютері у вигляді електричної схеми або іншим способом;

3. Алгоритмом навчання: деякі алгоритми накладають обмеження на вигляд функції активації, їх треба враховувати.

14.1.4 Сфери застосування штучних нейронних мереж

Сфери застосування нейромереж для обробки даних визначаються їх перевагами і недоліками.

1. Висока надійність роботи. Інформація в ШНМ кодується і запам'ятовується не в окремих елементах пам'яті, а в розподілі зв'язків між нейронами і в їх силі, тому стан кожного окремого нейрона визначається станом багатьох інших нейронів, пов'язаних з ним. Тому втрата одного або декількох зв'язків не має істотного впливу на результат роботи системи загалом, що забезпечує її високу надійність

2. Висока «природна» завадостійкість і функціональна надійність стосуються як спотворених (зашумлених) потоків інформації, так і відмов окремих нейронів. Цим забезпечуються висока оперативність і достовірність обробки інформації, а просте донавчання та перенавчання мереж дозволяють за зміни зовнішніх чинників своєчасно здійснювати перехід на новий рівень вирішуваних завдань.

Недоліки ШНМ

1. «Логічна непрозорість» нейронних мереж, тобто неможливість побудувати зрозумілу людині логічну конструкцію, яка б відтворювала дії мережі.

2. Неможливість інтегрування знань в нейронну мережу та витягування їх з неї, опис їхніх параметрів. Кількість нейронів та їх зв'язки, параметри навчального процесу можуть бути вибрані тільки експериментально.

До найпоширеніших сфер застосування штучних нейромереж можна віднести такі.

Класифікація образів. Задача класифікації полягає у віднесенні вхідного образу, поданого вектором ознак, до одного або кількох попередньо визначених класів. Під час навчання мережі пропонуються різні ознакові описи образів із зазначенням того, до якого класу вони відносяться. Під час подання на вхід мережі певного образу на одному з її виходів має з'явитися ознака того, що образ належить цьому класу. У той самий час на інших виходах має бути ознака того, що образ до цього класу не належить.

Образи можуть можуть бути подані різними за своєю природою об'єктами: символи тексту, зображення об'єктів, звуки мови чи двигунів тощо. Найбільш відомим практичними застосуваннями нейромереж в цьому напрямку є розпізнавання літер, розпізнавання мови, класифікація сигналу електрокардіограми, класифікація акустичних сигналів підводних човнів, класифікація клітин крові та інші.

Прийняття рішень. Ця задача близька до завдання класифікації. Класифікації підлягають ситуації, характеристики яких надходять на вхід нейронної мережі. На виході мережі має з'явитися ознака рішення, яке вона прийняла.

Управління. Нейронна мережа в такому застосуванні реалізує функцію оптимального управління деякою динамічною системою. На її входи подається вектор вхідних керувальних сигналів $u(t)$, під впливом яких вона має виробити часову послідовність вихідних сигналів $y(t)$. Навчена нейронна мережа в цій задачі має визначити таку часову послідовність вхідних впливів $u(t)$, за якої динамічна система діє за бажаною траєкторією, заданою еталонною моделлю. Як практичний приклад можна навести задачу оптимального керування двигуном.

Кластеризація. Під кластеризацією розуміється розбиття множини вхідних даних на класи подібних між собою образів. Ні кількість, ні ознаки класів заздалегідь не відомі. Під час вирішення задачі кластеризації використовують неконтрольоване навчання нейромережі. Після навчання така мережа здатна визначати, до якого класу належить вхідний об'єкт, що не належав до навчальної вибірки.

Прогнозування. Здатність нейронної мережі до прогнозування безпосередньо випливає з її здатності до узагальнення та виділення прихованих залежностей між вхідними та вихідними даними. Після навчання мережа здатна передбачити майбутнє значення якоїсь послідовності на основі декількох попередніх значень або якихось існуючих на поточний момент чинників.

Стиснення даних і асоціативна пам'ять. Здатність нейромереж до виявлення взаємозв'язків між різними параметрами дає можливість подати дані великої розмірності більш компактно, якщо дані тісно взаємопов'язані між собою. Обернений процес – відновлення вихідного набору даних за частиною відомої інформації називається асоціативною пам'яттю.

Апроксимація функцій. Під час розв'язання цієї задачі використовують контрольоване навчання нейронної мережі на навчальній вибірці прецедентів у вигляді множини пар вхід-вихід $\{(x_1, y_1), (x_2, y_2) ..., (x_n, y_n)\}$, яка генерується невідомою функцією *f*, спотвореною шумом. Завдання апроксимації полягає у побудові моделі нейронної мережі, здатної реалізувати регресійну функцію *f*. Апроксимацію функцій застосовують для вирішення багатьох інженерних та наукових задач моделювання.

14.1.5 Напрями реалізації штучних нейромереж

Нині існує три основних підходи до реалізації ШНМ, а саме:

- програмна реалізація на цифрових ЕОМ традиційної архітектури;
- програмно-апаратна реалізація у вигляді співпроцесорів до ЕОМ загального призначення;
- апаратна реалізація шляхом створення нейрокомп'ютерів на базі нейроплат у вигляді паралельних нейроподібних структур.

Ранні варіанти реалізації нейронних мереж відносяться до перших двох із вказаних напрямів.

Перший напрям характеризується універсальністю, дешевизною і досить високою швидкістю навчання та функціонування нейронних мереж.

Для другого напряму характерна висока швидкість моделювання функціонування мереж, але у цьому випадку існують серйозні фізичні обмеження кількості модельованих елементів і зв'язків між ними, а також можливостей навчання і донавчання.

Третій напрям поклав початок індустрії нейрокомп'ютерів, що подають сукупність апаратних і програмних засобів для реалізації моделей нейронних мереж. На сьогоднішній день відомо вже більше 200 різних парадигм нейронних мереж (не лише детермінованих, але і імовірнісних), десятки НПС реалізовані в спеціалізованих кристалах і платах, на їх основі створено потужні робочі станції та навіть суперкомп'ютери. Сучасні технології досягли того стану, коли стало можливим виготовлення технічної системи з 3,4 млрд. нейронів (саме така кількість їх в мозку людини). Проте їх з'єднання продовжує залишатися проблемою.

14.2 Класифікація нейронних мереж

Біологічні нейронні мережі з мікроскопічних компонентів існують у тривимірному просторі і здатні до різноманітних з'єднань. Але для реалізації штучних мереж присутні фізичні обмеження. Штучні

нейромережі конструюються з базового блоку – штучного нейрону. Іншою властивістю нейромереж є величезна кількість зв'язків, які пов'язують окремі нейрони. Кількість нейронів та типи зв'язків між ними є важливим фактором забезпечення ефективної роботи мережі.

Об'єднуючись у мережі, штучні нейрони утворюють систему обробки інформації, яка забезпечує ефективну адаптацію моделі до постійних змін з боку зовнішнього середовища. В процесі функціонування мережі відбувається перетворення вхідного вектора сигналів у вихідний. Конкретний вид перетворення визначається архітектурою нейромережі, характеристиками нейронних елементів, засобами керування та синхронізації інформаційних потоків між нейронами.

З наведених вище принципів побудови штучних нейронних мереж випливають основні ознаки, за якими здійснюють класифікацію мереж. До них відносяться:

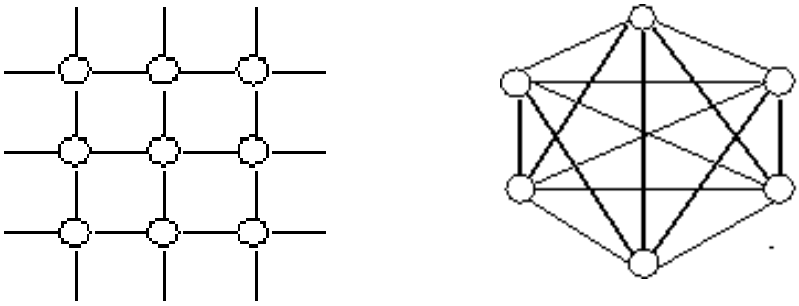
1) **структура нейромережі** – спосіб зв'язків нейронів у нейромережі,

2) **архітектура нейромережі** – структура нейромережі та типи нейронів,

3) **парадигма нейромережі** – спосіб навчання та використання нейромережі.

Різні парадигми може бути реалізовано на базі однієї архітектури нейромережі і навпаки, одній парадигмі можуть відповідати декілька архітектур.

Серед відомих архітектурних рішень виділяють групу *слабозв'язаних нейронних мереж*, в яких кожен нейрон мережі зв'язаний тільки із сусідніми нейронами. В *повнозв'язаних нейромережах* входи кожного нейрона зв'язані з виходами всіх інших нейронів. Названі архітектури подано на рис. 14.8, а) і рис. 14.8, б), відповідно.



а) б)

Рисунок 14.8 – Архітектура слабозв’язаної (а) і повнозв’язаної нейронної мережі (б)

Найпоширенішим варіантом архітектури є багатошарові мережі (рис. 14.9). В цих мережах нейрони об'єднуються у прошарки з єдиним вектором вхідних сигналів. На вхідний рецепторний прошарок нейронної мережі подається зовнішній вхідний вектор. Вихідні сигнали останнього ефекторного прошарку є виходами нейронної мережі. Окрім вказаних

вхідного та вихідного прошарків, нейромережа має також один або декілька прихованих прошарків нейронів, які з зовнішнім середовищем не контактують.

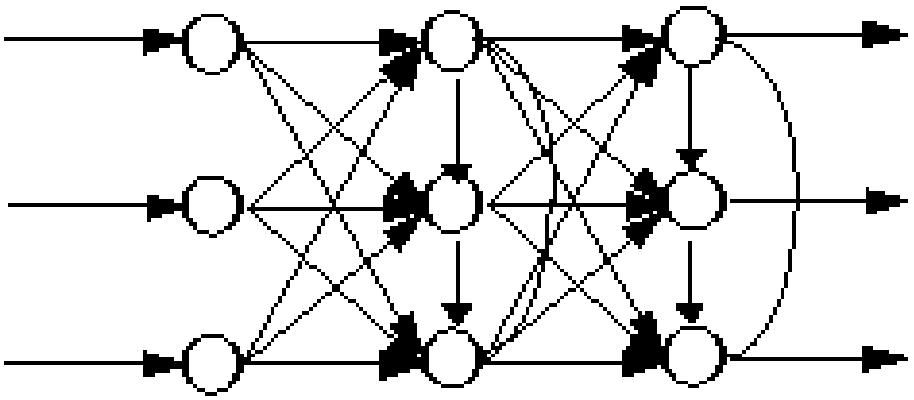


Рисунок 14.9 – З'єднання нейронів в багатошаровій мережі

У цьому випадку зв'язки між нейронами одного прошарку називають *латеральними* (бічними), а зв'язки між нейронами різних прошарків називають *проективними*.

Окрім показаної на рис. 14.9 типової структури штучних нейромереж існують мережі, які містять лише один прошарок або навіть один елемент. Проте більшість застосувань потребують наявності в мережі, як мінімум трьох типів прошарків – вхідного, прихованого та вихідного. Прошарок вхідних нейронів отримує дані або з вхідних файлів, або безпосередньо з електронних давачів. Вихідний прошарок пересилає інформацію безпосередньо до зовнішнього середовища, до вторинного комп'ютерного процесу або до інших пристроїв. Між цими двома прошарками може бути багато прихованих прошарків, які містять багато нейронів в різноманітних зв'язаних структурах. Входи та виходи кожного з прихованих нейронів сполучені з іншими нейронами.

За ознакою напрямку зв'язку від одного нейрона до іншого нейромережі поділяють на *аферентні*, в яких зв'язки скеровані від вхідних прошарків до вихідних, і *еферентні*, в яких зв’язки скеровані в зворотному напрямку.

В більшості мереж кожен нейрон прихованого прошарку отримує сигнали від всіх нейронів попереднього прошарку чи від нейронів вхідного прошарку. Після виконання операцій над сигналами нейрон передає свій вихід до всіх нейронів наступних прошарків, забезпечуючи передачу вперед (*feedforward*) на вихід.

За зворотного зв'язку вихід нейронів прошарку скеровується до нейронів попереднього прошарку (рис. 14.10).

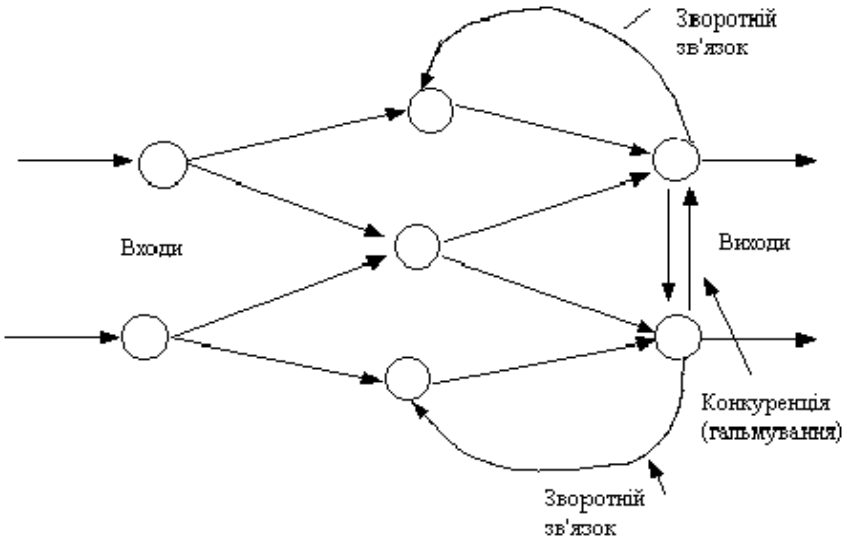


Рисунок 14.10 – Нейромережі зі зворотним зв’язком у шарах

Напрямок зв’язків нейронів має значний вплив на роботу мережі. Більшість програмних нейромереж дозволяють користувачу додавати, вилучати та керувати з'єднаннями як завгодно. Корегуючи параметри, можна налаштувати зв'язки як на посилення, так і на послаблення величини сигналів.

За ознакою архітектури зв'язків більшість відомих нейромереж можна поділити на два великих класи:

1. **Мережі прямого поширення** (з односкерованими послідовними зв'язками);
2. **Мережі зворотного поширення** (з рекурентними зв'язками).

Приклади нейронних мереж, що відносяться до названих класів, наведено в таблиці 14.1

Мережі прямого поширення відносять до статичних, тут на входи нейронів надходять вхідні сигнали, які не залежать від попереднього стану мережі.

Рекурентні мережі вважаються динамічними, оскільки за рахунок зворотних зв'язків (петель) входи нейронів модифікуються в часі, що призводить до зміни станів мережі.

Таблиця 14.1 – Приклади нейромереж прямого і зворотного поширення (послідовних і рекурентних)

Мережі прямого поширення	**Рекурентні мережі**
Перцептрони Мережа Back Propagation Мережа зустрічного поширення Карта Кохонена Згортальні (конволютивні CNN)	Мережа Хопфілда Мережа Хемінга Мережа адаптивної резонансної теорії Двоскерована асоціативна пам'ять

Якщо брати до уваги третю класифікаційну ознаку (парадигму), то як було вище відзначено, нині відомо вже більше 200 різних парадигм нейронних мереж (не лише детермінованих, але і ймовірнісних). Їх опис детально подано в спеціальній літературі з тематики нейромереж (за приклад можна назвати опис різних типів ШНМ на сайті https://www.asimovinstitute.org/neural-network-zoo/), тому в цьому розділі розгляду підлягають тільки окремі з них.

14.3 Навчання штучних нейронних мереж

Головною перевагою нейронних мереж є їх розподілена структура та здатність до навчання. До головних недоліків в цьому плані можна віднести неможливість інтегрування знань в нейронну мережу та їх добування з неї. Вибір під час навчання кількості нейронів, їх зв’язків та параметри навчального процесу можна зробити тільки експериментально.

В загальному вигляді процедуру навчання можна описати таким чином: існує збірник задач – набір прикладів з заданими відповідями. Ці приклади надаються системі. Нейрони отримують по вхідних зв'язках сигнали – «умови прикладу», перетворюють їх, кілька разів обмінюються перетвореними сигналами і, нарешті, видають відповідь – також набір сигналів. Відхилення від правильної відповіді штрафується. Навчання складається в мінімізації штрафу як (неявної) функції зв'язків.

Процес навчання нейромереж розглядається як налаштування архітектури та вагових коефіцієнтів синаптичних зв'язків відповідно до даних навчальної множини для ефективного вирішення поставленої задачі.

Для навчання нейромереж можливе застосування таких методів:

- контрольоване навчання (навчання з вчителем),
- неконтрольоване навчання (навчання без вчителя).

14.3.1 Контрольоване навчання нейромережі

Більшість реалізацій нейромереж використовують контрольоване навчання, де поточний вихід постійно порівнюється з бажаним виходом. Вагові коефіцієнти зв’язків на початку встановлюються випадково (*ініціалізація мережі*), але під час наступних ітерацій коректуються, щоб досягти близької відповідності між бажаним та біжучим виходами. Такі методи навчання націлені на мінімізації біжучих похибок всіх елементів обробки, що відбувається завдяки неперервній зміні синаптичних ваг до досягнення прийнятної точності мережі.

Перед використанням нейромережа з контрольованим навчанням має бути навченою. Фаза навчання займає певний час. Навчання вважається закінченим у разі досягнення нейромережею визначеного користувачем рівня ефективності і бажаної статистичної точності. Після навчання вагові коефіцієнти зв’язків фіксуються для подальшого застосування. Деякі типи

мереж дозволяють під час використання продовжувати навчання, і це допомагає мережі адаптуватись до змінних умов.

Навчальні множини мають бути достатньо великими, щоб містити всю необхідну інформацію для виявлення важливих особливостей і зв'язків. Навчальні приклади мають містити широке різноманіття даних. Якщо мережа навчається лише для одного прикладу, вагові коефіцієнти, що старанно встановлено для цього прикладу, радикально змінюються у навчанні для наступного прикладу. Попередні приклади в процесі навчання наступних просто забуваються. Як результат – система має навчатись всьому разом, знаходячи найкращі вагові коефіцієнти для загальної множини прикладів.

Наприклад, у навчанні системи розпізнавання пікселних образів для десяти цифр, які подано двадцятьма прикладами кожної цифри, всі приклади цифри «сім» недоцільно подавати послідовно. Краще надати мережі спочатку один тип подання всіх цифр, потім другий тип і так далі.

Головною компонентою для успішної роботи мережі є подання і кодування вхідних та вихідних даних. Штучні мережі працюють лише з числовими вхідними даними, отже, необроблені дані, що надходять із зовнішнього середовища, мають перетворюватись. Важливою є нормалізація даних, тобто приведення всіх значень даних до єдиного діапазону. Нормалізація виконується шляхом ділення кожної компоненти вхідного вектора на довжину вектора, що перетворює вхідний вектор в одиничний. Попередня обробка зовнішніх даних, отриманих за допомогою сенсорів, у машинний формат є спільною і легкодоступною для стандартних комп'ютерів.

Якщо після контрольованого навчання нейромережа ефективно опрацьовує дані навчальної множини, важливим стає її ефективність при роботі з даними, які не використовувались для навчання. У випадку отримання незадовільних результатів для тестової множини, навчання продовжується. Тестування використовується для забезпечення запам'ятовування не лише даних заданої навчальної множини, але і створення загальних образів, що можуть міститись в даних.

Навчання багатошарової нейронної мережі методом оберненого поширення помилки (backpropagation). Даний метод був запропонований у 1986 р. Румельхартом, Макклеландом і Вильямсом.

Навчання мережі починається з подання образу й обчислення відповідної реакції. Порівняння з бажаною реакцією дає можливість змінювати ваги зв'язків таким чином, щоб мережа на такому кроку могла видавати більш точний результат. Навчальне правило забезпечує налаштування ваг зв'язків. Інформація про виходи мережі є вихідною для нейронів попередніх прошарків. Ці нейрони можуть налаштовувати ваги своїх зв'язків для зменшення похибки на кожному ітераційному кроці навчання.

Під час подання на неналаштовану мережу вхідного образу вона буде видавати деякий випадковий вихід. Функція помилки являє собою різницю між поточним виходом мережі й значенням ідеального виходу, що необхідно одержати. Для успішного навчання мережі потрібно наблизити вихід мережі до бажаного виходу, тобто послідовно зменшувати розмір функції помилки. Це досягається настроюванням міжнейронних зв'язків. Узагальнене дельта-правило навчає мережу шляхом обчислення функції помилки для заданого входу з наступним її оберненим поширенням від кожного прошарку до попереднього. Кожний нейрон у мережі має свої ваги, що настроюються, щоб зменшити розмір функції помилки. Для нейронів вихідного прошарку відомі їх фактичні і бажані значення виходів. Тому настроювання ваг зв'язків для таких нейронів є відносно простим. Проте для нейронів попередніх прошарків настроювання не настільки очевидне. Інтуїтивно ясно, що нейрони внутрішніх прошарків, які пов'язані з виходами, що мають велику похибку, мають змінювати свої ваги значно сильніше, чим нейрони, сполучені з майже коректними виходами. Іншими словами, ваги цього нейрона мають змінюватися прямо пропорційно помилці тих нейронів, із якими цей нейрон пов'язаний. От чому обернене поширення цих помилок через мережу дозволяє коректно настроювати ваги зв'язків між усіма прошарками. У цьому випадку розмір функції помилки зменшується і мережа навчається.

В математичному плані в загальному випадку задача навчання ШНМ зводиться до знаходження певної функціональної залежності $\boldsymbol{Y}=f(X)$, де $\boldsymbol{X}$ – вхідний, а $\boldsymbol{Y}$ – вихідний вектори. Загалом таке завдання за обмеженого набору вхідних даних має безліч розв’язків. Для обмеження простору пошуку під час навчання ставиться задача мінімізації цільової функції помилки нейронної мережі, яку знаходять за методом найменших квадратів:

$$E(W) = \frac{1}{2}\sum_{j=1}^{p}(y_j - d_j)^2, \qquad (14.6)$$

де y_j – значення j-го виходу нейромережі,

d_j-цільове значення j-го виходу,

p – число нейронів у вихідному шарі,

W – вектор ваг зв’язків між нейронами.

Множник 1/2 введено для спрощення операції диференціювання квадратичної функції помилки.

Навчання нейромережі використовується за методом градієнтного спуску, тобто на кожній ітерації зміна ваги здійснюється за формулою:

$$\Delta w_{ij} = -\eta \cdot \frac{\partial E}{\partial w_{ij}}, \qquad (14.7)$$

де η – параметр, що задає швидкість навчання,

w_{ij} – вага зв'язку між i -м і j -м нейронами,

а частинна похідна за цим зв’язком визначається як:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j} \cdot \frac{\partial S_j}{\partial w_{ij}} \ . \qquad (14.8)$$

У виразі (14.8) y_j – значення виходу j-го нейрона; S_j – зважена сума вхідних сигналів, яка визначається за формулою:

$$S = \sum_{i=1}^{n} x_i w_i \ , \qquad (14.9)$$

звідки випливає, що множник

$$\frac{\partial S_j}{\partial w_{ij}} \equiv x_i \qquad (14.10)$$

є пропорціональним значенню сигналу x_i на i-му вході нейрона.

Перший співмножник у виразі (14.8) визначиться, як:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot \frac{\partial S_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{dS_k} \cdot w_{jk}^{(n+1)} \ , \qquad (14.11)$$

де k – кількість нейронів в n+1 прошарку.

Введемо допоміжну змінну, пропорціональну добутку частинної похідної від помилки нейромережі за сигналом її j–го вихідного нейрона на частинну похідну сигналу виходу j-го нейрона за його вхідним сигналом:

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{dS_j} \ . \qquad (14.12)$$

Тепер ми можемо визначити рекурсивну формулу для визначення $\delta_j^{(n)}$ n-ного шару, якщо нам відомо $\delta_k^{(n+1)}$ наступного n+1-го шару.

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)} \cdot w_{jk}^{(n+1)} \right] \cdot \frac{dy_j}{dS_j} \qquad (14.13)$$

Знаходження $\delta_j^{(n)}$ останнього шару нейромережі не завдасть труднощів, оскільки є відомий цільової вектор, тобто вектор тих значень, які має видавати НМ для заданого набору вхідних значень.

$$\delta_j^{(N)} = \left(y_i^{(N)} - d_i\right) \cdot \frac{dy_i}{dS_i}. \qquad (14.14)$$

Тепер формулу (14.7) можна записати в розкритому вигляді:

$$\Delta w_{ij}^{(n)} = -\eta \cdot \delta_j^{(n)} \cdot x_i^n \ . \qquad (14.15)$$

Сформульований в математичному плані алгоритм *back propagation* дозволяє надати вербальний опис алгоритму навчання ШНМ таким методом:

1 – подати на вхід НМ один із необхідних образів та визначити значення виходів нейронів нейромережі;

2 – розрахувати для вихідного шару нейромережі $\delta^{(N)}$ за формулою (14.14) та розрахувати зміни ваги $\Delta w_{ij}^{(N)}$ вихідного шару N за формулою (14.15);

3 – Розрахувати за формулами (14.13) та (14.15) відповідно $\delta^{(n)}$ і $\Delta w_{ij}^{(n)}$ для інших шарів НМ, *n*=N-1...1

4 – Скоригувати всі ваги НМ (14.7)

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t). \qquad (14.16)$$

5 – Якщо помилка є суттєвою, то перейти на крок 1.

На кроці 2 мережі подаються почергово у випадковому порядку вектори з навчальної вибірки.

Розглянутий вище найпростіший метод градієнтного спуску є досить неефективним у випадку, коли похідні за різними вагами сильно відрізняються.

Це відбувається в ситуації, коли значення функції *S* для деяких нейронів близьке за модулем до 1 або коли модуль деяких ваг набагато більший 1. В такому разі для плавного зменшення помилки потрібно вибирати дуже маленьку швидкість навчання, але у цьому випадку навчання може зайняти багато часу.

Найпростішим методом удосконалення градієнтного спуску є введення моменту μ, який дозволяє вплив градієнта на зміну ваги змінювати з часом. Тоді формула (14.16) набуває вигляду

$$\Delta w_{ij}^{(n)}(t) = -\eta \cdot \delta_j^{(n)} \cdot x_i^n + \mu \Delta w_{ij}^{(n)}\left(t-1\right). \qquad (14.17)$$

Додатковою перевагою від запровадження цього моменту є здатність алгоритму долати дрібні локальні мінімуми.

14.3.2 Неконтрольоване навчання

Навчання без учителя (неконтрольоване навчання) може стати великим здобутком у майбутньому, оскільки воно відзначається можливістю комп'ютерів автоматично навчатися в справжньому розумінні цього терміна. Нині неконтрольоване навчання використовується в таких типах нейромереж, як самоорганізовані карти. У цих мережах ваги не коригуються зовнішніми впливами, а сама мережа внутрішньо контролює свою ефективність, шукаючи закономірності або тенденції у вхідних сигналах та адаптуючись відповідно до навчальної функції. Навіть без надання їй повідомлення про правильність або неправильність дій, потрібно, щоб мережа мала інформацію щодо власної організації, яка вбудована у топологію мережі, та правила навчання.

Алгоритм неконтрольованого навчання спрямований на виявлення близькості між групами нейронів, що працюють разом. Якщо зовнішній сигнал активізує будь-який вузол в групі нейронів, активність всієї групи збільшується. Також, коли зовнішній сигнал у групі зменшується, це призводить до зниження активності всієї групи.

Основу для навчання становить конкуренція між нейронами. Навчання конкурентних нейронів підсилює реакцію певних груп на певні сигнали. Це забезпечує зв'язок між групами та їх реакцією. Під час конкуренції змінюються ваги лише у нейрона-переможця.

14.4 Опис принципів роботи окремих нейромереж

Оскільки всі штучні нейронні мережі базуються на концепції нейронів, з'єднань та передатних функцій, існує подібність між різними структурами або архітектурами нейронних мереж. Більшість змін випливає з різних парадигм навчання. Розглянемо деякі з найвідоміших штучних нейромереж.

14.4.1 Перцептрон Розенблатта

Перцептрон Розенблатта вважають першою моделлю нейромереж. Перцептронна теорія є базовою для багатьох типів штучних НМ прямого поширення, які вважаються класикою для вивчення.

Перцептрон є одношаровою НМ, яка здатна розпізнавати найпростіші образи. В ній один нейрон обчислює зважену суму сигналів вхідних елементів, віднімає значення зсуву і формує результат на виході з використанням жорсткої порогової функції, вихід якої дорівнює +1 чи -1. Залежно від значення сигналу на виході рішення приймається таким чином:

якщо +1 – вхідний сигнал належить до класу А,

якщо -1 – вхідний сигнал належить до класу В.

Схему такого одношарового перцептрона, графік його передатної функції і схему областей прийняття рішенньь у двовимірному просторі

вхідних сигналів показано на рис. 14.11. Перцептрон, що складається з одного нейрона, формує дві вирішувальні області, які для багатовимірного простору розділяються гіперплощиною. Вирішувальні області визначають, які вхідні образи будуть віднесені до одного з двох класів.

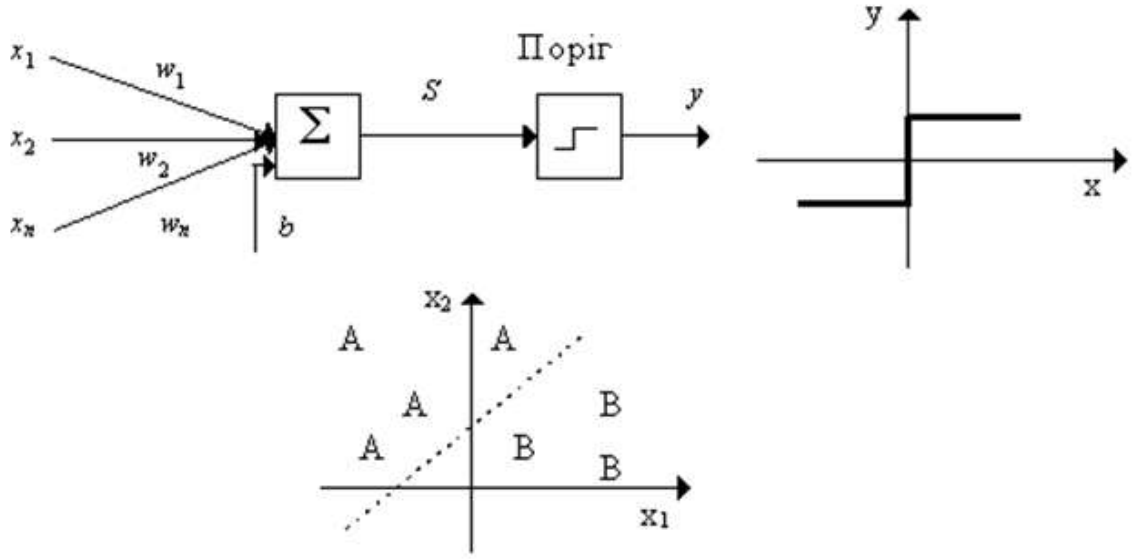


Рисунок 14.11 – Схема нейрона з пороговою передатною функцією і його розподільна поверхня

На рис. 14.11 наведено випадок для даних, описаних в двовимірному ознаковому просторі. Розподільна поверхня є прямою лінією на площині. Рівняння, яке описує розподільну пряму, залежить від значень синаптичних ваг і зсуву.

Алгоритми навчання одношарового перцептрона детально описані в розділі 9.2 цієї частини посібника, тому детально розглядати їх тут не будемо.

Наведемо основні характеристики перцептронної нейромережі.

Тип вхідних сигналів – бінарні або неперервні (аналогові).

Кількості входів і виходів за програмної реалізації обмежені тільки можливостями обчислювальної системи, на якій моделюється нейронна мережа, а за апаратної реалізації – технічними можливостями.

Основні сфери застосування – розпізнавання образів, класифікація.

Недоліки – лінійні розподільні поверхні (гіперплощини) дозволяють вирішувати лише найпростіші завдання розпізнавання.

Переваги – простота програмної і апаратної реалізації моделей та нескладний і швидкий алгоритм навчання.

Модифікаці: багатошарові перцептрони дають можливість будувати складні розподільні поверхні і є більш поширеними (рис. 14.12).

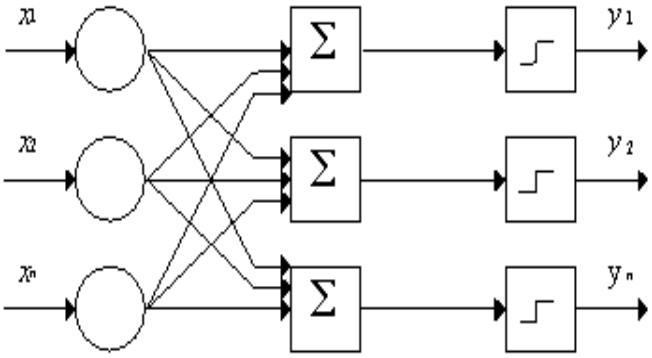


Рисунок 14.12 – Схема багатошарового перцептрона

14.4.2 Нейромережа зворотного поширення похибки (back propagation)

Нейронні мережі зворотного поширення – це потужний інструмент пошуку закономірностей, прогнозування, якісного аналізу. Таку назву вони отримали через використовуваний алгоритм навчання, у якому помилка поширюється від вихідного шару до вхідного, тобто у напрямі, протилежному напряму поширення сигналу за нормального функціонування мережі. Цей алгоритм був детально описаний вище.

Нейронна мережа зворотного поширення складається з кількох шарів нейронів, причому кожен нейрон шару *i* пов'язані з кожним нейроном шару *i*+1, тобто йдеться про повнозв'язну архітектуру НМ. Нині парадигма навчання *back propagation* є популярною та простою моделлю навчання для багатошарових складних мереж. Вона використовується в різних застосуваннях і породила великий клас нейромереж з різними структурами та методами навчання. Типова мережа *back propagation* має вхідний прошарок, вихідний прошарок та принаймні один прихований прошарок (рис. 14.13). Теоретично обмежень щодо числа прихованих прошарків не існує.

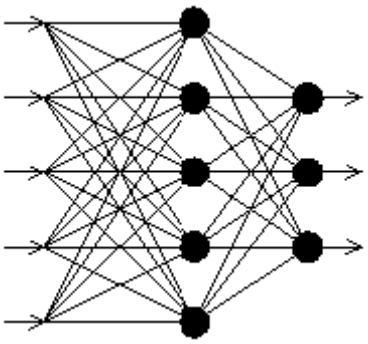


Рисунок 14.13 – Схема мережі *back propagation* з трьома шарами

Нейрони організовано в пошарову структуру з прямою передачею (вперед) сигналу. Кожний нейрон мережі продукує зважену суму своїх входів, пропускає цю величину через передатну функцію і видає вихідне значення. Мережа може моделювати функцію практично будь-якої

складності, причому число прошарків і число нейронів у кожному прошарку визначають складність функції.

Важливим в процесі моделювання мережі є визначення числа проміжних прошарків і числа нейронів в них. У цьому випадку використовують такі загальні правила:

1. Кількість входів та виходів мережі визначаються кількістю вхідних та вихідних параметрів досліджуваного об'єкта, явища, процесу тощо. На відміну від зовнішніх прошарків, число нейронів прихованого прошарку $n_{прих}$ вибирається емпіричним шляхом. В більшості випадків достатньою кількістю нейронів буде $N_{прих} <= N_{вх} + N_{вих}$, де $N_{вх}$, $N_{вих}$ – кількість нейронів у вхідному і вихідному прошарках, відповідно.
2. Якщо складність у відношенні між отриманими та бажаними даними на виході збільшується, кількість нейронів прихованого прошарку має також збільшитись.
3. Якщо процес, що моделюється, може розділятись на багато етапів, потрібні один або декілька додаткових прихованих прошарків. Якщо розділення на етапи не потрібне, то наявність додаткових прошарків може призвести до виникнення явища перезапам'ятовування і викликати неправильне загальне рішення.

Після того, як визначено число прошарків і число нейронів в кожному з них, потрібно знайти значення для синаптичних ваг і порогів мережі, які спроможні мінімізувати похибку роботи мережі. Для цього використовують алгоритми навчання, які забезпечують підгонку моделі НМ до наявної навчальної вибірки даних. Помилка для конкретної моделі мережі визначається шляхом проходження через мережу всіх навчальних прикладів і порівняння отриманих вихідних значень з бажаними. Множина помилок створює функцію похибок, значення якої можна розглядати як похибку мережі. Як функції помилки найчастіше використовують критерій суми середньоквадратичних відхилень отриманих результатів від заданих.

Краще розуміння алгоритму навчання *Back Propagation* можливе після уяснення поняття поверхні станів. Кожному значенню синаптичних ваг і порогів мережі (вільних параметрів моделі кількістю N) відповідає один вимір в багатовимірному просторі. N+1-ий вимір відповідає похибці мережі. Для різноманітних сполучень ваг відповідну похибку мережі можна зобразити точкою в N+1-вимірному просторі, всі ці точки утворюють деяку поверхню – поверхню станів. Мета навчання нейромережі полягає в знаходженні на багатовимірній поверхні найнижчої точки.

Поверхня станів, на якій розглядається процес навчання нейромережі, характеризується складною структурою та різноманітними властивостями, серед яких можна виділити локальні мінімуми (точки, які є найнижчими у своєму навколишньому середовищі, але вищі за глобальний мінімум), пласкі ділянки, сідлові точки і довгі вузькі яри. Аналітичні

методи не дозволяють однозначно визначити положення глобального мінімуму на цій поверхні, тому процес навчання нейромережі полягає у вивченні цієї поверхні.

Алгоритм навчання, виходячи зі стартової конфігурації ваг і порогів (з випадково обраної точки на поверхні), поступово шукає глобальний мінімум. Для цього обчислюється вектор градієнта поверхні помилок, який вказує напрямок найшвидшого спуску з заданої точки по поверхні. Пройшовши трохи в цьому напрямку, помилка зменшується. Алгоритм зупиняється, коли досягає нижньої точки, яка може бути як локальним, так і глобальним мінімумом.

Однією із складностей є вибір довжини кроків. Велика довжина кроку прискорює збіжність, але може призвести до перестрибування рішення або руху в неправильному напрямку. Малий крок дозволяє правильно визначити напрямок, але збільшує кількість ітерацій. Зазвичай розмір кроку вибирають пропорційно крутизні схилу з певною константою – швидкістю навчання. Вибір цієї константи залежить від конкретної задачі і визначається експериментально. Ця константа також може змінюватися з часом, зменшуючись із мірою прогресу алгоритму.

Алгоритм працює ітеративно, розбиваючи процес на епохи. На кожній ітерації всі навчальні приклади подаються на вхід мережі почергово, після чого вихідні значення порівнюються з бажаними результатами і обчислюється похибка. Це значення похибки, а також градієнт поверхні стану використовуються для корекції ваг, і цей процес повторюється. Навчання завершується, якщо пройдено певну кількість епох, якщо досягнуто певного рівня похибки або якщо похибка не зменшується (вибір критерію зупинки залишається за користувачем).

Вхідні сигнали – цілого або дійсного типу.

Вихідні сигнали – дійсного типу з інтервалу, який визначається передатною функцією нейронів.

Тип передатної функції – сигмоїдна. Функції цього типу є монотонно зростаючими і мають похідні, що відрізняються від нуля на всій області визначення. Такі їхні характеристики забезпечують правильне навчання і функціонування мережі.

Сфери застосування – класифікація, прогнозування і розпізнавання образів.

Недоліки – мала швидкість навчання.

Переваги – зрозумілий та ефективний алгоритм для вирішення багатьох практичних задач.

Модифікації – алгоритм back propagation можна модифікувати за рахунок використання різних критеріїв помилки, різних процедур визначення напрямку і величини кроку.

14.4.3 Мережа Кохонена

Мережа цього типу (рис. 14.14) розроблена на початку 1980-х рр. Тойво Кохоненом і відрізняється принципово від розглянутих вище мереж, оскільки вона використовує неконтрольоване навчання з використанням навчальної множини, що складається тільки із значень вхідних змінних.

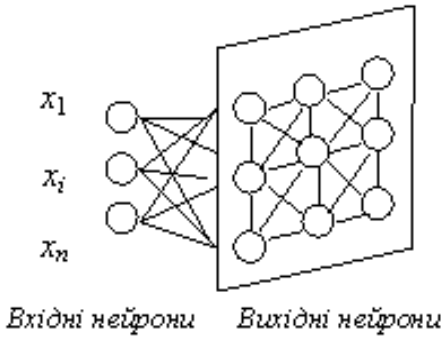


Рисунок 14.14 – Схема нейромережі Кохонена

Мережа визначає кластери в навчальних даних та розподіляє дані до знайдених відповідних груп. Якщо після цього мережа зустрічається з набором даних, який не подібний до будь-якого з відомих зразків, вона призначає його до нового кластера. За наявності міток класів у даних мережа може виконувати завдання класифікації. Мережі Кохонена корисно застосовувати і у випадках, коли класи відомі – їх перевага полягає у здатності виявляти подібності між різними класами.

Мережа Кохонена складається лише з двох шарів: вхідного та вихідного. Елементи карти розташовуються у певному просторі, зазвичай, двовимірному. Процес навчання мережі Кохонена базується на послідовних наближеннях. Під час навчання на входи подаються дані, проте мережа вирішує не реагувати на еталонне значення виходу, а адаптується до закономірностей у вхідних даних. Навчання розпочинається з випадкового вибору розташування центрів.

Під час подавання на вхід мережі в послідовному порядку навчальних прикладів визначається найбільш схожий нейрон (той, у якого скалярний добуток ваг і поданого на вхід вектора є мінімальним). Цей нейрон призначається переможцем і є центром під час налаштування ваг в сусідніх нейронах. Описане правило навчання передбачає «змагальне» навчання з врахуванням відстані нейронів від «нейрона-переможця».

В цій процедурі метою навчання є не мінімізація помилки, а налаштування ваг (внутрішніх параметрів нейронної мережі) для максимального збігу з вхідними даними.

Алгоритм Кохонена працює за ітеративною схемою, в якій кожна ітерація відповідає окремій епосі. На кожній епосі мережа обробляє один приклад з навчального набору. Вхідні сигнали послідовно подаються мережі без визначення бажаних вихідних сигналів. Після того, як достатня кількість вхідних векторів була подана, синаптичні ваги мережі можуть

визначити кластери. Ваги організовані так, що вузли, які топологічно близькі, реагують на схожі вхідні сигнали.

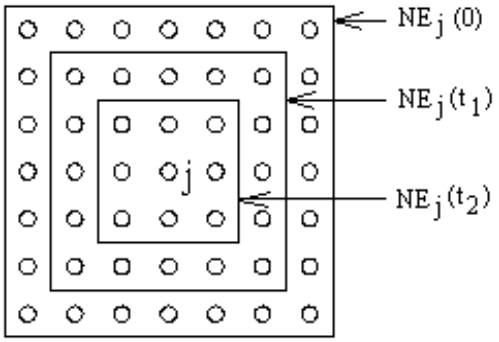


Рисунок 14.15 – Початкова зона сусідства в карті Кохонена

Внаслідок роботи алгоритму центр кластера фіксується у певній позиції, що задовольняє кластеризовані приклади, для яких цей нейрон є «переможцем». Під час навчання мережі необхідно визначити міру сусідства нейронів, або окіл нейрона-переможця, який містить кілька нейронів, що оточують нейрон-переможець.

Спочатку до цього околу належить велика кількість нейронів, а потім його розмір поступово зменшується. Таким чином, мережа формує топологічну структуру, в якій схожі приклади утворюють групи, розташовані близько одна до одної на топологічній карті.

Алгоритм роботи мережі Кохонена складається з таких кроків:

1 – Початкова ініціалізація мережі, під час якої ваговим коефіцієнтам мережі надаються невеликі випадкові значення. Початкову зону сусідства показано на рис. 14.15.

2 – Подання мережі нового вхідного сигналу.

3 – Обчислення відстані до всіх нейронів мережі.

Відстані d_j від вхідного сигналу до кожного нейрона j визначаються за формулою:

$$d_j = \sum_{i=1}^{N} (x_i(t) \cdot w_{ij}(t))^2 \ , \qquad (14.18)$$

де x_i – *і-ий* елемент вхідного сигналу в момент часу t,

$w_{ij}(t)$ – вага зв'язку від *і-го* елемента вхідного сигналу до нейрона j у момент часу t.

4 – Вибір нейрона з найменшою відстанню:

Вибирається нейрон-переможець j^*, для якого відстань d_j найменша.

5 – Налаштування ваг нейрона j^* і його сусідів:

Робиться налаштування ваг для нейрона j^* і всіх нейронів з його околу NE. Нові значення ваг:

$$w_{ij}(t+1)=w_{ij}(t)+r(t)(x_i(t)-w_{ij}(t)), \qquad (14.19)$$

де $r(t)$ – швидкість навчання, що зменшується з часом (додатне число, менше одиниці).

6 – Повернення до кроку 2.

У цьому алгоритмі використовується коефіцієнт швидкості навчання, який поступово зменшується, щоб точніше виконати корекцію в кожній новій епосі. Це призводить до фіксації позиції центра у певному положенні, що ефективним способом кластеризує приклади, для яких відповідний нейрон визнається переможцем.

Властивість топологічного порядку досягається за допомогою поняття околу в цьому алгоритмі. Окіл являє собою групу нейронів, в оточенні нейрона-перможця. Згідно з коефіцієнтом швидкості навчання, розмір околу поступово зменшується, з початковою великою кількістю нейронів (можливо, всю карту), і на останніх етапах він звужується до нуля, складаючись лише з нейрона-переможця.

У цьому алгоритмі корекція застосовується не лише до нейрона-переможця, а й до всіх нейронів його поточного околу. Внаслідок цієї зміни розміру околу початково великі ділянки мережі поступово мігрують в напрямку навчальних прикладів. Таким чином, мережа формує грубу структуру топологічного порядку, за якої подібні приклади активують групи нейронів, розташованих близько один до одного на топологічній карті.

З кожною наступною епохою швидкість навчання та розмір околу зменшуються, це призводить до виявлення більш тонких відмінностей всередині ділянок карти, що зрештою дозволяє точніше налаштувати кожен нейрон.

Навчання часто розбивають навмисно на дві фази: першу – коротку, з високою швидкістю навчання і великими околами, та другу – більш тривалу, з низькою швидкістю навчання і околами, які дуже малі або майже не відрізняються від нуля.

Після завершення процедури навчання мережі розпізнавати структури даних її можна використовувати для візуалізації під час аналізування даних. Це може бути корисно в таких сферах як кластерний аналіз, розпізнавання образів та класифікація.

Недоліки – обмеження використання мережі для кластерного аналізу тільки в тому випадку, коли заздалегідь передбачено кількість кластерів.

До переваг можна віднести те, що мережа Кохонена може працювати в умовах завад, оскільки кількість кластерів фіксована, ваги мережі модифікуються повільно і налаштування ваг завершується після завершення навчання.

14.4.4 Мережа Хопфілда

Асоціативна ШНМ Хопфілда є прикладом моделі, що застосовує адресовану за змістом пам’ять. Вона була запропонована американським вченим Джоном Хопфілдом у 1982 році для вирішення задач оптимізації.

Модель Хопфілда складається з трьох шарів: вхідного, внутрішнього (прошарка Хопфілда) та вихідного. Кожен шар містить однакову кількість нейронів. Сигнали з нейронів вхідного рівня передаються на входи відповідних нейронів внутрішнього шару Хопфілда, зв’язки з’єднань між ними мають фіксовані ваги. Виходи внутрішнього шару з’єднуються з входами всіх нейронів цього самого рівня, крім самих себе, і з відповідними нейронами вихідного рівня. Процес навчання містить спрямування даних з вхідного шару до внутрішнього, значення виходів якого коливаються до завершення певної кількості циклів, передаючи поточний стан на вихідний шар. Цей стан відповідає образу, який мережа запам'ятовує.

Навчання потребує подання навчального образу на вхідному і вихідному шарах одночасно. Рекурсивна природа внутрішнього рівня дозволяє коригувати ваги всіх зв'язків. Відповідні пари «вхід-вихід» для правильного навчання мають відрізнятися між собою.

Схему нейромережі Хопфілда показано на рис. 14.16.

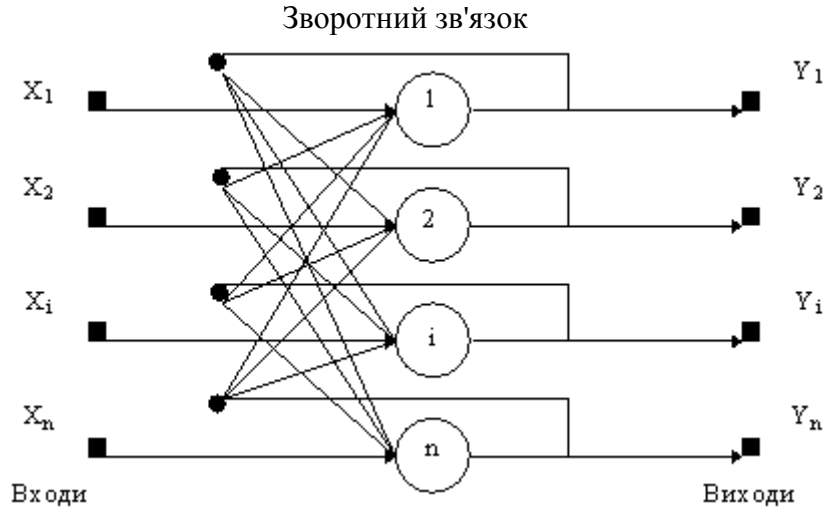


Рисунок 14.16 – Структура нейронної мережі Хопфілда

У випадку використання мережі як асоціативної пам'яті, вона має містити два основних обмеження.

1 – Кількість образів, які можна зберегти та точно відтворити, обмежена. У випадку зберігання надто великої кількості образів, мережа може перейти до нового, неіснуючого образу, який відрізняється від усіх запрограмованих, або зовсім не збігтися. Межа пам'яті для мережі становить приблизно 15 % від кількості нейронів у внутрішньому шарі Хопфілда.

2 – Якщо навчальні приклади схожі між собою, прошарок Хопфілда може стати нестабільним. Зразок вважається нестабільним, якщо він активується протягом нульового часу і мережа збігається до іншого образу

з навчальної вибірки. Цю проблему можна вирішити, вибираючи навчальні приклади, які більш ортогональні між собою.

В задачі асоціативної пам'яті є певний набір двійкових сигналів, які є зразковими поданнями об'єктів або процесів. Мережа має бути здатна розпізнавати вхідний сигнал, навіть якщо він змішаний («зачумлений»), та відновлювати або робити висновок про те, що вхідні дані не схожі ні на один відповідний зразок.

Загальним поданням будь-якого сигналу є вектор $x_1, x_2, ..., x_n$, де n – кількість нейронів у мережі. Кожен елемент x_i набуває значення або +1, або -1. Зразок k описується вектором X_k, його компонентами є x_{ik}, де $k = 0, ..., m$-1, а m – кількість зразків. Якщо мережа успішно розпізнає зразок на основі введених даних, вихідні значення мережі будуть збігатися з відповідним зразком, тобто $Y=X_k$, де Y – вектор вихідних значень мережі: $Y=(y_1, y_2, ..., y_n)$. У випадку невдалого розпізнавання вихідний вектор не збігається ні з одним із зразків.

Наприклад, якщо сигнали подано у формі зображень, графічне відображення даних з виходу мережі може показати або точне відтворення одного зі зразків (у випадку успіху), або ж «імпровізацію» мережі (у випадку невдачі).

Алгоритм роботи мережі Хопфілда складається з таких етапів:

1. На стадії ініціалізації мережі вагові коефіцієнти синапсів встановлюються таким чином:

$$w_{ijj} = \begin{cases} 0, & якщо \ \ i = j, \\ \sum_{i=1}^{m} x_i^k \cdot x_j^k, & якщо \ i \neq j. \end{cases} \qquad (14.20)$$

У виразі (14.20) i і j – індексидосинаптичного і післясинаптичного нейронів, відповідно;

x_{ik}, x_{jk} – відповідно i-ий і j-ий елементи вектора k-ого зразка.

2. На входи мережі подається невідомий сигнал $y(t)$ (t – номер ітерації). Його поширення безпосередньо встановлює значення виходів:

$$y_i(0) = x_i, \ i = 0...n-1,$$

тому позначення на схемі мережі вхідних сигналів у явному вигляді носить чисто умовний характер. Нуль у дужках справа від y_i означає нульову ітерацію в циклі роботи мережі.

3. Обчислюються новий стан нейронів

$$S(t+1) = \sum_{i=1}^{n} w_{ij} y_j(t), \ j - 0,1,...,n-1 \qquad (14.21)$$

і нові значення виходів

$$y_j(t+1) = f(S_j(t+1)), \qquad (14.22)$$

де *f* – передатна функція у вигляді порогової, наведена на рис. 14.17.

4. На цьому етапі перевіряємо, чи відбулися зміни в вихідних значеннях після останньої ітерації. Якщо так, то повертаємося до кроку 2, інакше (якщо вихідні значення стабілізувалися) – завершуємо процес. У цьому випадку вихідний вектор являє собою зразок, який найкраще відповідає вхідним даним.

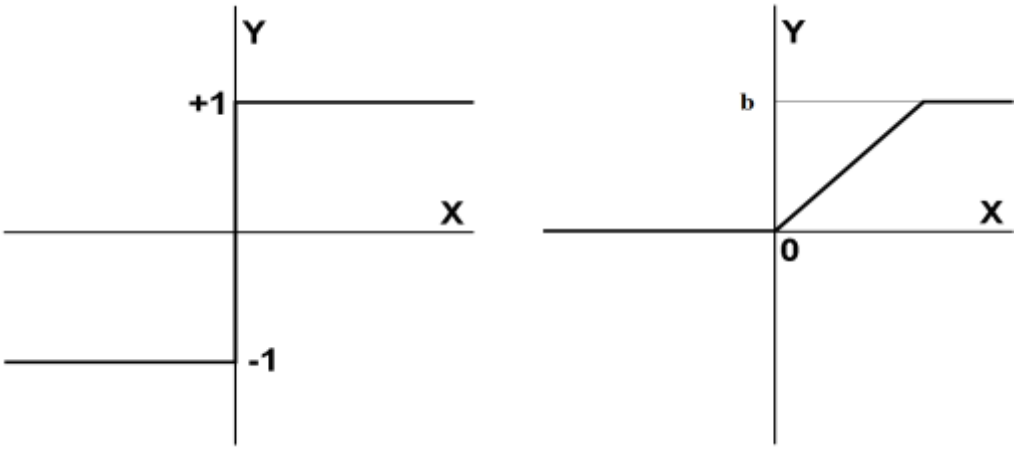


Рисунок 14.17 – Вигляд передаточної функції нейрона

В деяких випадках мережа не здатна виконати розпізнавання і надає на виході неіснуючий образ. Це виникає через обмеженість можливостей мережі. Для мережі Хопфілда кількість збережених образів *m* не може перевищувати величину 0.15**n* (де *n* - кількість нейронів вихідного шару). Крім того, якщо два образи, наприклад, А і Б, дуже схожі між собою, вони можуть викликати у мережі перехресні асоціації. Іншими словами, подача на входи мережі вектора Б може призвести до появи на виходах мережі вектора А і навпаки.

14.4.5 Мережа Хемінга

Мережа Хемінга, що є розширенням мережі Хопфілда, створена Річардом Ліппманом у середині 80-х років. Вона використовується як класифікатор для векторів двійкових входів, базуючись на мінімальній похибці, яка вимірюється за відстанню Хемінга – числом неодинакових бітів у двох векторах фіксованої довжини. Один вхідний вектор – це незашумлений зразок, а інший – спотворений. Вихідний вектор подає класи, до яких належать зразки. Під час навчання вектори призначаються до категорій, для яких відстань між зразковими векторами та поточним вектором є мінімальною.

В мережі Хемінга є три шари: вхідний з кількістю вузлів, що відповідає кількості ознак; прошарок категорій з кількістю вузлів, що відповідає кількості класів; та вихідний прошарок, що має таку саму кількість вузлів, як і прошарок категорій.

Ця мережа має просту архітектуру прямого поширення, де вхідний прошарок повністю з'єднаний з прошарком категорій. Кожен нейрон у

прошарку категорій зв'язаний з кожним нейроном у тому самому прошарку та з вихідним нейроном. Вихід з прошарку категорій у вихідний прошарок формується через конкурентну динаміку.

Процес навчання мережі Хемінга відповідає методології Хопфілда, як зображено на рисунку 14.18. У вхідний прошарок подається бажаний навчальний зразок, і на виході вихідного прошарку отримується значення бажаного класу, до якого належить вектор. Вихід містить лише значення класу, до якого належить вхідний вектор. Рекурсивна природа прошарку Хопфілда забезпечує можливість корекції ваг всіх зв'язків.

Алгоритм роботи мережі Хемінга складається з таких кроків:

1. На етапі ініціалізації вагові коефіцієнти першого шару та поріг передатної функції отримують такі значення:

$$w_{ik} = \frac{x_i^k}{2}, i = \overline{0..n-1}, k = \overline{0..m-1}, \quad b_k = \frac{n}{2}. \tag{14.23}$$

У виразі (14.23) – x_i^k - *i*-ий елемент *k*-ого зразка

Для гальмувальних синапсів другого шару вагові коефіцієнти призначають такими, що дорівнюють деякій величині $0 < v < 1/m$. Синапс нейрона, який пов'язаним з його ж виходом, має вагу +1.

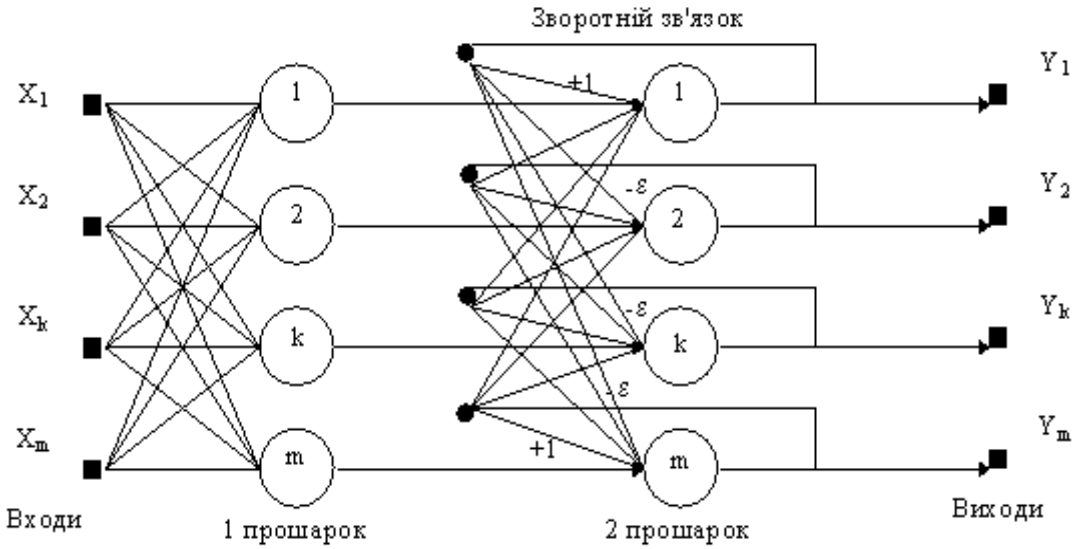


Рисунок 14.18 – Структурна схема мережі Хемінга

2. Після подання на входи мережі невідомого вектора (x_1,.. x_i,.. x_n) розраховуються стани нейронів першого шару:

$$y_j^{(1)} = S_j^{(1)} = \sum_{i=1}^{n} w_{ij} \cdot x_i + b_j, \quad j = \overline{0..m-1}. \tag{14.24}$$

Верхній індекс у дужках виразу (14.24) вказує номер прошарку.

Після цього отримані значення станів нейронів першого шару ініціалізують значення сигналів на виходах другого прошарку:

$$y_j^{(2)} = y_j^{(1)}, j = 0,1, ..., m-1 \qquad (14.25)$$

3. Обчислюються нові стани нейронів другого прошарку

$$S_j^{(2)}(t+1) = y_j(t) - \varepsilon \sum_{j=1}^{m} y_k^{(2)}(t), \quad k \neq j, j = \overline{1..m} \qquad (14.26)$$

та значення їх виходів:

$$y_j^{(1)}(t+1) = f(S_j^{(2)}(t+1)). \qquad (14.27)$$

Передатна функція *f* має форму порога, причому його величина *b* має бути достатньо великою, щоб ніякі значення аргументу передатної функції не призводили до насичення.

4. Здійснюється перевірка, чи змінилися виходи нейронів другого шару під час останньої ітерації. Якщо так – переходимо до кроку 3. В іншому випадку – завершуємо процес.

Роль першого прошарку у мережі Хемінга є умовною: після використання його вагових коефіцієнтів на першому етапі, мережа більше не звертається до нього, тому цей прошарок можна взагалі прибрати з мережі.

Мережа Хемінга має кілька переваг порівняно з мережею Хопфілда. Вона реалізує оптимальний класифікатор з мінімальними помилками у випадку, коли помилки вхідних бітів є випадковими та незалежними. Окрім того, для роботи мережі Хемінга потрібна менша кількість нейронів, оскільки середньому прошарку потрібен лише один нейрон на клас, замість нейрона на кожен вхідний вузол. І нарешті, мережа Хемінга не здійснює тих неправильних класифікацій, які можуть відбуватись у мережі Хопфілда. Загалом, мережа Хемінга є і швидшою, і точнішою від мережі Хопфілда.

Контрольні питання та завдання

1. За яким принципом побудовано штучні нейронні мережі?
2. Опишіть будову мозку людини з погляду реалізації ним процесів обробки інформації.
3. Який біологічний елемент є базовим в будові мозку людини?
4. Які основні переваги обробки інформації мозком порівняно з комп’ютером?
5. Наведіть означення штучної нейронної мережі.

6. З яких елементів складається штучна нейронна мережа?
7. Опишіть будову біологічного нейрона.
8. Опишіть будову і принцип роботи штучного нейрона.
9. Які типи передатних функцій використовують для моделювання штучного нейрона?
10. Які переваги сигмоїдної передатної функції нейрона?
11. В яких випадках використовують в штучному нейроні жорстку порогову передатну функцію?
12. Назвіть основні переваги і недоліки штучних нейромереж.
13. Назвіть основні сфери застосування нейромережевих технологій.
14. За якими основними ознаками здійснюють класифікацію нейромереж?
15. Чим відрізняється повнозв'язана штучна нейрона мережа від слабозв'язаної?
16. Нарисуйте структурну схему багатошарової нейронної мережі.
17. Як поділяються зв'язки між нейронами за ознакою напрямку з'єднання?
18. На які класи поділяються нейромережі за ознакою архітектури зв'язків?
19. Назвіть приклади послідовних і рекурентних мереж.
20. Які типи машинного навчання нейронних мереж вам відомі?
21. Яка різниця між контрольованим і неконтрольованим навчанням нейронної мережі?
22. Сформулюйте математичну постановку задачі навчання нейромережі.
23. Яка основна ідея навчання нейромережі методом backpropagation?
24. Дайте вербальний опис алгоритму навчання штучної нейромережі методом backpropagation.
25. Опишіть принцип роботи перцептрона Розенблатта.
26. Опишіть принцип роботи нейромережі Кохонена.
27. Опишіть принцип роботи нейромережі Хопфілда.
28. Опишіть принцип роботи нейромережі Хемінга.
29. Опишіть принцип роботи загорткової нейромережі.

ЛІТЕРАТУРА

1. Bykov N. M., Kuzmin I. V., Yakovenko A. I. Development of effective strategy of pattern recognition // *Proceedings of SPIE*, 2001, Vol. 4225, pp.76–83.

2. Brink H., Richards J., Fetherolf M. Real-World Machine Learning. 1st Edition / Henrik Brink, Joseph W. Richards, Mark Fetherolf . New Yourk: Manning Publishing Co, 2016. 264 p., ISBN 978-1617291920.

3. Casella G., Fienberg S. An Introduction to Statistical Learning with Applications in R. New York: Springer, 2015.

4. Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning (Adaptive Computation and Machine Learning series), The MIT Press, 2016. 800 p.

5. T. Hastie, R. Tibshirani, J. Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed. Springer-Verlag, 2009. 746 p.

6. T. Hastie, R. Tibshirani. Statistical Learning with Sparsity. The Lasso and Generalizations. McGraw-Hill Irwin, 2015.

7. M. H. Kutner, C. J. Nachtsheim, J. N. Neter. Applied Linear Regression Models. McGraw-Hill Irwin, 2005. 1396 p

8. Tom M. Mitchell. Mashine Learning. McGraw-Hill Science/Engineering/Math, 1997. p. 432.

9. Muller A., Guido S. Introduction to Mashine Learning with Python. A Guide for Data Scientists. Beijin - Boston-Tokyo: O'Reily, 2018. 386 p.

10. Richert W., Coehlo L. Pedro. Building Mashine Learning with Python / Willi Richert, Luis Pedro Coehlo. – NY: Packt, 2013. – 290 p.

11. Russel S. J., Norvig P. Artificial Intelligence. A modern Approach. – Second Edition. New Jersey: Prentice Hall, 2003. 1408 p.

12. Smith L. An Introduction to Neural Networks / Leslie Smith – University of Stirling, 2008. – Електронний ресурс. Режим доступу: http://www.cs.stir.ac.uk/~lss/NNIntro/ InvSlides.html .

13. Brian Steele, John Chandler, Swarna Reddy. «Algorithms for Data Science». Springer. 2016. 430 p.

14. Биков М. М. Методичні вказівки до виконання лабораторних робіт з дисципліни «Інтелектуальні засоби систем автоматизації» для студентів спеціальності 151 – Автоматизація та комп'ютерно- інтегровані технології / Уклад. М. М. Биков, Т. В. Грищук, Г. Ю. Дерман, В. В. Ковтун. Вінниця : ВНТУ, 2018. 48 с.

15. Зікратий С. В. Системи штучного інтелекту: практикум. ІваноФранківськ : ІФНТУНГ, 2013. 106 с.

16. Зікратий С. В., Паньків Х. В. Системи штучного інтелекту: лабораторний практикум. Івано-Франківськ : ІФНТУНГ, 2015. 91 с.

12. Зікратий С. В. Системи штучного інтелекту. Курсове проектування. Івано-Франківськ: ел. варіант, 2018. 68 с.

Навчальне електронне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Микола Максимович Биков, Вячеслав Васильович Ковтун,

Тетяна Вікторівна Грищук

ОСНОВИ ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ
Частина 2. Технології машинного навчання

НАВЧАЛЬНИЙ ПОСІБНИК

Рукопис оформив *М. Биков*

Редактор *Т. Старічек*

Оригінал-макет виготовила *Т. Старічек*

Підписано до видання 15.07.2024 р.
Гарнітура Times New Roman.
Зам. № Р2024-131.

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95, м. Вінниця, 21021.
press.vntu.edu.ua;
E-mail: irvc.ed.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009.