МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Машинне навчання та обробка сигналів в біомедичних електронних системах. Конспект лекцій

Рекомендовано Методичною радою КПІ ім. Ігоря Сікорського
як навчальний посібник для студентів,
які навчаються за спеціальністю 153 «Мікро- та наносистемна техніка»,
освітньою програмою « Електронні мікро- і наносистеми та технології»

Київ
КПІ ім. Ігоря Сікорського
2020

Машинне навчання та обробка сигналів в біомедичних електронних системах. Конспект лекцій [Електронний ресурс]: навч. посіб. для студ. спеціальності 153 «Мікро- та наносистемна техніка», освітньої програми «Електронні мікро- і наносистеми та технології» / КПІ ім. Ігоря Сікорського; уклад.: К.О. Іванько, А.О. Попов, Н.Г. Іванушкіна. – Електронні текстові данні (1 файл: 5947 Кбайт). – Київ: КПІ ім. Ігоря Сікорського, 2020. – 97 с.

Гриф надано Методичною радою КПІ ім. Ігоря Сікорського (протокол № від. 2020 р.)
за поданням Вченої ради факультету електроніки (протокол № від .2020 р.)

Електронне мережне навчальне видання

Машинне навчання та обробка сигналів в біомедичних електронних системах. Конспект лекцій

Укладачі: *Іванько Катерина Олегівна,* к.т.н., доц.
Попов Антон Олександрович, к.т.н., доц.
Іванушкіна Наталія Георгіївна, к.т.н., доц.
Відповідальний редактор *Тимофєєв Володимир Іванович*, д-р техн. наук, проф.
Рецензентка: *Хіжняк Тетяна Андріївна*, к.т.н., доц.

Навчальний посібник присвячено допомозі студентам при вивченні навчальної дисципліни «Машинне навчання та обробка сигналів в біомедичних електронних системах». Викладено теоретичні відомості та приклади задач, присвячених обробці, перетворенню, аналізу та класифікації біомедичних сигналів і зображень, розробки прикладних програм обробки біомедичних сигналів і зображень.

The manual is dedicated to helping students study the discipline "Machine learning and signal processing in biomedical electronic systems." Theoretical information and tasks of processing, transformation, analysis and classification of biomedical signals and images, development of applied programs of processing of biomedical signals and images are provided.

© КПІ ім. Ігоря Сікорського, 2020

ЗМІСТ

Вступ

Мета дисципліни «Машинне навчання та обробка сигналів в біомедичних електронних системах» – отримання теоретичних знань та навичок практичного застосування обробки біомедичних сигналів та зображень для машинного навчання у біомедичних електронних системах. Дисципліна «Машинне навчання та обробка сигналів в біомедичних електронних системах» базується на знаннях, набутих під час вивчення вищої математики, основ техніки вимірювань, схемотехніки, теорії сигналів, імовірнісних основ обробки даних, біофізики.

Згідно з вимогами освітньо-професійної програми студенти після засвоєння навчальної дисципліни мають отримати навички обробки, перетворення, аналізу та класифікації біомедичних сигналів і зображень, розробки прикладних програм обробки біомедичних сигналів і зображень, для аналізу біопотенціалів серця та мозку, тиску крові, пульсу, зовнішнього дихання та ін. Студенти зможуть реалізовувати методи обробки та аналізу біомедичних сигналів і зображень та застосовувати методи машинного навчання для інтелектуального аналізу даних в біомедичних електронних системах. Ці знання є необхідними для формування світогляду фахівців з електронної техніки біомедичного призначення.

ТЕМА №1
ЗАСАДИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

З задачами аналізу даних та розпізнавання об’єктів, ситуацій, явищ або процесів кожна людина постійно стикається впродовж всього свого існування, це є найбільш розповсюджена задача, яку людина повинна вирішувати кожної миті свого життя. Коли людина переходить вулицю, необхідно розрізняти кольори світлофора, коли ми хочемо прочитати книжку або рукописний текст – треба знати, як виглядають букви, коли ми хочемо знайти знайомого у натовпі – необхідно вміти розрізняти обличчя, коли ми їдемо за кермом автомобіля – треба вміти аналізувати ситуацію на дорозі та передбачати її. Людина робить всі ці операції завдяки своєму мозку, який дає можливість сприймати навколишній світ, аналізувати все, що відбувається навколо, робити висновки та керувати діями і думками.

Аналіз даних за допомогою технічних систем найчастіше використовується в таких галуззях:

– діагностика захворювань;

– прогнозування поведінки часових рядів (прогноз розвитку підприємств, фінансові ряди);

– системи інтелектуального пошуку, системи фільтрації спама у мережі Інтернет;

– розпізнавання друкованих символів, жестів та мови;

– системи контекстної реклами;

– біометричні системи (ідентифікація особи з використанням відбитків пальців, рисунку мережі капілярів, рогівки ока тощо).

Сукупність методів, які використовуються для аналізу даних різної природи, називається **data mininig –** інтелектуальний аналіз даних – знаходження в *даних* про досліджувані об’єкти корисної та доступної для інтерпретації *інформації*. Інтелектуальний аналіз даних виконується з використанням машинного навчання.

Машинне навчання – побудова алгоритмів, які можуть навчатися.

Один з засновників машинного навчання Артур Самуель, який зробив першу в світі комп'ютерну гру в шахи, називав машинне навчання «галуззю, яка дає комп'ютерам здатність навчатися без наявності в них точної програми». Більш формальне і загальноприйняте визначення машинного навчання було було дане відомим спеціалістом з штучного інтелекту Томом Мітчелом: можна сказати, що комп'ютерна програма навчається на основі свого досвіду E відносно деякого класу задач T та міри ефективності P, якщо її ефективність по виконанню задач із T, яка міряється з допомогою P, підвищується із збільшенням E («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»).

Термін «навчатися» в даному контексті використовується в тому сенсі, що алгоритм, який будується, зможе аналізувати дані так, як треба досліднику. З точки зору машинного навчання питання «Чи може машина думати?» треба замінити на питання «Чи може машина робити те, що роблять люди (як думаючі істоти)?» і дати на нього ствердну відповідь.

Машинне навчання – комплексна галузь знань, яка знаходиться на межі теорії штучного інтелекту, теорії сигналів, математичної статистики, методів оптимізації. Машинне навчання буває двох видів:

- Індуктивне – при якому машина вчиться виявляти закономірності в експериментальних даних. В результаті отримують системи класифікації.

- Дедуктивне – при якому знання експертів-людей формалізуються та переносяться в машину. В результаті створюють експертні системи.

Задачі, що вирішуються з використанням машинного навчання:

– *розпізнавання образів* – знаходження групи, до якої належить об'єкт;

– *прогнозування* – знаходження одних параметрів об'єкта на основі інших його параметрів;

– *зниження розмірності даних* – виявлення найбільш значущих характеристик об'єктів;

– *пошук асоціативних правил* – знаходження залежностей між об’єктами та подіями.

В техніці постає задача автоматизації різних складних процесів та операцій, зокрема, процесів прийняття рішень. Розпізнавання образів є ключовим для цього. Відомо, що всі об’єкти реального світу представлені в технічних системах не безпосередньо, а в результаті вимірювання деяких сигналів. Все, що система «знає» про об’єкт дослідження – це сигнали, які супроводжують існування об’єкта, надходять від об’єкта, вимірюються, реєструються, обробляються та аналізуються. Отже, до того, як робити висновки про об’єкти: розпізнавати, класифікувати – треба проаналізувати сигнали та отримати параметри цих сигналів. На основі вивчення параметрів сигналів робиться висновок щодо того, який об’єкт ми досліджуємо та до якого класу його слід віднести. Саме через це методи теорії сигналів є інструментом, завдяки якому стає можливим розпізнавання образів.

Теорія розпізнавання образів – галузь знань, яка досліджує методи описання та класифікації об’єктів: явищ, процесів, предметів, сигналів та ситуацій.

Загалом, розпізнавання об’єктів використовують скрізь, де треба якимось чином організувати отримані дані. Після вибору належного принципу розпізнавання, всі об’єкти необхідно розділити на класи, які мають спільні характеристики. Розпізнавання виконується в результаті перетворення вхідної інформації, в якості якої розглядають певні ознаки об’єктів, що розпізнаються, у вихідну інформацію, яка є висновком про те, до якого класу відноситься образ, що розпізнається.

Об’єкт – це частина матеріального світу, яка цікавить дослідника та пізнається ним в ході практичної діяльності (дослідів та експериментів). Треба ще раз наголосити, що за допомогою технічних систем об’єкти можна досліджувати не безпосередньо, а лише на основі аналізу сигналів, якими вони описуються. Тому при розпізнаванні об’єктів насправді виконують розпізнавання сигналів від об’єктів.

Системам розпізнавання образів у біомедичних експертних системах притаманна наступна функціональна схема (рис. 1.1). Вхідні дані, що в загальному випадку реєструються з пацієнта та підлягають подальшому розпізнаванню, подаються на вхід системи і піддаються ***попередній обробці*** з метою їх перетворення в необхідний для наступного етапу вигляд. Під час попередньої обробки відбувається придушення паразитних складових досліджуваного сигналу. При кодуванні інформації для передачі каналом зв'язку попередня обробка включає також стиснення-відновлення даних.

На ***етапі аналізу*** створюється формальний опис біомедичного сигналу пацієнта, який називається **образом сигналу**. Образ несе в собі інформацію, яка є найбільш істотною з точки зору якості прийняття рішення при класифікації. Крім того, створення формального опису пов'язано з процедурою подальшої редукції даних, яка на відміну від стиснення даних в процесі попередньої обробки грунтується не на надмірності інформації, а на оцінці корисності інформації для правильної класифікації. Це етап виділення з усієї множини ознак найбільш інформативних, характерних для певного захворювання ознак.

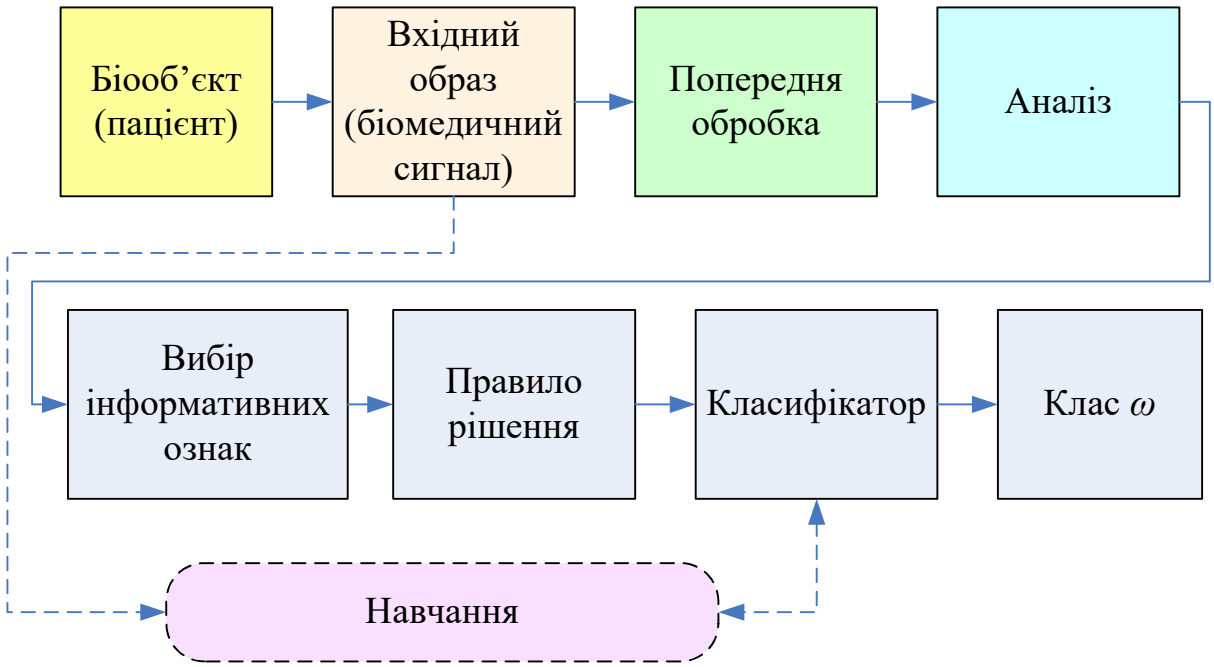


Рис. 1.1. Функціональна схема системи розпізнавання образів

Під ***класифікацією*** розуміють процедуру віднесення сигналів до певних сукупностей, образи яких мають загальні властивості. Такі сукупності називаються класами, а пристрої, за допомогою яких виконується віднесення сигналів ─ класифікаторами. Можливість розпізнавання спирається на схожість однотипних об'єктів. Звідси виникає поняття класифікації ─ розбиття всієї множини об'єктів на підмножини-класи, елементи яких мають деякі схожі властивості, що відрізняють їх від елементів інших класів. І, таким чином, завданням розпізнавання є віднесення розглянутих об'єктів або явищ по їх опису до потрібних класів. На основі правила прийняття діагностичного рішення формується відповідь класифікатора, наприклад, системою видається назва хвороби (рис. 1.2). Класифікатор має n входів (для кожної ознаки свій вхід) і єдиний вихід, на якому за правилом прийняття рішення $d(X)$ виникає сигнал ω_j, рівень якого відповідає одному з K класів. Проблема класифікації образів полягає у визначенні оптимальних границь або у розробці процедур прийняття рішення для віднесення даних до різних класів образів з використанням векторів ознак.

Клас являє собою деяку сукупність (підмножину) об'єктів, що володіють близькими властивостями. В якості ознак при розпізнаванні образів у медичній експертній системі можуть виступати показники пацієнта, наприклад, лабораторні дані, дані інструментальних досліджень, скарги пацієнта і т.д. Прикладом класів можуть служити захворювання. Кінцевою метою аналізу біомедичних сигналів є класифікація даного сигналу з метою віднесення його до однієї з кількох відомих категорій та отримання діагностичного рішення, яке відноситься до стану пацієнта.

Віднесення об’єкту до того чи іншого класу проводиться в результаті групування. Віднести об’єкти до тої чи іншої групи можна на основі аналізу їх характеристик, та з використанням заздалегідь визначених критеріїв. Критерії в широкому сенсі – деякі чисельні міри схожості одних об’єктів на інші, за величиною яких можна судити про те, чи відносяться дані об’єкти до однієї групи, чи ні.

Можна виділити такі підходи до групування об’єктів у класи: класифікація та кластеризація.

Класифікація – віднесення об’єктів до однієї з наперед визначених груп. Під тим, що клас наперед визначений, мається на увазі, що до проведення класифікації відома кількість класів, до яких потенційно можуть відноситися всі об’єкти, а також відомими є ознаки, на основі яких можна віднести конкретний об’єкт до того чи іншого класу.

Кластеризація – віднесення об’єктів у змістовні або корисні групи (кластери) без використання апріорної інформації про групи, а лише ґрунтуючись на тій інформації, яка міститься в самих даних про об’єкти. При проведенні кластеризації кількість класів заздалегідь невідома, а визначається вона за різними математичними методами на основі аналізу параметрів об’єктів, які відомі в результаті експериментів. Кластеризація базується лише на тих даних, які є в розпорядженні технічної системи.

Крім описаних вище етапів системи розпізнавання образів передбачають ***етап навчання системи***. Метою навчання системи є формування в її пам'яті набору відомостей, необхідних для розпізнавання класів вхідних даних.

Ознаки об’єктів – це набір характеристик, якими описується об’єкт, що підлягає розпізнаванню. Ознаки можуть бути кількісними або якісними. До кількісних ознак відносяться числові значення характеристик (вага, довжина, сила струму, потужність та т. ін.), а до якісних – характеристики, що не мають числового вираження, а є лінгвістичними змінними (сильний, високий, швидкий, різкий та т.і н.) Ознаки повинні бути такими, які описують саме ті властивості об’єктів, що істотні для даної мети класифікації. Тобто ознаки, які будуть корисні для групування об’єктів, повинні різнитися у об’єктів різних класів.

Досить часто у БЕС мають справу з ознаками об’єктів, які отримані в результаті дослідження сигналів. Це можуть бути амплітуди та тривалості деяких коливань, форма коливань, ймовірнісні характеристики сигналів (закони розподілу густини ймовірностей, значення моментів тощо), кількість перетинів

нульової лінії, розташування піків АКФ, розташування локальних максимумів в спектрі сигналів, потужності в певних діапазонах частот та т.ін. При розпізнаванні зображень ознаками можуть виступати розміри областей, компоненти кольору областей, середня яскравість області, гістограми, спектральні характеристики, топологічні ознаки (поєднання областей), морфологічні ознаки (форма областей) та т.ін. Загалом, все, що можна розрахувати, маючи сигнал та знаючи математичний апарат методів теорії сигналів, може бути використане в якості ознак для розпізнавання. Головне, щоб дані ознаки різнились для різних класів об’єктів.

Ознаки можна розділити на:

Ймовірнісні ознаки – випадкові значення яких розподілені по всім класам об’єктів, при цьому рішення щодо належності даного об’єкта до того чи іншого класу можуть прийматися лише на основі конкретних значень цих ознак, отриманих в результаті досліду.

Детерміновані (логічні) ознаки – елементарні висловлювання, які можуть приймати лише два значення істинності: «так» або «ні». Як правило є такими, що не мають кількісного значення, а є судженнями якісного характеру щодо наявності чи відсутності певних властивостей у об’єктів. Також логічними є такі ознаки, для яких важливою є не величина ознаки, а лише факт потрапляння її в заданий інтервал. Прикладом можуть бути симптоми в медичній діагностиці (характеристика болю в горлі), вид двигуна у літака (реактивний, поршневий, турбогвинтовий) та т.ін.

В системах розпізнавання образів всі об’єкти треба представити таким чином, щоб їх було зручно досліджувати – класифікувати чи проводити кластеризацію. За якою б схемою не проводилося групування об’єктів, все, що автоматична система знає про об’єкт – це його характеристики, які отримані в результаті аналізу сигналів. Отже, об’єкт представлений в системі не безпосередньо, а опосередковано – набором ознак. Ознаки об’єкта, якими ми користуємось для розпізнавання – це все, що надається автоматичній системі

для роботи, тобто всі реальні об’єкти зводяться нами до набору ознак – образу об’єкта.

Образ – представлення об’єкта, який підлягає класифікації чи кластеризації, у вигляді сукупності ознак. У розпізнаванні образів ми розглядаємо для кожного об’єкта деякий набір з N ознак – характеристик об’єктів. Кожна характеристика (ознака) – це число, яке є параметром об’єкта та характеризує вираженість у конкретного об’єкта певної властивості.

Наприклад, нехай об’єктом є деякий сигнал $U(t)$ тривалістю 1 секунда. Однією з можливих характеристик для цього сигналу є спектральна густина потужності (СГП). Якщо для сигналів, які відносяться до різних класів, СГП суттєво відрізняється, то її можна обрати як ознаку для розпізнавання. Параметром, за яким проводять класифікацію можна обрати, наприклад, середнє значення СГП для всього спектру або окремо для деяких частотних проміжків. Ще одним параметром, який характеризуватиме СГП сигналу, може бути значення частоти f_{edge}, такої, що 90% потужності сигналу знаходиться в тій частині спектру, яка розташована на частотах від 0 до f_{edge} Гц. Можна розрахувати інші параметри, які будуть чисельно описувати властивості СГП кожного об’єкта-сигнала, і всі вони потенційно можуть бути використані в якості ознак образу.

Після того як були отримані інформативні ознаки, можна представити вектор ознак у формі

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix}.$$

Ознакові змінні або ознаки можуть описувати як кількісні так і якісні характеристики сигналів. Коли величини x_i є числами, вектор **x** є точкою в *n*-вимірному просторі ознак. Найбільш часто вектори образів розглядаються у

вигляді точок n-мірного евклідова простору. Безліч образів, що належать одному класу, відповідає сукупності точок, розсіяних в деякій області. Для випадку двох класів (C_1 і C_2) вектори образів матимуть вигляд X = [x_1, x_2], а кожен вектор образу можна вважати точкою двовимірного простору. Очікується, що вектори схожих об'єктів формують кластери. Для ефективної класифікації образів необхідні такі ознаки, які можуть дати набори, які не перетинаються у просторі ознак, або кластери для векторів ознак (рис. 4.3). Ця обставина підкреслює важливість правильного вибору процедур попередньої обробки і виділення ознак.

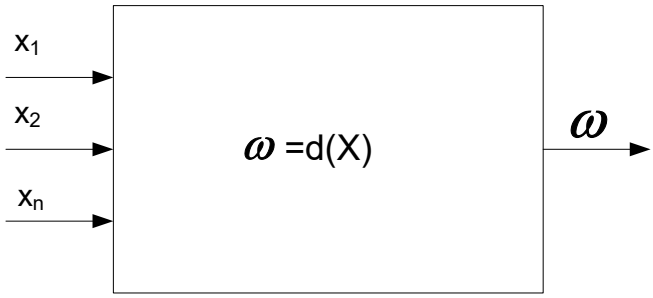


Рис.1.2. Схема класифікатора

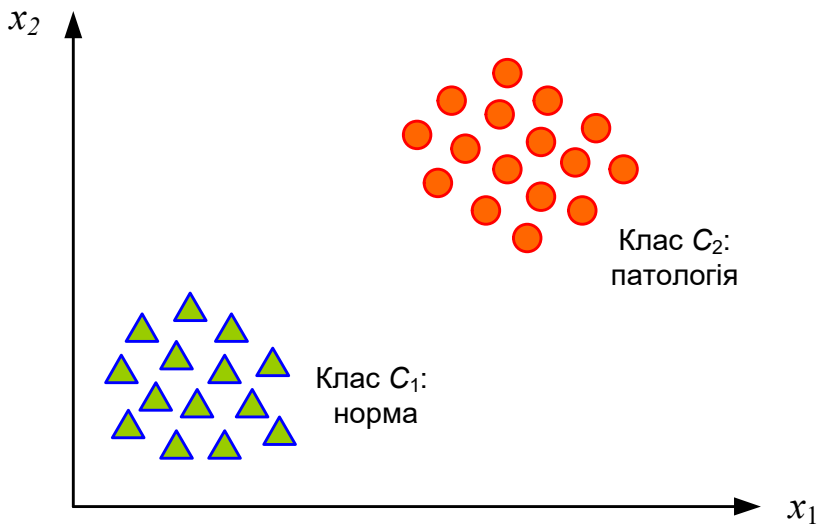


Рис. 1.3. Два кластери, що відповідають класам C_1 та C_2 і не перетинаються у двовимірному просторі ознак

Таким чином, процес розпізнавання образів складається з етапів:

- формування інформативних ознак, які створюють досліджуваний образ;
- зниження розмірності простору ознак:

- селекція ознак ─ вибір суттєвих та несуттєвих ознак;
- екстракція ознак ─ видалення несуттєвих ознак;

- вибір вирішального правила (синтез класифікатора, навчання класифікатора);

- класифікацію образів.

Методи формування інформативних ознак

Початкове формування ознак є результатом фізичного експерименту або теоретичного розгляду проблеми або явища. Формування ознак можна формалізувати як перетворення $\boldsymbol{T}$ сигналу $\boldsymbol{S(t)}$ до вектору ознак $\boldsymbol{Y}$: $Y = T\{S(t)\}$. В якості оператора $\boldsymbol{T}$ можуть виступати ортогональні перетворення (наприклад, Фур'є, Адамара, Хаара, Карунена-Лоєва і т.д.).

На етапі виділення інформативних ознак необхідно користуватися наступними правилами (рис.1.4):

- виконувати вибір величин **з мінімальною дисперсією всередині класу** для того, щоб їх розподіли не накладалися один на інший;

- вибирати величини з **максимальними відстанями між класами** (навіть якщо їх дисперсії однакові, то класифікація не викличе труднощів).

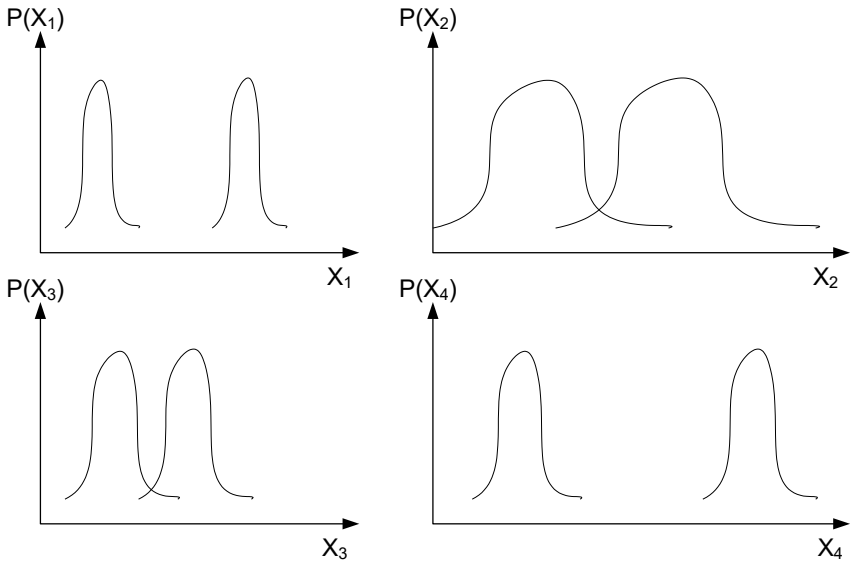


Рис.1.4. Виділення інформативних ознак за максимальними відстанями між класами

Всі значення числових ознак об'єкта можна представити у вигляді впорядкованого набору чисел $x_1, x_2, ...x_N$. Ці величини можна розглядати як координати деякого вектору $X = [x_1, x_2, ...x_N]$ з N-вимірного евклідового простору. Цей простір містить вектори, координатами яких є всі можливі комбінації параметрів об'єктів, отже кожному об'єкту відповідає вектор з цього простору. Вектори цього простору представляють собою образи об'єктів; існує взаємно однозначна відповідність між об'єктом реального світу та вектором в просторі ознак, який відповідає цьому об'єкту.

Нехай проведено дослідження і визначено, що для описання кожного об'єкта достатньо мати N ознак. Нехай проведено розбиття об'єктів на класи $\Omega_1, \Omega_2, ...\Omega_M$ та обрано ознаки. Тоді в просторі ознак можна виділити деякі області $D_1, D_2, ...D_M$, які еквівалентні класам. Ці області повинні характеризуватися залежністю: якщо об'єкт, що має ознаки $x_1^0, x_2^0, ...x_N^0$ відноситься до класу Ω_i, то точка, яка його представляє в просторі ознак, належить області D_i. В ідеальному випадку області в просторі ознак не повинні перекриватися, щоб унеможливити ситуацію, коли один об'єкт може належати двом чи більше класам одночасно. Якщо треба класифікувати новий об'єкт, то спочатку визначаються значення ознак (параметри образа). Ці числа визначають, де в просторі ознак знаходиться точка, що відповідає об'єкту. Задача класифікації зводиться до визначення, в яку саме область потрапив вектор, що відповідає образу об'єкта. Ця область в просторі ознак відповідатиме класу, до якого треба віднести сам об'єкт.

В іншій трактовці необхідно побудувати розділяючі функції $F_i(x_1, x_2, ...x_N)$, $i = \overline{1, M}$, такі, що для них виконується умова: якщо об'єкт з ознаками $x_1^0, x_2^0, ...x_N^0$ відноситься до класу Ω_i, то величина $F_i(x_1^0, x_2^0, ...x_N^0)$ повинна бути максимальною. Це можна записати математично:

$$F_i(X_i) > F_j(X_i), \quad i, j = \overline{1, M}, \quad i \neq j.$$

В такому випадку в просторі ознак границя розбиття між областями D_i, що відповідають класам Ω_i, виражається рівнянням:

$$F_i(X) - F_j(X) = 0.$$

Отже, з точки зору теорії сигналів, *образ* є вектором в просторі ознак. Числові значення ознак (параметри образа) є координатами цього вектора. Тепер можна працювати в такому просторі ознак об'єктів, залучаючи всі відомі методи математичного аналізу.

До основних методів виділення ознак для сигналів відносяться такі:

1. Аналіз сигналу в часовій області

Аналіз енергії сигналу. Метод базується на аналізі середньої енергії сигналу, яка визначається як:

$$E[n] = \frac{1}{N} \sum_{i=1+(n-1)(N-D)}^{n(N-D)+D} x[i]^2$$

де E – енергія сигналу, x – відліки вхідного сигналу, N – кількість відліків у вікні спостереження, D – зсув вікна.

Довжина кривої. Довжина кривої CL в дискретному вигляді розраховується як:

$$CL[n] = \sum_{i=1+(n-1)(N-D)}^{n(N-D)+D} \left| x[i-1] - x[i] \right|$$

де x – відліки вхідного сигналу, N – довжина вікна спостереження у відліках, D – перекриття вікон.

Довжину кривої, як ознаку для прогнозування нападів, використовують разом з середньою та нелінійною енергією сигналу, спектральною ентропією та іншими ознаками.

Нелінійна енергія. Нелінійна енергія NE сигналу визначається як:

$$NE[n] = x^2[n] - x[n-1] \cdot x[n+1]$$

де x – відліки вхідного сигналу.

Отримані значення *NE* потім осереднюється з використанням вікна Ханна:

$$ANE[n] = \frac{1}{N} \sum_{i=1+(n-1)(N-D)}^{n(N-D)+D} NE_w[i]$$

де N – довжина вікна спостереження у відліках, D – перекриття вікон у відліках.

Евклідова відстань. Евклідова відстань *ED* є мірою схожості двох послідовностей X та Y і визначається як

$$ED_{xy} = \sqrt{\sum_{i=1}^{N} (x_i - y_i)^2}$$

де N- кількість відліків у досліджуваних послідовностях

2. Аналіз результатів перетворення сигналів

(спектральний аналіз, вейвлет-аналіз)

3. Оцінка синхронізації

(фазова синхронізація, синхронізація затримок)

4. Ентропійний аналіз

(спектральна ентропія, ентропія перестановок)

5. Аналіз у фазовому просторі

Показник Ляпунова. Показник Ляпунова є нелінійною мірою середньої швидкості дивергенції/конвергенції двох сусідніх траєкторій в динамічної системи і залежить від чутливості початкових умов. Кількісно міра відстані між двома траєкторіями у фазовому просторі з початковою відстанню ΔZ_0 може бути описана як:

$$|\Delta Z(t)| \approx e^{\lambda t} |Z_0|$$

де λ - показник Ляпунова. Позитивний знак показника Ляпунова означає, що траєкторії сходяться, що зазвичай приймається як ознака того, що система є

хаотичною. Кількість показників Ляпунова дорівнює кількості вимірів у фазовому просторі.

Кореляційна розмірність. Кореляційна розмірність є мірою того, наскільки близько в деякому просторі знаходяться фазові траєкторії для деякої системи (розмірності простору, зайнятого набором випадкових точок) і визначається як

$$D_2 = \lim_{N\to\infty}\lim_{\varepsilon\to 0}\frac{d\ln C(\varepsilon)}{d\ln\varepsilon}$$

де $C(\varepsilon)$ є кореляційною сумою, яка визначається як

$$C(\varepsilon) = \frac{1}{N^2}\sum_{\substack{i,j=1\\ i\neq j}}^{N}\Theta(\varepsilon - \left\|\vec{x}(i)-\vec{x}(j)\right\|)$$

де N - кількість станів $\vec{x}(i)$, ε - порогова відстань, $\|\cdot\|$ - норма (наприклад, евклідова), $\Theta(\cdot)$ - функція Хевісайда.

6. Отримання ознак на основі моделювання

(авторегресійне моделювання, ланцюги Маркова)

Вибір ознак полягає у виборі з усіх виділених ознак саме тих, за допомогою яких буде здійснюватися прогнозування. Блок вибору ознак може бути відсутній, оскільки для багатьох методів виділення ознак подальший вибір ознак для коректної класифікації не потрібний. Також, блок вибору ознак може бути суміщеним з блоком класифікації (наприклад, при використанні нейронних мереж). Обробка ознак проводиться при необхідності їхнього поліпшення задля подальшого аналізу і, зазвичай, використовує стандартні методи обробки сигналів, такі як згладжування сигналу, нормалізація та інші. Тому далі розглянемо лише методи вибору ознак.

Генетичні алгоритми. Генетичні алгоритмі працюють по аналогії до біологічної еволюції. На виході генетичних алгоритмів кожен результат має вигляд потоку бітів, який називається «хромосомою». Кожна хромосома складається з рядів бітів або «генів», зміст яких називається алелями. Така хромосома використовується для того, щоб визначити значущі ознаки з набору

всіх ознак. Прикладом застосування генетичних алгоритмів для вибору ознак при прогнозуванні епілептичних нападів є робота, в якій хромосома складалася з чотирьох генів, які, в свою чергу, містили інформацію про: 1) канали сигналу; 2) ознаки першого рівня; 3) ознаки другого рівня; 4) ознаки третього рівня. Ефективність кожної ознаки визначається за допомогою роздільного відношення Фішера (Fisher's discriminant ratio) для двох класів: стан перед нападом і стан, коли напад не очікується. Роздільне відношення Фішера *FDR* для k-ї ознаки визначається як

$$FDR_k = \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{(\mu_k^i - \mu_k^j)^2}{(\sigma_k^i)^2 + (\sigma_k^j)^2},$$

де N - кількість класів, μ_k^i -середнє значення k-ї ознаки i-го класу, σ_k^i - стандартне відхилення k-ї ознаки i-го класу.

Селекція ознак

Після формування ознак необхідно оцінити їх "вагу". Виконується перехід від простору $\boldsymbol{Y^m \rightarrow X^n}$ до простору з меншою розмірністю $\boldsymbol{n \le m}$. Зниження розмірності простору можна досягти двома шляхами: селекцією і екстракцією.

Селекція ознак означає виявлення несуттєвих змінних (які практично не впливають на рішення задачі класифікації). Найбільш відомими є алгоритми ознакові змінні і на кожному кроці розраховують один з критеріїв, наприклад, відстані між класами) або відношення дисперсій (дисперсія між класами / дисперсія всередині класів).

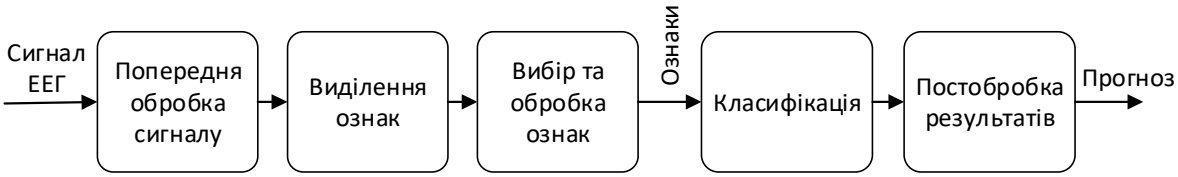
Дисперсію всередині класу можна визначити за допомогою дисперсійної матриці: $D(X) = \sum_{j=1}^{R} P(\omega_j) \int (X - \mu_j)(X - \mu_j)^T p(X/\omega_j) dX$, де $p(X/\omega_j)$ — умовні щільності ймовірностей появи образу $\boldsymbol{X}$ в класах $\boldsymbol{\omega_j}$; $P(\omega_j)$ — апріорні ймовірності появи класів $\boldsymbol{\omega_j}$ при класифікації; $\mu_j = \int_X X_p (X/\omega_j) dX$ — математичні очікування ознак образів.

Дисперсію між класами можна обчислити за формулою:

$$B(X) = \sum_{j=1}^{R-1} \sum_{S=j+1}^{R} P(\omega_j) P(\omega_s)(\mu_j - \mu_s)(\mu_j - \mu_s)^T$$

Якщо для матриці дисперсій існує зворотна матриця, то всі дисперсійні властивості можна оцінити за єдиним критерієм: $J_{j1}(X) = tr(D^{-1}(X)B(X))$ де ***tr*** — слід матриці (сума її діагональних елементів). При класифікації важливі ті ознаки, які мають меншу дисперсію всередині класу і в той же час найбільша відстань (дисперсію) між класами.

<u>ПРИКЛАД</u>: Структура системи класифікації на прикладі системи прогнозування епілептичного нападу.



Система прогнозування епілептичних нападів складається з таких основних блоків: блок попередньої обробки сигналу, блок виділення ознак, блок вибору та обробки ознак, блок класифікації на основі ознак та блок постобробки результатів. В кожному блоці системи застосовуються різні методи аналізу сигналів. Результат прогнозування залежить від властивостей цих методів, їх обмежень та способу їх використання в поєднанні один з одним. При попередній обробці сигналу ЕЕГ для прогнозування епілептичних нападів використовуються фільтри для зменшення впливу мережевої перешкоди на сигнал та для позбавлення сигналу від постійної складової. Виділення ознак полягає в розрахунку параметрів сигналу ЕЕГ, які будуть використані для подальшої класифікації та прогнозування. При прогнозуванні епілептичних нападів на основі аналізу сигналу ЕЕГ може бути одночасно використано велику кількість ознак. Ознаки можуть бути отримані як незалежно одна від одної, так і декілька методів виділення можуть бути поєднані в один (наприклад, отримання ознак після вейвлет-перетворення).

ТЕМА №2
РОЗРОБКА СИСТЕМ РОЗПІЗНАВАННЯ ОБРАЗІВ

Принципова особливість будь якої діагностики, навіть з використанням потужних технічних засобів: досягти 100% достовірності діагнозу практично неможливо в силу принципово складноусувних причин як об'єктивного, так і суб'єктивного характеру. Причини об'єктивного характеру ─ індивідуальні особливості організму, нетиповий перебіг хвороби, помилки вимірювання ознак, перекриття областей простору ознак. Причинами суб'єктивного характеру є помилки лікаря, зумовлені неправильною організацією обстежень, невірним тлумаченням результатів обстежень внаслідок втоми, психофізіологічних особливостей або некомпетентності лікаря. Це джерело помилок можна для стислості називати "людським фактором".

При проведенні обстежень з використанням технічних засобів неминуче виникають помилки оцінювання ознак, обумовлені наявністю:

• перешкод зовнішніх (наведення, вібрації тощо);

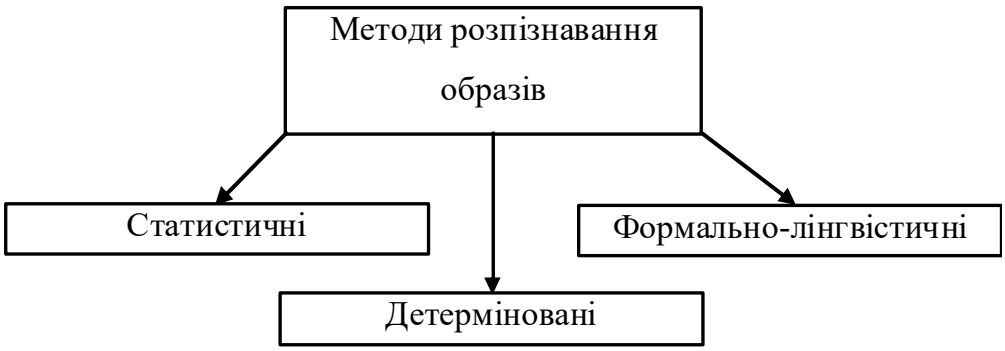
• перешкод внутрішніх (шуму, нестабільність параметрів і т.д.);

• апаратурних похибок вимірювань (систематичні і випадкові складові похибки, властиві обраному способу технічної реалізації даного методу вимірювань);

• методичних похибок вимірювань (систематична і випадкова складові).

Перекриття областей простору ознак, що характеризують різні захворювання, призводить до необхідності додаткової діагностики - фактично це означає необхідність збільшення кількості діагностичних ознак. Однак навіть збільшуючи кількість діагностичних ознак, не завжди вдається домогтися поліпшення якості діагностики. Більш того, збільшення кількості діагностичних ознак ускладнює процедуру прийняття рішення ─ в теорії розпізнавання образів цей феномен називають "прокляттям розмірності" (мається на увазі надмірно висока розмірність простору ознак, утрудняє прийняття рішення навіть з використанням потужних обчислювальних засобів).

Основні підходи до розпізнавання образів:



Детерміновані методи (методи еталонів) – використовують у випадку, коли образи об’єктів, що відносяться до одного класу, виявляють тенденцію до тісного групування в просторі ознак навколо одного типового репрезентативного еталонного образу. В цьому випадку для класифікації образів використовують функції відстаней та проводять класифікацію по критерію мінімуму відстані.

Статистичне розпізнавання образів – техніка розпізнавання, в якій класифікація виконується виходячи із статистичних характеристик класів та базується на ймовірнісних характеристиках розподілу ознак образів. Як правило, образ розглядається як випадковий вектор.

Нечітке розпізнавання – використовує апарат нечіткої логіки (fuzzy logic) для описання границь між класами та правил класифікації. В основі лежать логічні операції над множинами з нечіткими границями.

Нейронні мережі – система класифікації, яка ґрунтується на моделюванні роботи нейронів мозку. Математично представляє собою систему лінійних рівнянь зі змінними коефіцієнтами (класифікація з учителем).

Формально-лінгвістичні методи – для кожного класу об’єктів задають множину базисних елементів-примітивів, з яких можна побудувати всі члени даного класу, і правила породження нових елементів. При появі на вході системи розпізнавання нового об’єкту, який потрібно класифікувати, система визначає, чи може такий об’єкт бути побудований із заданих базисних

елементів по заданим правилам. Розділяють структурне співставлення та синтаксичний аналіз.

Методи розпізнавання образів можна розділити на три групи:

Детерміновані методи (методи еталонів) – використовують у випадку, коли образи об’єктів, що відносяться до одного класу, виявляють тенденцію до тісного групування в просторі ознак навколо одного типового репрезентативного еталонного образу. В цьому випадку для класифікації образів використовують функції відстаней та проводять класифікацію по критерію мінімуму відстані.

Статистичне розпізнавання образів – техніка розпізнавання, в якій класифікація виконується виходячи із статистичних характеристик класів та базується на ймовірнісних характеристиках розподілу ознак образів. Як правило, образ розглядається як випадковий вектор.

Формально-лінгвістичні методи – для кожного класу об’єктів задають множину базисних елементів-примітивів, з яких можна побудувати всі об’єкти даного класу, і правила породження нових об’єктів. При появі на вході системи розпізнавання нового об’єкту, який потрібно класифікувати, система визначає, чи може такий об’єкт бути побудований із заданих базисних елементів по заданим правилам. Ці методи розділяються на два види: структурне співставлення та синтаксичний аналіз.

Окремі класи представляються групами точок-образів, які розташовані близько одна до одної. Якщо точка в просторі, яка відповідає некласифікованому образу, опиняється у певній групі – об’єкт відноситься до відповідного класу. Мірою належності об’єкта до класу є відстань від його образу до групи образів. В детермінованих методах для класифікації образів використовують функції відстаней та проводять класифікацію по критерію мінімуму відстані.

Навчання системи класифікації виконується трьома методами:

- «навчання з учителем» (supervised learning) – при цьому існує тренувальна вибірка вірно класифікованих об’єктів (які класифікував експерт-«учитель»), на основі якої треба створити правило класифікації;
- «навчання без учителя» (unsupervised learning) – при цьому треба сформувати правило, яке буде використано для групування об’єктів в кластери;
- «навчання з підкріпленням» (reinforcement learning) – при якому комп’ютерна програма взаємодіє із динамічним навколишнім середовищем, в якому вона повинна досягти певної мети (наприклад, система автоматичного керування автомобілем).

1. Навчання з вчителем:
- вирішальні (дискримінантні) функції;
- функції відстані;
- правило найближчого сусіда.

2. Навчання без вчителя:
- методи пошуку кластерів.

3. Ймовірнісні моделі:
- класифікатор Байєса.

Навчання з вчителем: Дискримінантні функції

При класифікації вимагається попередня наявність навчальної вибірки об’єктів для того, щоб автоматична система змогла класифікувати конкретний об’єкт. Належність кожного об’єкта із навчальної вибірки до певного класу має бути відома заздалегідь. Визначаються ті характеристики об’єктів, значення яких для різних класів відрізняються суттєво. Ці характеристики є ознаками об’єктів, за якими буде проводитися розпізнавання. Потім значення характеристик нового об’єкта аналізуються і робиться висновок щодо його

приналежності до того чи іншого класу. Говорять, що після визначення класів, на які будуть ділитися об'єкти даною системою, складають словник ознак, який використовується для апріорного описання класів та для апостріорного описання невідомого об'єкта, який треба розпізнати.

У випадку навчання без учителя початкової вибірки класифікованих об'єктів не існує. При роботі системи розпізнавання характеристики об'єктів аналізуються, та обираються ті, які достатньо сильно відрізняються для різних груп об'єктів. Ці групи називається кластерами, а ті характеристики, які дозволили провести кластеризацію, використовуються надалі в якості ознак.

Набір векторів ознак з відомою класифікацією називають *навчальною вибіркою*. Наявність репрезентативної навчальної вибірки, що представляє набір ознак з відомою класифікацією, дозволяє знайти математичні функції, які характеризують границі між різними класами у просторі ознак. Такі функції, які називаються дискримінантними, в подальшому використовуються для розпізнавання нових векторів ознак з невідомою заздалегідь класифікацією. Такий підхід відомий як навчання з учителем. Набір векторів ознак з відомою класифікацією, який використовується для оцінки розробленого подібним способом класифікатора, називається *контрольною вибіркою*.

Узагальнена лінійна дискримінантна функція задається у формі

$$d(\mathbf{x}) = w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + w_{n+1} = \mathbf{w}^T \mathbf{x},$$

де $\mathbf{x} = (x_1, x_2, \ldots., x_n, 1)^T$ — вектор ознак, доповнений додатковим членом, що дорівнює одиниці, $\mathbf{w} = (w_1, w_2, \ldots., w_n, w_{n+1})^T$ — доповнений ваговий вектор. Тоді правило класифікації образів на два класи буде мати наступний вигляд

$$d(\mathbf{x}) = \mathbf{w}^T \mathbf{x} \begin{cases} > 0, & \text{якщо } \mathbf{x} \in C_1, \\ \leq 0, & \text{якщо } \mathbf{x} \in C_2, \end{cases}$$

де C_1 та C_2 — задані два класи, а дискримінантна функція $d(\mathbf{x})$ — границя, яка розділяє ці класи у просторі ознак.

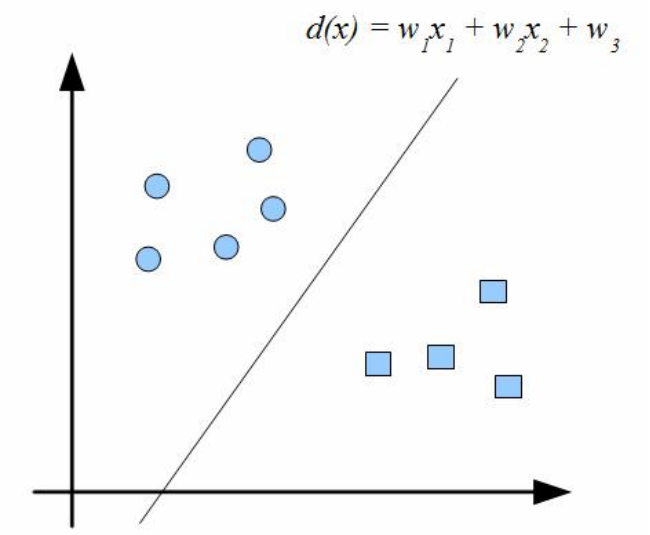


Рис.2.1. Лінійна дискримінантна функція

У загальному випадку при існуванні K класів образів C_1, C_2, ..., C_K для n-вимірного вектора ознак $\mathbf{x} = (x_1, x_2, ..., x_n)^T$ потрібно знайти K вагових вектори та K дискримінантних функцій $d_1(\mathbf{x}), d_2(\mathbf{x}), ..., d_K(\mathbf{x})$, таких що якщо образ $\mathbf{x}$ належить класу C_i, тоді

$$d_i(\mathbf{x}) > d_j(\mathbf{x}), \quad j = 1,2,....N;\ j \neq i.$$

Некласифікований образ $\mathbf{x}$ відносять до i-ого класу, якщо при підстановці $\mathbf{x}$ в усі дискримінантні функції найбільше числове значення дає функція $d_i(\mathbf{x})$. Розділяюча поверхня між двома класами описується єдиною функцією $d_{ij}(\mathbf{x}) = d_i(\mathbf{x}) - d_j(\mathbf{x}) = 0$:

$$d_{ij}(\mathbf{x}) \begin{cases} > 0, & \text{якщо } \mathbf{x} \in C_i, \\ \leq 0, & \text{якщо } \mathbf{x} \in C_j. \end{cases}$$

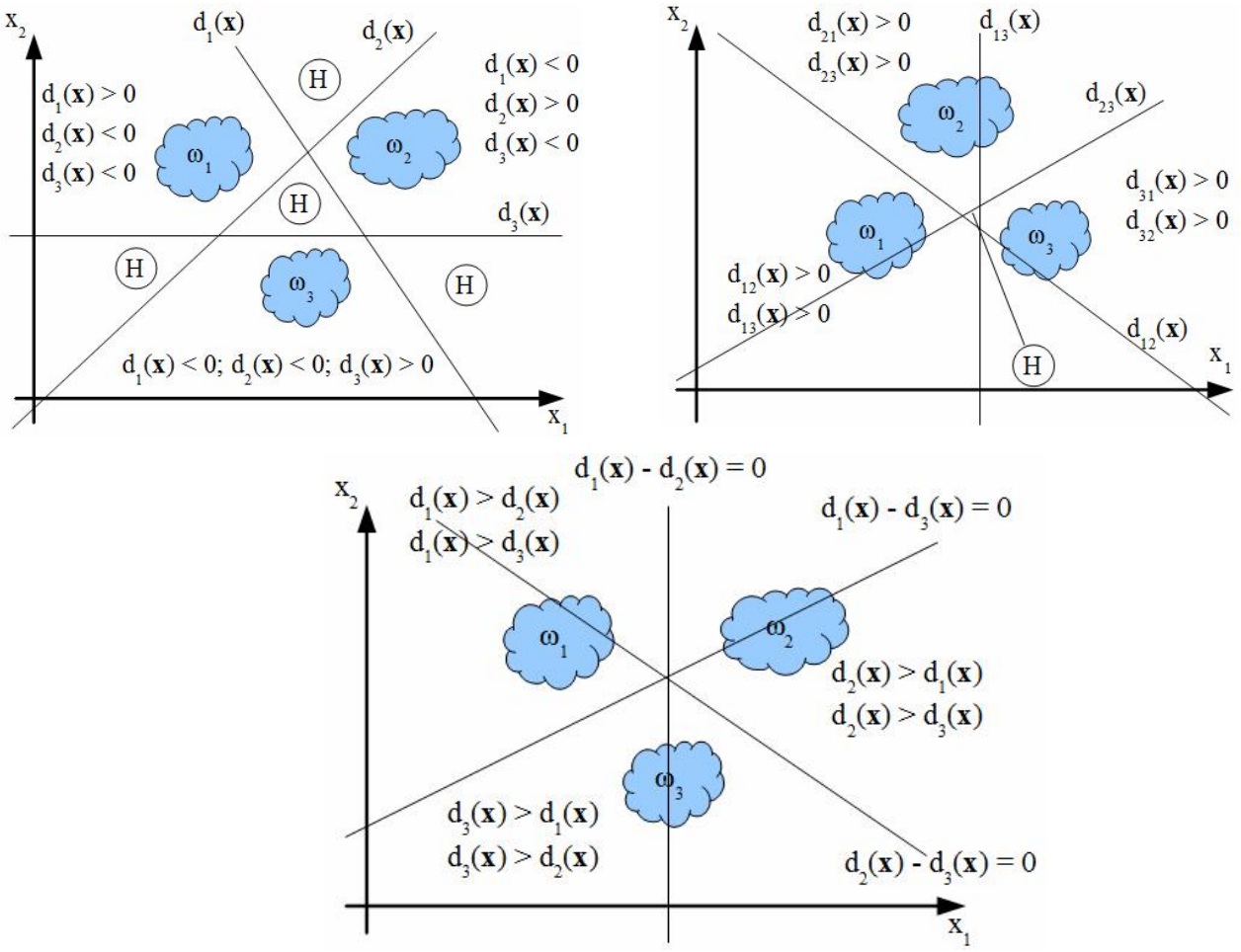


Рис. 2.2. Приклади розділення простору ознак на декілька класів

Образи, які можуть бути розділені з використанням лінійних вирішальних функцій називають лінійно роздільними. В інших випадках потрібні складні вирішальні границі з використанням вирішальних функцій, які є нелінійними функціями від векторів ознак:

$$d(\mathbf{x}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \ldots + w_K f_K(\mathbf{x}) + w_{K+1} = \sum_{i=1}^{K+1} w_i f_i(\mathbf{x}).$$

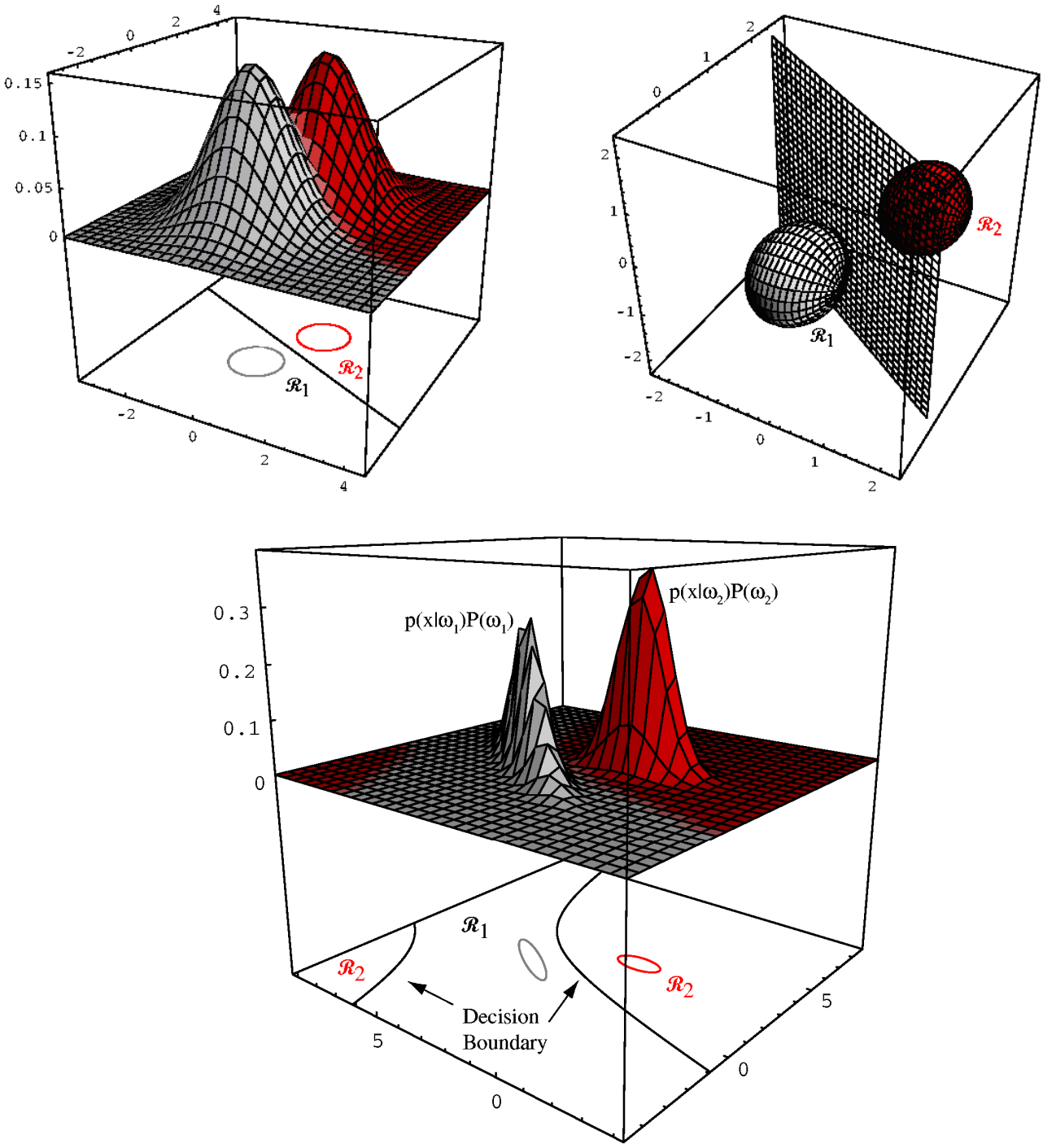


Рис. 2.3. Приклади розподілу ознак та границь розділу

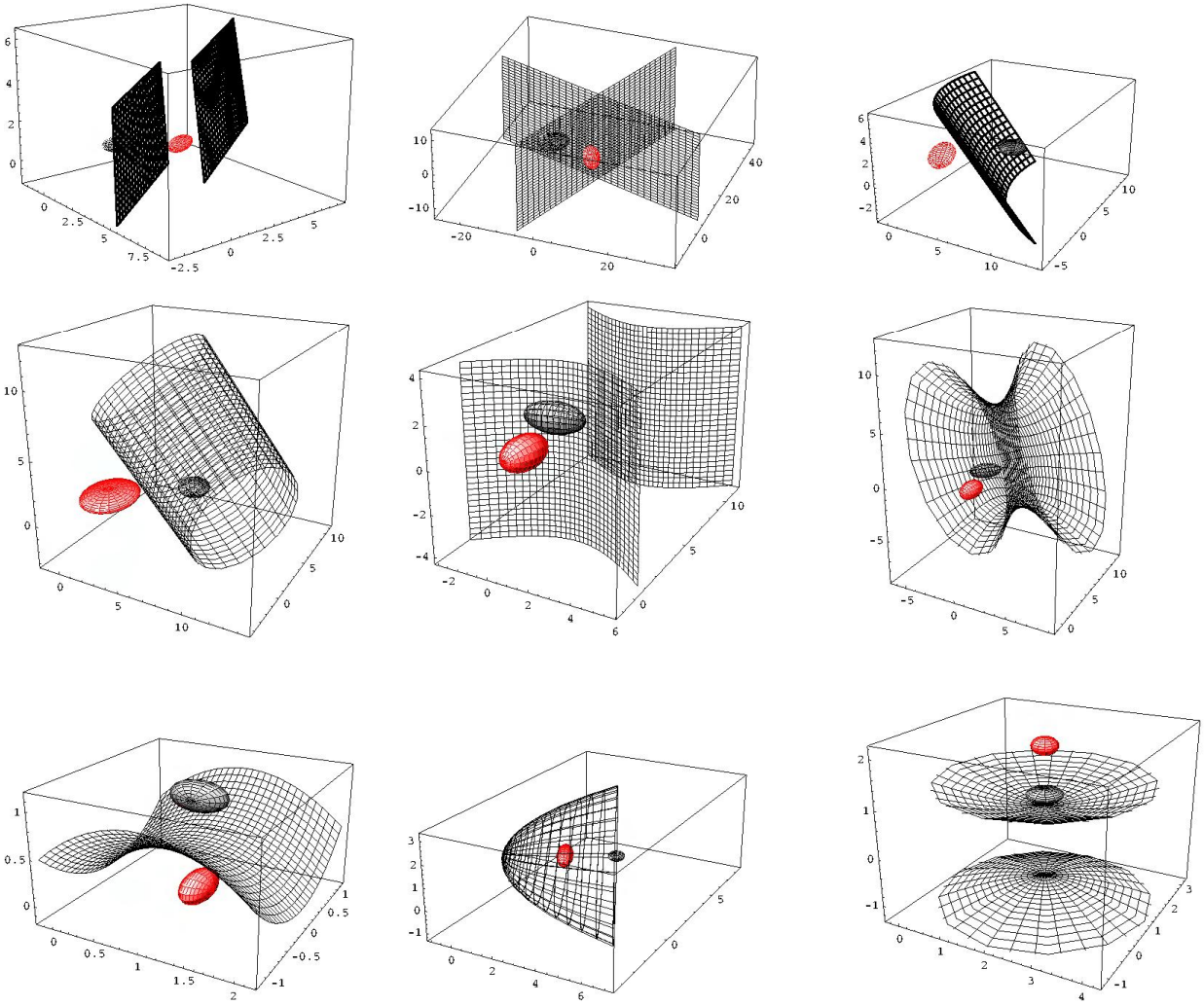


Рис.2.4. Приклади поверхонь розділу

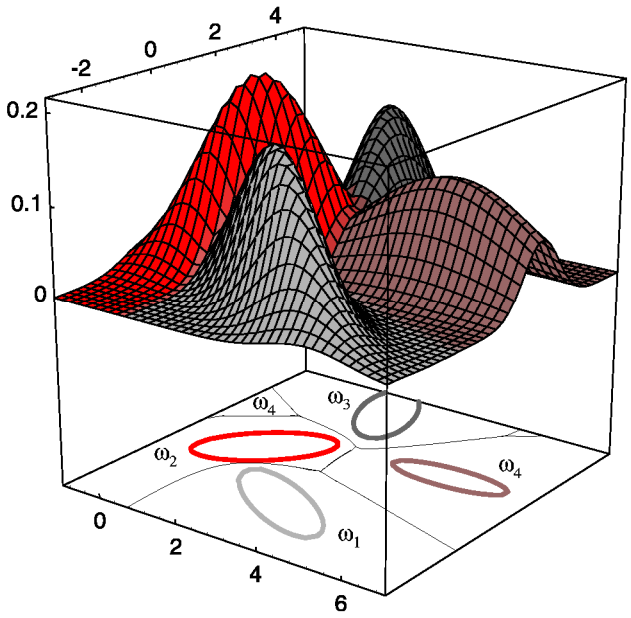


Рис. 2.5. Приклад границь прийняття рішень, які мають складну форму, для 4 нормальних розподілів густин ймовірностей ознак

При проектуванні системи розпізнавання після визначення набору і виду вирішальних функцій основним завданням є визначення їх коефіцієнтів. Для цього, як правило, використовується деяка навчальна вибірка об'єктів.

Навчання з вчителем: Функції відстані

Показником подібності образів є ступінь близькості точок, що описують ці образи в Евклідовому просторі. На етапі розпізнавання використовується критерій мінімуму відстані між точкою об'єкта, який розпізнається, і кластером класу, до якого цей об'єкт повинен бути віднесений. Методика є ефективною тільки за умови наявності кластеризаційних властивостей у розглянутих класів об'єктів, тобто коли відстань між об'єктами всередині класів є істотно меншою за відстань між групами точок, що утворюють класи. Класифікатор, що діє за принципом мінімальної відстані, є окремим випадком лінійного класифікатора.

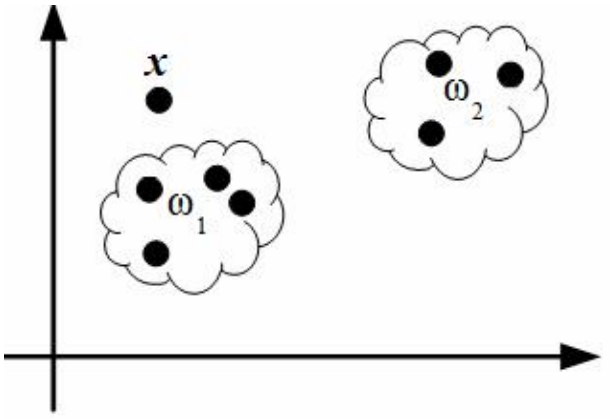


Рис. 2.6. Віднесення об'єкта до класу на основі відстані

У деяких випадках об'єкти кожного класу групуються навколо деякого одиничного образу, що є типовим або репрезентативним для цього класу. У цьому випадку говорять, що клас представляється своїм єдиним **еталоном**. У цій ситуації ефективним виявляється застосування критерію мінімальної відстані між об'єктом x, який розпізнається, і кожним з еталонів наявних класів. Відстань між образом x і одним з еталонів z_i в Евклідовому просторі визначається як

$$D = \left\| \mathbf{x} - \mathbf{z}_i \right\| = \sqrt{(\mathbf{x} - \mathbf{z}_i)^T (\mathbf{x} - \mathbf{z}_i)}\,.$$

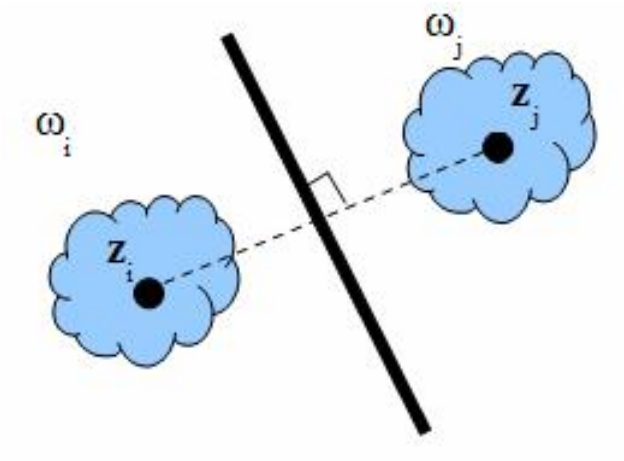


Рис.2.7. Поділ класів, що задаються одним еталоном

Іноді клас ω_i може характеризуватися не єдиним еталоном z_i, а деякою їх групою. В цьому випадку відстань від образу x, що розпізнається, до даного класу можна задати як

$$D = \min \left\| \mathbf{x} - \mathbf{z}_i^{\,K} \right\|, \quad K = 1, 2, \ldots N_i$$

тобто використовувати найменшу відстань від образу до одного з еталонів.

Метод еталонів є найбільш очевидним шляхом розпізнавання об'єктів, образи яких є точками в деякому просторі. Розглянемо випадок, коли образи будь-якого з існуючих класів проявляють тенденцію до тісного групування навколо деякого образу, який є типовим або репрезентативним для відповідного класу – еталонного образу. Це може бути образ, який знаходиться в центрі групи точок, яка відповідає певному класу в просторі.

Якщо існує M класів $\Omega_1, \Omega_2, \ldots, \Omega_M$, кожен з них може бути представлений за допомогою еталонних образів:

$$z^{e1} = \left[z_1^{e1}, z_2^{e1}, \ldots, z_N^{e1} \right],\ z^{e2} = \left[z_1^{e2}, z_2^{e2}, \ldots, z_N^{e2} \right], \ldots, z^{eM} = \left[z_1^{eM}, z_2^{eM}, \ldots, z_l^{eM} \right].$$

Коли класифікатору пред'являється новий образ z^0, то для нього розраховуються величини відстаней до кожного з еталонів за обраною мірою відстані d:

$$m_i = d\left(z^0, z^{ei}\right), \quad i = \overline{1, M},$$

та визначається мінімальна відстань

$$m_j^* = \min_{i=\overline{1,M}}\left(m_i\right).$$

Новий образ буде віднесено до того класу, який представляється найближчим до образу еталоном:

$$z^0 \in \Omega_j.$$

Навчання з вчителем: Правило найближчого сусіда

Область застосування підходу, заснованого на використанні критерію мінімальної відстані, не вичерпується випадками завдання класів за допомогою еталонів. Також до детермінованих відносяться класифікатори, побудовані на критерії одного найближчого сусіда (1-nearest neighbour, 1-NN). В цьому випадку новий образ вважається належним до того класу, до якого належить найближчий до нього образ. У більш складних випадках може використовуватися критерій кількох найближчих сусідів (*k*-nearest neighbour, *k*-NN). В цьому випадку образ, що підлягає розпізнаванню, відноситься до того класу, до якого відноситься більшість з k найближчих до нього образів.

Для того, щоб визначити кластер в просторі ознак, треба насамперед ввести поняття схожості (подібності), яке можна покласти в основу правила віднесення образів до області, яка характеризується деяким центром кластера. Одним з основних питань в детермінованих методах розпізнавання є питання побудови функції відстані між образами, тобто, введення метрики в просторі ознак.

Розглянемо випадок, коли задана вибірка об'єктів $\{S_1, S_2, ..., S_n\}$, для кожного з яких відома приналежність одного з класів $\omega_1, \omega_2, ..., \omega_M$. Визначимо правило класифікації на основі принципу найближчого сусіда: об'єкт, який розпізнається, слід віднести до того класу, до якого належить його найближчий сусід з вибірки $\{Si\}$.

Найближчим сусідом образу **x** називається такий образ s_i, для якого виконується

$$\mathbf{x} \in w_i, \text{ якщо } D(s_i, \mathbf{x}) = \min_k \{D(s_k, \mathbf{x})\}, \quad k = 1, 2, ...N_i$$

де $D(s_i, \mathbf{x})$ – відстань між вектором x і кожним із зразків заданої вибірки.

В даному випадку враховується приналежність одного з класів тільки одного найближчого сусіда, внаслідок чого це правило можна назвати 1-НС-правилом. Більш загальний випадок – q-НС-правило. В цьому випадку для класифікації образу слід визначити q його найближчих сусідів і віднести його до того класу, до якого відноситься найбільше число образів з отриманої групи.

При розробці функцій відстані у просторі ознак необхідно враховувати можливості перекриття класів та близькості кластерів, що додатково ускладнює вирішення цієї задачі. Різноманітність наявних мір схожості сигналу та еталону вимагає підлаштування міри під конкретну задачу розпізнавання, що необхідно виконувати, використовуючи властивості об'єктів, які підлягають класифікації.

Приклади індексів ефективності процедури класифікації:

- середнє значення квадратів відстаней між зразками в області кластера;
- внутрішньокластерна дисперсія;
- середнє значення квадратів відстаней між зразками в різних кластерних областях;
- міжкластерна відстань;
- коваріаційні матриці;
- матриці розкиду.

Найменша відстань між ознаками всередині класу і, в той же час, найбільша відстань між ознаками різних класів – принцип, що лежить в основі класифікації образів:

$$K = \frac{S_e}{S_{ik}},$$

де S_e – відстань між центрами кластерів; S_{ik} – відстань між образами

всередині *k*-го кластера.

Приклад алгоритму кластеризації: **алгоритм *K*-середніх**, який мінімізує показник якості, який визначається як сума квадратів відстаней від усіх точок, що входять в кластерну область, до центру кластера.

Імовірнісні моделі: Класифікатор Байєса

При розпізнаванні за критерієм Байєса стратегія прийняття рішень про віднесення об'єкта до певного класу обирається таким чином, щоб мінімізувати середній ризик помилки розпізнавання.

Відповідно до формули Байєса

$$P(C_i / \mathbf{x}) = \frac{P(C_i)\, p(\mathbf{x} / C_i)}{p(\mathbf{x})},$$

$P(C_i)$ – апріорна (умовна) ймовірність появи класу C_i, i = 1, 2, ..., *M;*

$P(C_i / \mathbf{x})$ – апостериорная вероятность того, что **x** относится к C_i;

$R_j(\mathbf{x}) = \sum_{i=1}^{M} L_{ij} P(C_i / \mathbf{x})$ – середній умовний ризик або втрати;

$p(\mathbf{x} / C_i)$ – функція правдоподібності класу C_i (умовна ФПВ від **x**)

Тоді

$$R_j(\mathbf{x}) = \frac{1}{p(\mathbf{x})} \sum_{i=1}^{M} L_{ij}\, p(\mathbf{x} / C_i) P(C_i)$$

Для 2-х класів класифікатор Байеса для нуль-одиничної функції втрат має дискримінантні функції виду

$$d_i(\mathbf{x}) = p(\mathbf{x} \mid \mathrm{C}_i) P(\mathrm{C}_i), \quad i = 1,2$$

Статистичне розпізнавання образів – підхід до розпізнавання, в якому класифікація виконується виходячи із статистичних характеристик класів та

базується на імовірнісних характеристиках розподілу ознак образів. Як правило, образ розглядається як випадковий вектор**.**

Для описання класів необхідно визначити імовірнісні характеристики розподілу об'єктів по класам Ω_i: функції густини імовірності $f_i(x_1, x_2, ...x_N)$ значень параметрів $x_1, x_2, ...x_N$ за умови належності об'єкта до класу Ω_i, та апріорні імовірності $P(\Omega_i)$ того, що об'єкт, який випадково обрали, належатиме класу Ω_i. Основним статистичним підходом до проблеми класифікації образів є використання так званих Баєсовських класифікаторів. Загальний підхід при цьому – об'єкт відноситься до того класу, для якого імовірність помилки класифікації є найменшою.

Можна показати, що при розпізнаванні за критерієм Байєса стратегія прийняття рішень щодо віднесення об'єкту до певного класу обирається таким чином, щоб мінімізувати середній ризик помилки розпізнавання.

Якщо апріорні ймовірності появи образів відповідних класів невідомі, або їх не можна оцінити, використовують мінімаксний критерій класифікації. За цим критерієм рішення про приналежність об'єкта до певного класу виноситься на основі байєсовської стратегії, яка відповідає такому значенню $P(\Omega_i)$, при якому середній ризик максимальний. Мінімаксна стратегія забезпечує те, що при будь-якому виборі середні втрати не будуть перевищувати максимального значення середніх втрат за байєсовською стратегією.

Якщо невідомими є апріорні ймовірності появи образів та значення втрат від неправильної класифікації, то використовується критерій Неймана-Пірсона. В залежності від наявної інформації про статистичні характеристики класів, використовуються також кореляційні класифікатори, класифікатори на основі оцінки максимальної правдоподібності, лінійний дискримінант Фішера, класифікація за критерієм найближчого сусіда.

Формально-лінгвістичні методи розпізнавання

Відомо, що у багатьох випадках в задачах розпізнавання образів інформацію про те, які образи відносяться до даного класу, недоцільно або неможливо задавати у вигляді перерахунку всіх членів класу, представлення кількох членів класу або деякої репрезентативної вибірки. Існує можливість піти альтернативним шляхом – задати множину базисних елементів – підобразів або примітивів, з яких можна побудувати всі члени даного класу, і правила породження нових елементів. При появі на вході системи розпізнавання нового об'єкта, який потрібно класифікувати, необхідно визначити, чи може такий об'єкт бути побудований із заданих базисних елементів по заданим правилам. І віднести його до відповідного класу.

Формально-лінгвістичні методи розпізнавання образів розділяються на дві групи: структурне співставлення та синтаксичний аналіз (лінгвістичний метод). Розглянемо коротко особливості застосування цих методів класифікації для випадку, коли треба класифікувати сигнали.

При розпізнаванні методами структурного співставлення, примітивами для сигналу, як правило, є прямі лінії або прості криві. Поширеним методом розкладу на примітиви є кусково-лінійна апроксимація. При цьому обирається розбиття сигналу на сегменти та для кожного сегменту досліджуваного сигналу обирається максимальне допустиме відхилення апроксимованого сигналу за обраною нормою відхилення. Після цього проводиться апроксимація для отримання примітивів на кожному сегменті. Ці примітиви кодуються символами a_i з деякої множини T, які надалі використовуються для структурного співставлення сигналів.

Послідовність символів з множини T називають ланцюгом:

$$s = a_1 a_2 ... a_m,\ a_i \in T,\ m - ціле.$$

В разі, коли для розрізнення образів недостатньо лише ланцюга, для представлення використовують ланцюги з векторами атрибутів, який містить додаткові параметри, що характеризують примітиви.

При використанні лінгвістичних методів синтаксичного аналізу для класів об’єктів вводять поняття граматик, алфавіту та речень із лінгвістики та користуються граматичним виводом (grammatical inference). Теорія формальних мов (автор – Ноам Хомський) дозволяє застосувати структурні правила до ланцюгів примітивів образів аналогічно до застосування правил лінгвістики у природних мовах. Формальною граматикою називають квадруполь

$$G = (T, N, P, S),$$

де T – множина термінальних символів – примітивів (алфавіт образів);

N – множина, до якої входять нетермінальні символи, що позначають клас елементів T. Разом множини T і N формують словник формальної мови;

P – множина синтаксичних правил (правил утворення ланцюгів);

S – спеціальний символ класу-попередника (початковий символ), який використовується для генерування будь-якого ланцюга з використанням правил із множини P.

Накладанням додаткових умов на правила утворення з множини P, можна створювати граматики різних типів: довільні, контекстні, безконтекстні, регулярні, стохастичні, атрибутивні. Перед початком класифікації необхідно вирішити задачу синтезу граматик та граматичних описів, а після цього – вирішити задачі побудови алгоритмів граматичного розбору образів для ідентифікації.

ТЕМА №3
ОСНОВИ НЕЙРОМЕРЕЖЕВИХ ТЕХНОЛОГІЙ. АРХІТЕКТУРИ НЕЙРОННИХ МЕРЕЖ

Термін «нейронні мережі» сформувався у середині 50-х років XX століття. Основні результати в цій області пов'язані з іменами У. Маккалоха, Д. Хебба, Ф. Розенблатта, М. Мінського, Дж. Хопфілда. Штучні нейронні мережі будуються за принципами організації та функціонування їх біологічних аналогів. Вони здатні вирішувати широке коло завдань: розпізнавання образів, ідентифікація, прогнозування, оптимізація, управління складними об'єктами. Подальше підвищення продуктивності комп'ютерів пов'язують з нейронними мережами, зокрема, з нейрокомп'ютером, основу яких складає штучна нейронна мережа.

Модель нейрона є ключовим елементом нейронної мережі і від її вибору багато в чому залежатимуть не тільки архітектура мережі і алгоритм її навчання але й можливості мережі в цілому, її швидкодія і точність отриманих за її допомогою результатів.

Біологічний нейрон

Всі процеси обміну інформацією (передачі подразнень від шкіри, вух і очей до мозку, процеси мислення та управління діями) реалізовано в живому організмі як передача електричних імпульсів між нейронами.

Нейрон (нервова клітина) є особливою біологічною клітиною, яка обробляє інформацію (рис. 6.1). Він складається з тіла (cell body), або соми (soma), і відростків нервових волокон двох типів ─ дендритів (dendrites), за якими приймаються імпульси, і єдиного аксона (axon), за яким нейрон може передавати імпульс. Тіло нейрона має ядро (nucleus), яке містить інформацію про спадкові властивості, і плазму, що володіє молекулярними засобами для виробництва необхідних нейрону матеріалів.

Нейрон отримує сигнали (імпульси) від аксонів інших нейронів за допомогою дендритів (приймачів) і передає сигнали, згенеровані тілом клітки, вздовж свого аксона (передавача), що наприкінці розгалужується на волокна (strands). На закінченнях цих волокон знаходяться спеціальні утворення ─ синапси (synapses), які впливають на величину імпульсів.

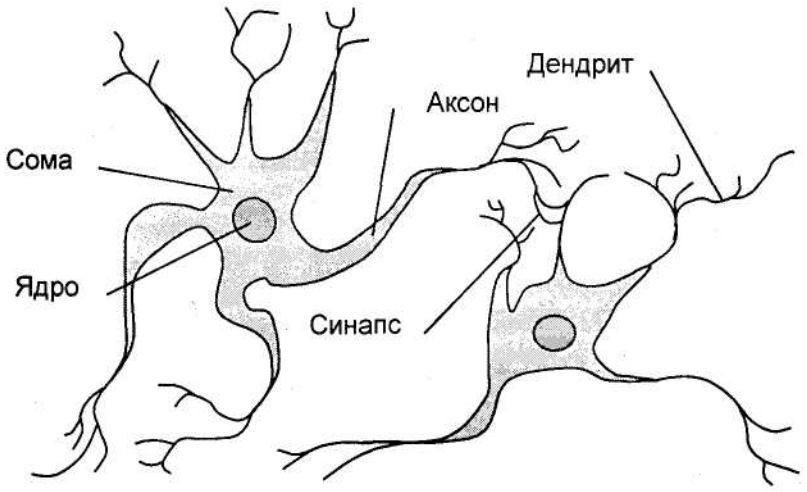


Рис. 3.1. Взаємодія біологічних нейронів

Синапс є елементарною структурою і функціональним вузлом між двома нейронами (волокно аксона одного нейрона і дендрит іншого). Коли імпульс досягає синоптичного закінчення, вивільняються хімічні речовини нейротрансмітери. Нейротрансмітери дифундують крізь синоптичну щілину, збуджуючи або гальмуючи, залежно від типу синапсу, здатність нейрона-приймача генерувати електричні імпульси. Результативність передачі імпульсу синапсом може налаштовуватися проходять крізь нього сигналами так, що синапси можуть навчатися в залежності від активності процесів, у яких вони приймають участь. Ця залежність від передісторії діє як пам'ять, яка, можливо, відповідальна за пам'ять людини. Важливо відзначити, що ваги синапсів можуть змінюватися з часом, а значить, змінюється і поведінка відповідних нейронів.

Кора головного мозку людини містить близько 10^{11} нейронів і являє собою протяжну поверхню товщиною від 2 до 3 мм з площею близько 2200 $см^2$.

Кожен нейрон пов'язаний з 10^3-10^4 іншими нейронами. В цілому мозок людини містить приблизно від 10^{14} до 10^{15} взаємозв'язків.

Нейрони взаємодіють короткими серіями імпульсів тривалістю, як правило, кілька мілісекунд. Повідомлення передається за допомогою частотно-імпульсної модуляції. Частота може змінюватися від декількох одиниць до сотень герц, що в мільйон разів повільніше, ніж робота швидкодіючих електронних схем з перемиканням. Тим не менш складні задачі розпізнавання людина вирішує за кілька сотень мілісекунд. Ці рішення контролюються мережею нейронів, які мають швидкість виконання операцій всього кілька мілісекунд. Це означає, що обчислення вимагають не більше 100 послідовних стадій. Іншими словами, для таких складних завдань мозок «запускає» паралельні програми, що містять близько 100 кроків. Міркуючи аналогічним чином, можна виявити, що кількість інформації, що посилається від одного нейрона іншому, має бути дуже малим (кілька біт). Звідси випливає, що основна інформація не передається безпосередньо, а захоплюється і розподіляється у зв'язках між нейронами.

Персептрон

Простий персептрон ─ це модель нейрона Маккаллока-Пітса з відповідною стратегією навчання.

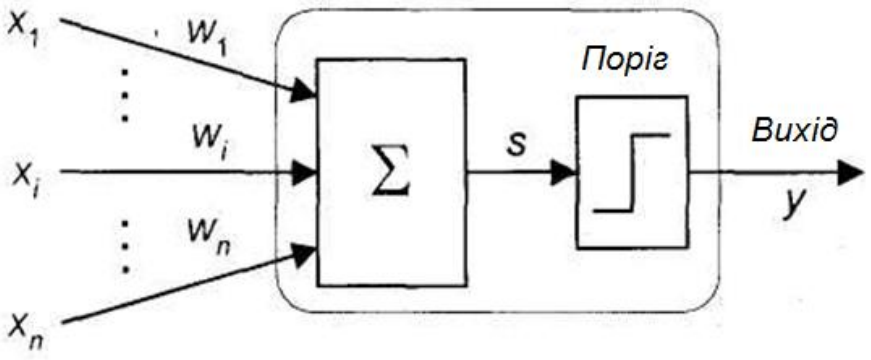


Рис. 3.2. Модель персептрона

Вагові коефіцієнти входів суматора, на які надходять вхідні сигнали x_j, позначаються w_{ij}, а порогове значення, яке надходить з так званого поляризатора, - w_{i0}. Нелінійна функція активації персептрона являє собою дискретну функцію сходинчастого типу, внаслідок чого вихідний сигнал нейрона може приймати тільки два значення ─ 0 або 1 відповідно до правила

$$y_i(u_i) = \begin{cases} 1, \to u \geq 0 \\ 0, \to u < 0 \end{cases}, \tag{3.1}$$

де u_i ─ вихідний сигнал суматора

$$u_i = \sum_{j=0}^{N} w_{ij} x_j \ . \tag{3.2}$$

У наведеній формулі мається на увазі, що вектор x, який має довжину N, доповнений нульовим членом x_0=1, що формує сигнал поляризації:

$$x = [x_0, x_1, ..., x_N].$$

Навчання персептрона вимагає наявності вчителя і полягає в такому підборі вагів w_{ij}, щоб вихідний сигнал y_i був найбільш близький до заданого значення d_i. Це навчання гетероасоціативної типу, при якому кожній навчальній вибірці, що подається вектором x, апріорі поставлено у відповідність очікуване значення d_i на виході i-го нейрона.

По завершенні уточнення вагових коефіцієнтів представляються черговий навчальний вектор x і пов'язане з ним очікуване значення d, і значення вагів уточнюються заново. Цей процес багаторазово повторюється на всіх навчальних вибірках, поки не будуть мінімізовані відмінності між усіма значеннями y_i і відповідними їм очікуваними значеннями d_i.

Слід зазначити, що правило персептрона являє собою окремий випадок запропонованого набагато пізніше правила Відроу-Хоффа. Відповідно до цього

правила підбір вагових коефіцієнтів нейрона (необов'язково персептронного типу) проводиться за формулами:

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij}\,, \tag{3.3}$$

$$\Delta w_{ij} = x_j(d_i - y_i). \tag{3.4}$$

Аналогічні співвідношення використовуються при підборі ваги поляризатора w_{i0}, для якого вхідний сигнал завжди дорівнює 1, у зв'язку з чим

$$\Delta w_{i0} = x_j(d_i - y_i). \tag{3.5}$$

Легко помітити, що якщо сигнали y_i, і d_i приймають тільки двійкові значення 0 і 1, то правило Відроу-Хоффа перетворюється на правило персептрона. Мінімізація відмінностей між фактичними реакціями нейрона y_i, і очікуваними значеннями d_i може бути представлена як мінімізація конкретної функції похибки (цільової функції) E

$$E = \sum_{k=1}^{p} \left(y_i^{(k)} - d_i^{(k)} \right)^2 , \tag{3.6}$$

де p означає кількість запропонованих навчальних вибірок. Така мінімізація при використанні правила персептрона проводиться за методом безградієнтной оптимізації. Ефективність методу при великій кількості навчальних вибірок відносно невелика, а кількість циклів навчання і його тривалість зростають дуже швидко, причому без жодної гарантії досягнення мінімуму цільової функції. Усунути ці недоліки можна тільки у разі застосування неперервної функції активації, при якій цільова функція E також стає неперервною, що дає можливість використовувати в процесі навчання інформацію про величину градієнта.

Сигмоїдальний нейрон

Нейрон сигмоїдального типу (рис. 3.3) має структуру, подібну моделі Маккаллока-Пітса, з тією різницею, що функція активації є неперервною і може бути виражена у вигляді сигмоїдальної уніполярної або біполярної функції. Уніполярна функція, як правило, представляється формулою

$$f(x) = \frac{1}{1 + e^{-\beta x}}, \tag{3.7}$$

тоді як біполярна функція має вигляд

$$f(x) = \tanh(\beta x). \tag{3.8}$$

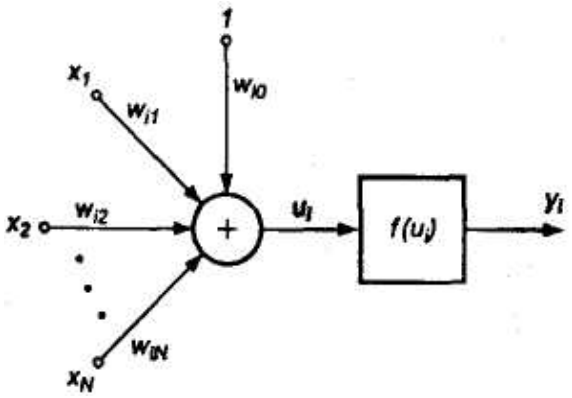


Рис. 3.3. Модель сигмоїдального нейрона

У цих формулах параметр β підбирається користувачем. Його значення впливає на форму функції активації. На рис. 3.4 представлені графіки сигмоїдальнойї функції від змінної x для різних значень β, причому на рис. 3.4а показана уніполярна, а на рис. 3.4б ─ біполярна функція.

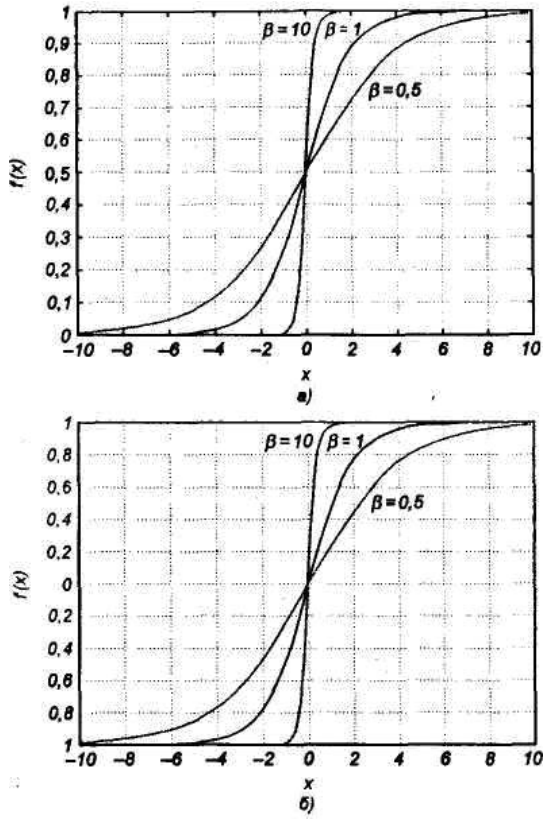


Рис. 3.4. Графік сигмоїдальної функції:
а) уніполярної; б) біполярної при різних значеннях коефіцієнта β

Важливою властивістю сигмоїдальной функції є її диференціюємість. Для уніполярної функції маємо

$$\frac{df(x)}{dx} = \beta f(x)(1-(x)), \tag{3.9}$$

тоді як для біполярної функції

$$\frac{df(x)}{dx} = \beta\left(1 - f^2(x)\right). \tag{3.10}$$

І в першому, і в другому випадку графік зміни похідної щодо змінної x має колоколоподібну форму, а його максимум відповідає значенню $x = 0$ (рис. 3.5).

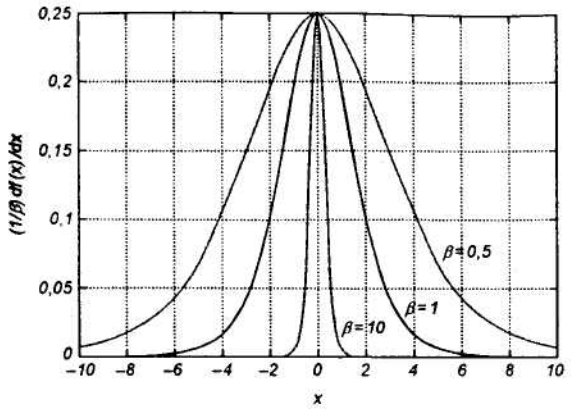


Рис. 3.5. Графік похідної від сигмоїдальної функції при різних значеннях коефіцієнта β

Сигмоїдальний нейрон, як правило, навчається з учителем за принципом мінімізації цільової функції, яка для *i*-го нейрона визначається у вигляді

$$E = \frac{1}{2}(y_i - d_i)^2, \tag{3.11}$$

де

$$y_i = f(u_i) = f\left(\sum_{j=0}^{N} w_{ij} x_j\right). \tag{3.12}$$

Функція $f(u_i)$ є сигмоїдальною, x — це вхідний вектор зі значенням $x_0 = 1$ при наявності поляризації і $x_0=0$ при її відсутності, d_i — відповідне йому очікуване значення на виході *i*-го нейрона. Застосування неперервної функції активації дозволяє використовувати при навчанні градієнтні методи. Найпростіше реалізувати метод найшвидшого спуску, відповідно до якого уточнення вектора вагів $w = [w_{i0}, w_{i1}, ..., w_{iN}]^T$ проводиться в напрямку негативного градієнта цільової функції. Якщо ця функція визначена виразом (3.11), *j*-та складова градієнта має вигляд:

$$\nabla_j E = \frac{dE}{dw_{ij}} = e_i x_j \frac{df(u_i)}{du_i}, \tag{3.13}$$

де $e = (y_i - d_i)$ означає різницю між фактичним і очікуваним значенням вихідного сигналу нейрона. Якщо ввести позначення $\delta_i = e_i \frac{df(u_i)}{du_i}$, то можна отримати вираз, що визначає *j*-ту складову градієнта у вигляді

$$\nabla_j E = \delta_i x_j . \tag{3.14}$$

Значення вагових коефіцієнтів також можуть уточнюватися дискретним способом:

$$w_{ij}(t+1) = w_{ij}(t) - \eta \delta_i x_j , \tag{3.15}$$

де η ─ це коефіцієнт навчання, значення якого, як правило, вибирають емпірично з інтервалу (0,1).

Модель нейрона типу WTA

Нейрони типу WTA (англ.: англ.: Winner Takes All ─ Переможець отримує все) мають вхідний модуль у вигляді стандартного суматора, який розраховує суму вхідних сигналів з відповідними вагами w_{ij}. Вихідний сигнал *i*-го суматора визначається згідно з формулою

$$u_i = \sum_{j=0}^{N} w_{ij} x_j . \tag{3.16}$$

Група конкуруючих між собою нейронів (рис. 3.11) отримує одні й ті ж вхідні сигнали x_j. Залежно від фактичних значень вагових коефіцієнтів сумарні

сигнали u_i окремих нейронів можуть розрізнятися. За результатами порівняння цих сигналів переможцем визнається нейрон, значення u_i у якого виявилося найбільшим. Нейрон-переможець виробляє на своєму виході стан 1, а решта нейронів (переможені) переходять у стан 0.

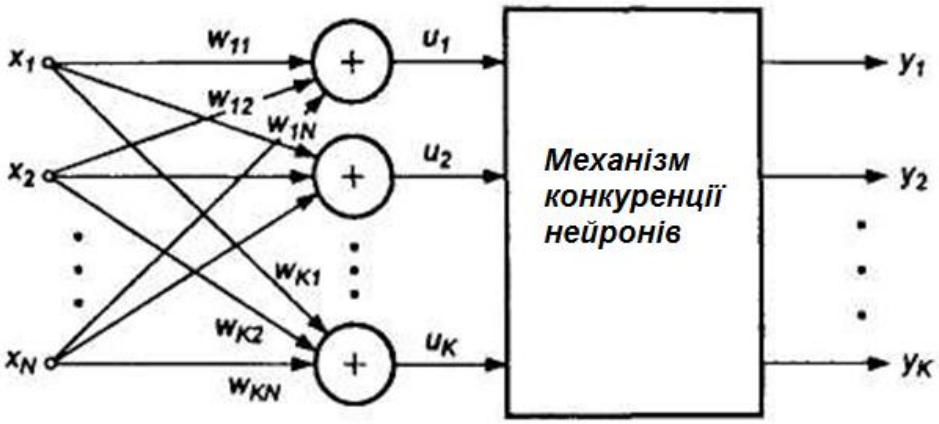


Рис. 3.11. Схема з'єднання нейронів типу WTA

Для навчання нейронів типу WTA не вимагається вчитель, воно протікає з використанням нормалізованих вхідних векторів x. На початковому етапі випадковим чином генеруються вагові коефіцієнти кожного нейрона, нормалізовані щодо 1. Після подання першого вхідного вектора x визначається переможець етапу. Нейрон, який переміг в цьому змаганні, переходить у стан 1, що дозволяє йому провести уточнення вагів його вхідних ліній w_{ij} (за правилом Гросберга).

Переможені нейрони формують на своїх виходах стан 0, що блокує процес уточнення їх вагових коефіцієнтів. Внаслідок бінарності значень вихідних сигналів конкуруючих нейронів (0 або 1) правило Гросберга може бути дещо спрощено:

$$w_{ij}(t+1) = w_{ij}(t) + \eta\left[x_j - w_{ij}(t)\right]. \qquad (3.17)$$

На функціонування нейронів типу WTA істотно впливає нормалізація вхідних векторів і вагових коефіцієнтів. Вихідний сигнал u_i i-го нейрона може бути поданий як

$$u_i = w^T x = \|w\|\|x\|\cos\varphi_i . \qquad (3.18)$$

Оскільки $\|w\| = \|x\| = 1$, значення u_i, визначається кутом між векторами x і w, $u_i = \cos\varphi_i$. Тому переможцем виявляється нейрон, вектор вагів якого виявляється найбільш близьким до поточного навчального вектора x. У результаті перемоги нейрона уточнюються його вагові коефіцієнти, значення яких наближаються до значень поточного навчального вектора x. Якщо на вхід мережі буде подаватися безліч близьких за значеннями векторів, перемагати буде один і той же нейрон. Тому його ваги стануть рівними усередненим значенням тих вхідних векторів, завдяки яким даний нейрон виявився переможцем. Переможені нейрони не змінюють своїх вагів. Тільки перемога при черговому поданні вхідного вектора дозволить їм провести уточнення вагових коефіцієнтів і продовжити процес навчання у разі ще однієї перемоги.

Наслідком такої конкуренції стає самоорганізація процесу навчання. Нейрони уточнюють свої ваги таким чином, що при пред'явленні групи близьких за значеннями вхідних векторів переможцем завжди виявляється один і той же нейрон. У процесі функціонування саме цей нейрон завдяки суперництву розпізнає свою категорію вхідних даних. Системи такого типу найчастіше застосовуються для класифікації векторів.

Архітектури нейронних мереж

Нейронна мережа являє собою сукупність нейроподібних елементів, певним чином з'єднаних один з одним і з зовнішнім середовищем за допомогою зв'язків, що визначаються ваговими коефіцієнтами.

Залежно від функцій, які виконуються нейронами в мережі, можна виділити три їх типи:

• *вхідні нейрони*, на які подається вектор, що кодує вхідний вплив або образ зовнішнього середовища; в них зазвичай не здійснюється обчислювальних процедур, а інформація передається з входу на вихід шляхом зміни їх активації;

• *вихідні нейрони*, вихідні значення яких представляють виходи нейронної мережі;

• *проміжні нейрони*, що становлять основу нейронних мереж.

Конкретний вид виконуваного мережею перетворення даних обумовлюється не тільки характеристиками нейроподібних елементів, але й особливостями її архітектури, а саме топологією міжнейронних зв'язків, вибором певних підмножин нейроподібних елементів для введення і виведення інформації, способами навчання мережі, наявністю або відсутністю конкуренції між нейронами, напрямком і способами управління і синхронізації передачі інформації між нейронами.

З погляду топології можна виділити три основні типи нейронних мереж:

• повнозв'язні (рис. 3.12, а);

• багатошарові або шаруваті (рис. 3.12, б);

• слабозв'язаних (з локальними зв'язками) (рис. 3.12, в).

У *повнозв'язних нейронних мережах* кожен нейрон передає свій вихідний сигнал іншим нейронам, у тому числі й самому собі. Всі вхідні сигнали подаються всім нейронам. Вихідними сигналами мережі можуть бути всі або деякі вихідні сигнали нейронів після кількох тактів функціонування мережі.

У *багатошарових нейронних мережах* нейрони об'єднуються в шари. Шар містить сукупність нейронів з єдиними вхідними сигналами. Кількість нейронів у шарі може бути будь-якою і не залежить від кількості нейронів в інших шарах. У загальному випадку мережа складається з Q шарів, пронумерованих зліва направо. Зовнішні вхідні сигнали подаються на входи нейронів вхідного шару (його часто нумерують як нульовий), а виходами мережі є вихідні сигнали останнього шару. Крім вхідного і вихідного шарів в багатошаровій нейронної мережі є один або кілька прихованих шарів. Зв'язки від виходів нейронів

деякого шару q до входів нейронів наступного шару (q+1) називаються послідовними.

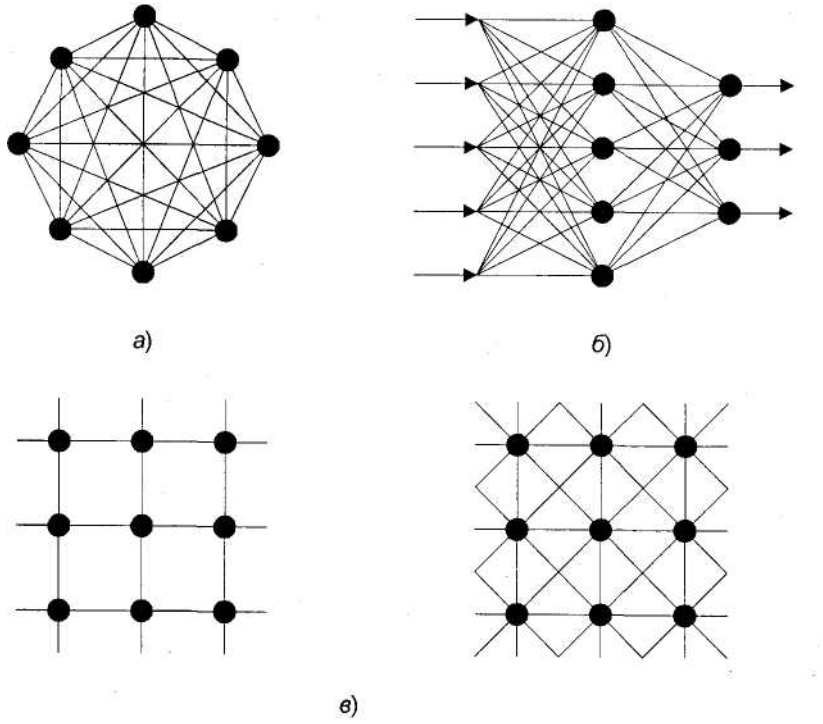


Рис. 3.12. Архітектури нейронних мереж: а) повнозв'язна мережа; б) багатошарова мережа з послідовними зв'язками; в) слабозв'язані мережі

У свою чергу, серед багатошарових нейронних мереж виділяють наступні типи:

1) *Монотонні мережі* — це окремий випадок шаруватих мереж з додатковими умовами, накладеними на зв'язки і нейрони. Кожен шар крім останнього (вихідного) розбитий на два блоки: збудливий і гальмуючий. Зв'язки між блоками теж поділяються на гальмуючі і збуджуючі. Якщо від нейронів блоку *A* до нейронів блоку *B* ведуть тільки збуджуючі зв'язки, то це означає, що будь-який вихідний сигнал блоку є монотонною неспадаючою функцією будь-якого вихідного сигналу блоку *A*. Якщо ж ці зв'язки є тільки гальмучими, то будь-який вихідний сигнал блоку *B* є незростаючою функцією будь-якого вихідного сигналу блоку *A*. Для нейронів монотонних мереж необхідна монотонна залежність вихідного сигналу нейрона від параметрів вхідних сигналів.

2) *Мережі без зворотних зв'язків*. У таких мережах нейрони вхідного шару отримують вхідні сигнали, перетворюють їх і передають нейронам першого прихованого шару, і так далі аж до вихідного, який видає сигнали для інтерпретатора і користувача. Якщо не зазначено протилежне, то кожен вихідний сигнал q-го шару подасться на вхід всіх нейронів $(q + 1)$-го шару; проте можливим є варіант з'єднання q-го шару з довільним $(q + p)$-м шаром.

Серед багатошарових мереж без зворотних зв'язків розрізняють *повнозв'язані* (вихід кожного нейрона q-го шару пов'язаний з входом кожного нейрона $(q + 1)$-го шару) і *частково повнозв'язані*. Класичним варіантом шаруватих мереж є повнозв'язані мережі прямого поширення (рис. 3.13).

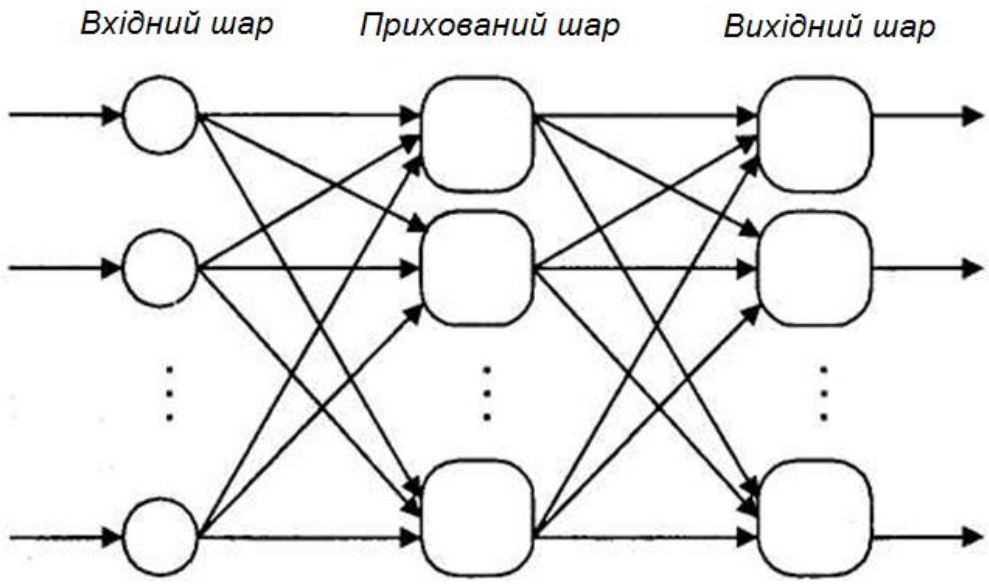


Рис. 3.13. Багатошарова (двошарова) мережа прямого поширення

3) *Мережі зі зворотними зв'язками*. У мережах зі зворотними зв'язками інформація з наступних шарів передається на попередні.

Як приклад мереж зі зворотними зв'язками на рис. 3.14 представлені частково-рекурентні мережі Елмана і Жордана.

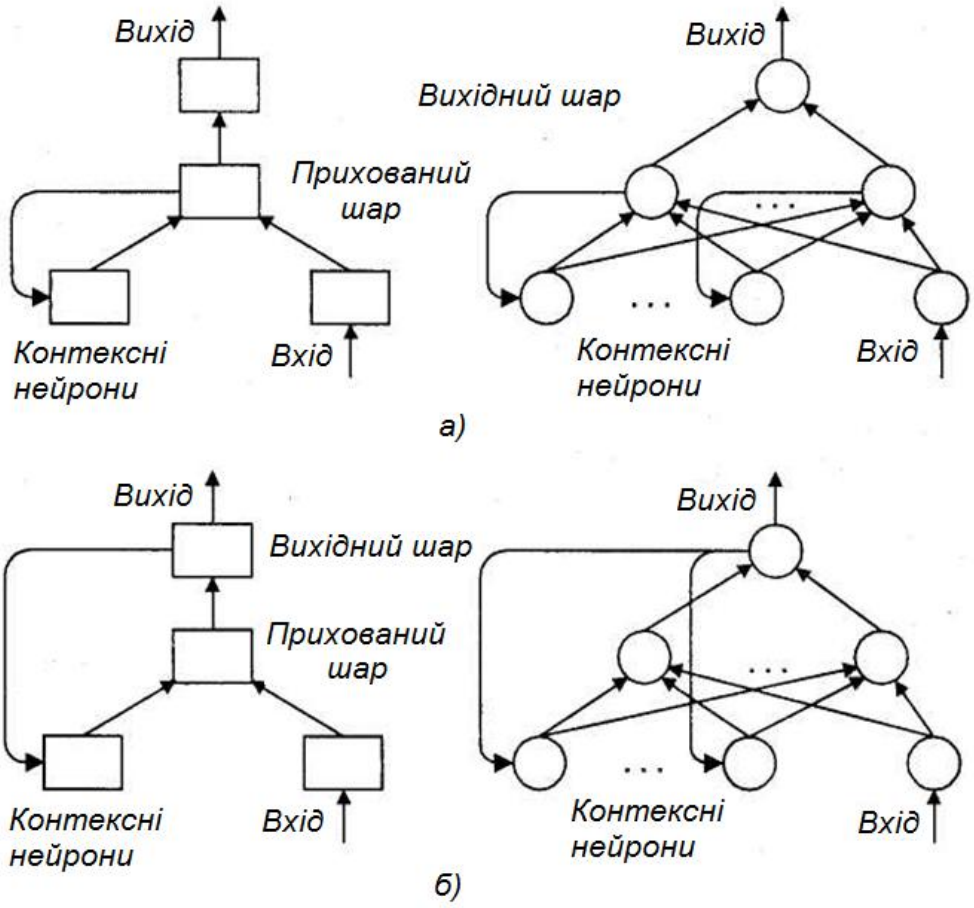


Рис. 3.14. Частково-рекурентні мережі: а) Елмана, б) Жордана

У слабозв'язаних нейронних мережах нейрони розташовуються у вузлах прямокутної або гексагональної решітки. Кожен нейрон зв'язаний з чотирма (околиця фон Неймана), шістьма (околиця Голея) або вісьмома (околиця Мура) своїми найближчими сусідами.

Відомі нейронні мережі можна розділити за типами структур нейронів на гомогенні (однорідні) і гетерогенні. Гомогенні мережі складаються з нейронів одного типу з єдиної функцією активації, а в гетерогенну мережу входять нейрони з різними функціями активації.

Існують бінарні і аналогові мережі. Перші з них оперують тільки двійковими сигналами, і вихід кожного нейрона може приймати значення логічного нуля (загальмований стан) або логічної одиниці (збуджений стан).

Ще одна класифікація ділить нейронні мережі на синхронні і асинхронні. У першому випадку в кожен момент часу лише один нейрон змінює свій стан, у

другому ─ стан змінюється відразу у цілої групи нейронів, як правило, у всього шару. Алгоритмічно хід часу в нейронних мережах задається ітераційним виконанням однотипних дій над нейронами.

Мережі можна класифікувати також за кількістю шарів. Теоретично число прошарків і число нейронів у кожному шарі може бути довільним, однак фактично воно обмежене ресурсами комп'ютера або спеціалізованих мікросхем, на яких зазвичай реалізується нейронна мережа. Чим складніше мережа, тим більш складні завдання вона може вирішувати.

Вибір структури нейронної мережі здійснюється відповідно до особливостей і складності завдання. Для вирішення окремих типів завдань вже існують оптимальні конфігурації. Якщо ж завдання не може бути зведене до жодного з відомих типів, доводиться вирішувати складну проблему синтезу нової конфігурації. При цьому необхідно керуватися такими основними правилами:

• можливості мережі зростають зі збільшенням кількості нейронів мережі, щільності зв'язків між ними і кількості шарів;

• запровадження зворотних зв'язків поряд зі збільшенням можливостей мережі піднімає питання про динамічну стійкість мережі;

• складність алгоритмів функціонування мережі, введення декількох типів синапсів сприяє посиленню потужності нейронної мережі.

Питання необхідних і достатніх властивостей мережі для вирішення завдань того чи іншого роду являє собою цілий напрям нейрокомп’ютерної науки. Так як проблема синтезу нейронної мережі сильно залежить від розв'язуваної задачі, дати загальні докладні рекомендації важко. У більшості випадків оптимальний варіант отримують на основі інтуїтивного підбору, хоча в літературі наведено докази того, що для будь-якого алгоритму існує нейронна мережа, яка може його реалізувати.

ТЕМА №4
МАШИННЕ НАВЧАННЯ ЗА ДОПОМОГОЮ МОВИ ПРОГРАМУВАННЯ PYTHON

Бібліотека **Scikit-learn** використовується для вирішення завдань класичного машинного навчання. Scikit-learn працює на основі декількох поширених математичних бібліотек і інтегрує їх одина з одною.

Для своєї роботи scikit-learn використовує такі бібліотеки:

NumPy: математичні операції

SciPy: науково-технічні обчислення

Matplotlib: візуалізація даних

SymPy: символьна математика

Pandas: обробка, маніпуляції і аналіз даних

Scikit-learn спеціалізується на алгоритмах машинного навчання для вирішення завдань навчання з учителем: **класифікації** (прогноз ознаки, множина допустимих значень якого обмежена) і **регресії** (прогноз ознаки з речовими значеннями), а також для задач навчання без учителя: **кластеризації** (розбиття даних по класах, які модель визначить сама), **зниження розмірності** (подання даних в просторі меншої розмірності з мінімальними втратами корисної інформації) і детектування аномалій.

Бібліотека Scikit-learn реалізує наступні основні методи:

Лінійні: моделі, завдання яких побудувати розділяє (для класифікації) або аппроксимирующую (для регресії) гіперплоскость.

Метричні: моделі, які обчислюють відстань по одній з метрик між об'єктами вибірки, і приймають рішення в залежності від цієї відстані (K найближчих сусідів).

Дерева рішень: навчання моделей, що базуються на безлічі умов, оптимально обраних для вирішення завдання.

Ансамблеві методи: методи, засновані на деревах рішень, які комбінують

міць множини дерев, і таким чином підвищують їх якість роботи, а також дозволяють проводити відбір ознак (бустінг, беггінг, випадковий ліс, мажоритарне голосування).

Нейронні мережі: комплексний нелінійний метод для задач регресії і класифікації.

SVM: нелінійний метод, який навчається визначати межі прийняття рішень.

Наївний Байєсівський метод: пряме ймовірнісне моделювання для задач класифікації.

PCA: лінійний метод зниження розмірності і відбору ознак

t-SNE: нелінійний метод зниження розмірності

K-середніх: найпоширеніший метод для кластеризації, який вимагає подати на вхід число кластерів, за якими повинні бути розподілені дані.

Крос-валідація: метод, при якому для навчання використовується весь набір даних (на відміну від розбиття на вибірки train/test), проте навчання відбувається багаторазово, і в якості валідаційної вибірки на кожному кроці виступають різні частини датасета. Підсумковий результат являє собою усереднення отриманих результатів.

Grid Search: метод для знаходження оптимальних гіперпараметрів моделі шляхом побудови сітки з значень гіперпараметрів і послідовного навчання моделей з усіма можливими комбінаціями гіперпараметрів з сітки.

Крім цього, Scikit-learn містить функції для розрахунку значень метрик, вибору моделей, препроцесінга даних та інші.

З метою написання коду в Python, інтегрування модулів і бібліотек, необхідно обрати для себе та підготувати інтегроване середовище розробки (наприклад, PyCharm, Komodo IDE, Spyder, PyScripter або інш.) (рис.4.1).

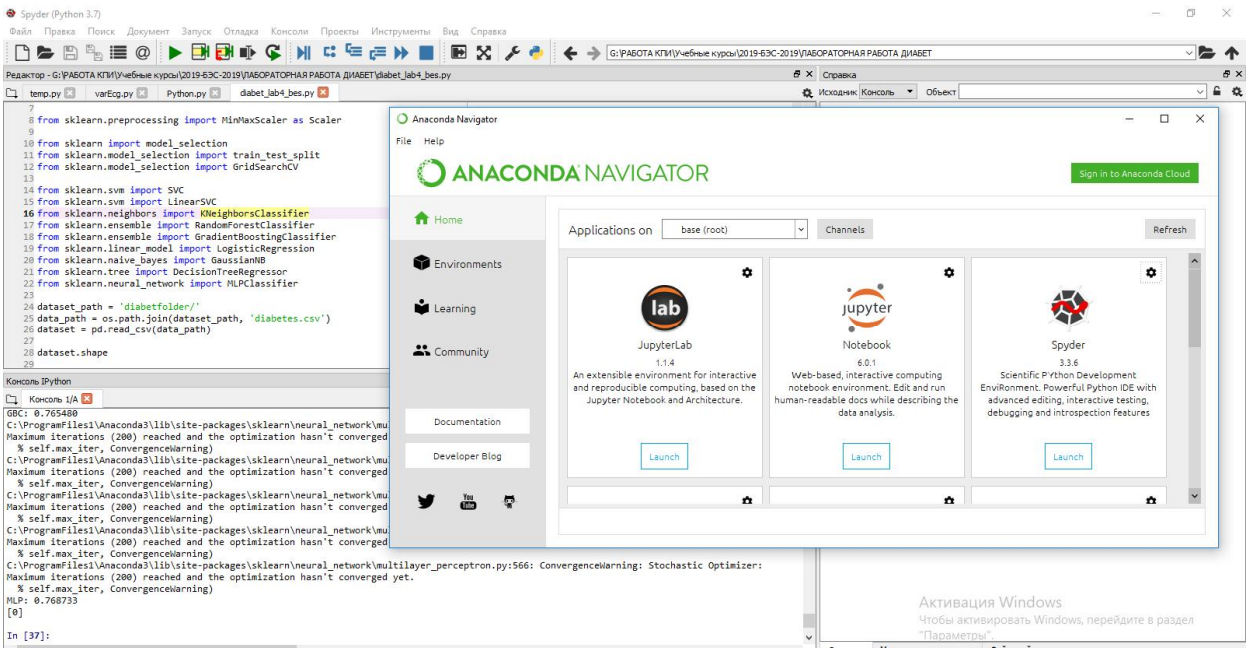


Рис.4.1. Дистрибутив Anaconda, використання Spyder

МЕТОДИ МАШИННОГО НАВЧАННЯ З ВЧИТЕЛЕМ

Метод k-найближчих сусідів

Алгоритм k найближчих сусідів, є одним з найпростіших алгоритмів машинного навчання. Побудова моделі полягає в запам'ятовуванні навчального набору даних. Для того, щоб зробити прогноз для нової точки даних, алгоритм знаходить найближчі до неї точки навчального набору, тобто знаходить «найближчих сусідів».

У найпростішому варіанті алгоритм k найближчих сусідів розглядає лише одного найближчого сусіда - точку навчального набору, найближче розташовану до точки, для якої ми хочемо отримати прогноз. Прогнозом є відповідь, вже відома для даної точки навчального набору (рис. 4.2).

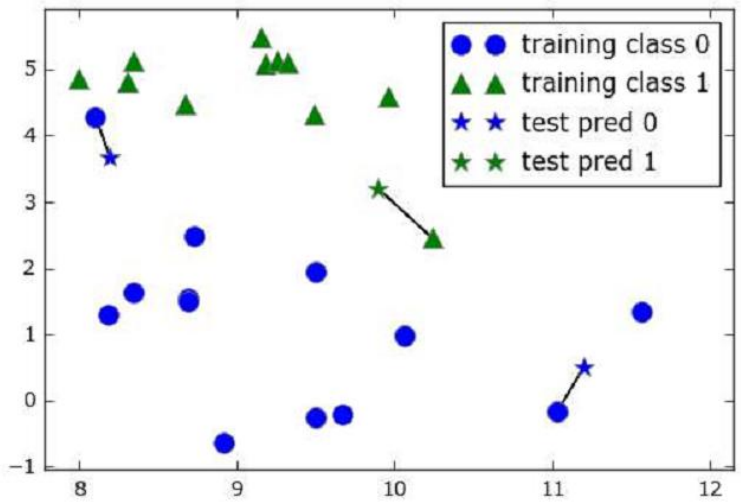


Рис. 4.2. Прогнози, отримані за допомогою моделі одного найближчого сусіда

Замість того, щоб враховувати лише одного найближчого сусіда, можна розглянути будь-яку кількість (k) сусідів. Звідси і походить назва алгоритму k найближчих сусідів. Коли приймають до уваги більше одного сусіда, для присвоєння мітки використовується голосування (voting). Це означає, що для кожної точки тестового набору підраховується кількість сусідів, що відносяться до класу 0, і кількість сусідів, що відносяться до класу 0. Потім точці тестового набору присвоюється клас, який найбільш часто зустрічається: іншими словами, обирається клас, який набрав більшість серед k найближчих сусідів. У прикладі, наведеному нижче (рис. 4.3), використовуються три найближчих сусіда. Хоча даний малюнок ілюструє завдання бінарної класифікації, цей метод можна застосувати до наборів даних з будь-якою кількістю класів. У разі мультикласової класифікації підраховується кількість сусідів, що належать до кожного класу, і прогнозується клас, який найбільш часто зустрічається.

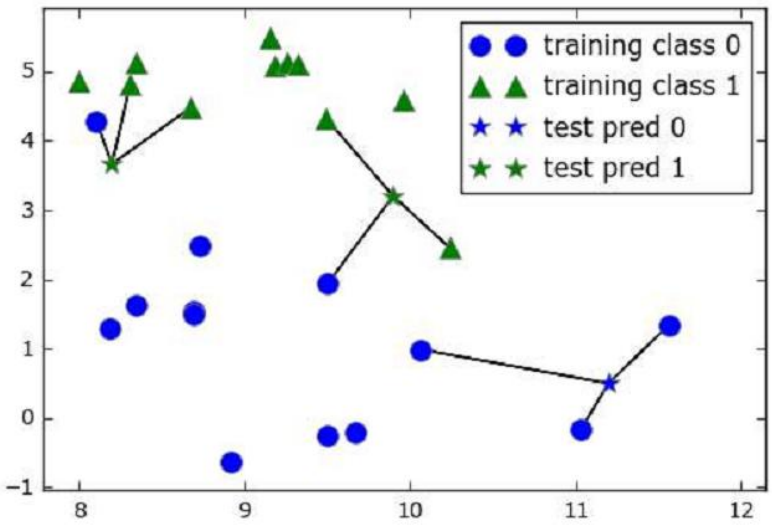


Рис.4.3. Прогнози, отримані за допомогою моделі 3х найближчих сусідів

Алгоритм k найближчих сусідів можна застосувати, використовуючи бібліотеку Python **scikit-learn**. По-перше, потрібно розділити дані на навчальний і тестовий набори, щоб оцінити узагальнюючу здатність моделі

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
```

Далі потрібно створити об'єкт-екземпляр класу, задаючи параметри, наприклад, кількість сусідів, що буде використовуватися для класифікації. В даному прикладі вона рівнює 3:

```
from sklearn.neighbors import KNeighborsClassifier
nnclsf = KNeighborsClassifier(n_neighbors=3)
```

Потім необхідно налаштувати класифікатор, використовуючи навчальний набір – для KNeighborsClassifier це означає запам'ятовування набору даних для того, щоб обчислити сусідів в ході прогнозування:

```
nnclsf. fit(X_train, y_train)
```

Щоб отримати прогнози для тестових даних, потрібно викликати метод **predict**. Для кожної точки тестового набору він обчислює її найближчих сусідів у навчальному наборі і знаходить серед них клас, який найбільш часто зустрічається.

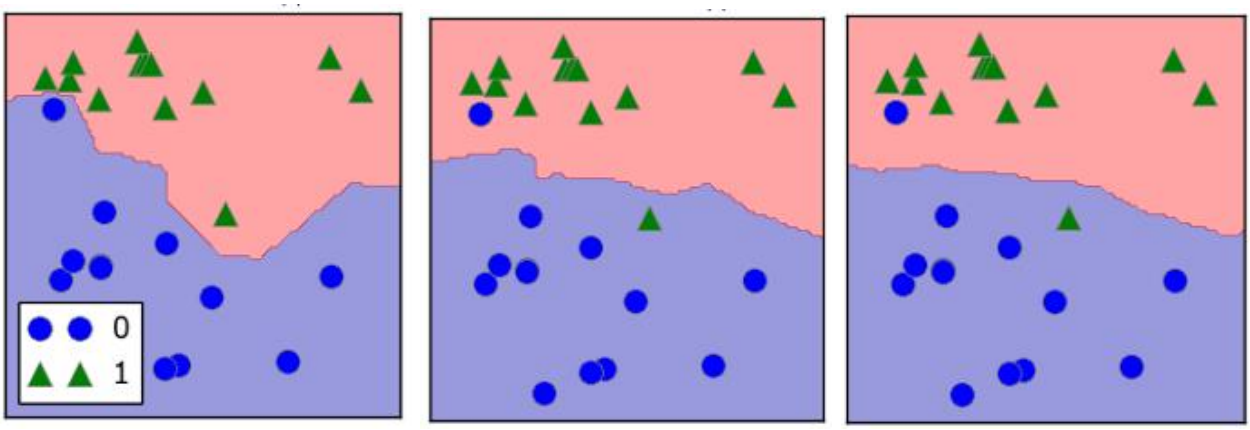
```
print("Прогнози на тестовій вибірці: {}".format(nnclsf. predict(X_test)))
```

Для оцінювання узагальнюючої здатності моделі викликається метод **score** з тестовими даними й тестовими мітками:

```
print("Вірність на тестовій вибірці: {}". format(nnclsf. score(X_test, y_test)))
```

Наприклад, якщо модель має правильність 79%, то модель правильно передбачила клас для 79% прикладів тестового набору.

Для двовимірних масивів даних можна показати прогнози для всіх можливих точок тестового набору, розмістивши в їх площині xy. Колір площини задано відповідно до того класу, який буде присвоєно точці в цій області. Це дозволяє сформувати границю прийняття рішень (*decision boundary*), яка розбиває площину на дві області: область, де алгоритм прогнозує клас 0, і область, де алгоритм прогнозує клас 1. Рис.4.4. візуалізує границі прийняття рішень для одного, трьох і дев'яти найближчих сусідів.



n_neighbors=1 n_neighbors=3 n_neighbors=9

Рис.4.4. Границі прийняття рішень, створені моделлю найближчих сусідів для різних значень n_neighbors

На рисунку зліва можна побачити, що використання моделі одного найближчого сусіда дає границю прийняття рішень, яка дуже добре узгоджується з навчальними даними. Збільшення числа сусідів призводить до згладжування границі прийняття рішень. Більш гладка границя відповідає більш простій моделі. Іншими словами, використання лише декількох сусідів відповідає високої складності моделі, а використання великої кількості сусідів відповідає низькій складності моделі. Якщо взяти крайній випадок, коли кількість сусідів буде дорівнює кількості точок даних навчального набору, кожна точка тестового набору матиме одних і тих же сусідів (сусідами буде все точки навчального набору) і всі прогнози будуть однаковими: буде обраний клас, який є найбільш часто зустрічається в навчальному наборі.

Регресія k-найближчих сусідів

Існує також варіант регресії за алгоритмом k найближчих сусідів. Найпростіший випадок – використання 1 найближчого сусіда.

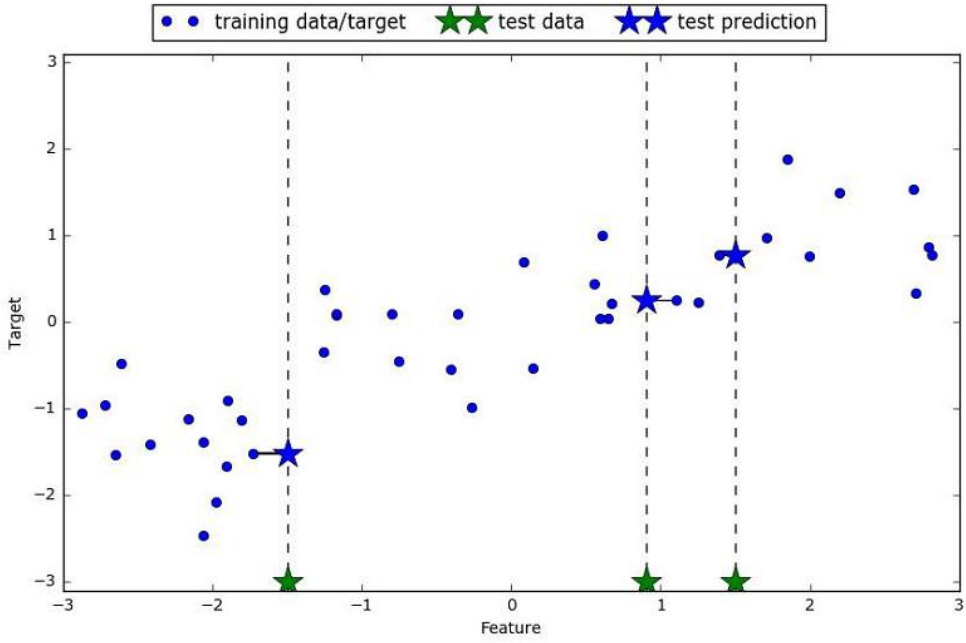


Рис.4.5. Прогнози, отримані за допомогою регресійної моделі одного найближчого сусіда

Для регресії також можна використовувати більшу кількість найближчих сусідів. При використанні декількох найближчих сусідів прогнозом стає середнє значення відповідних сусідів (рис.4.6).

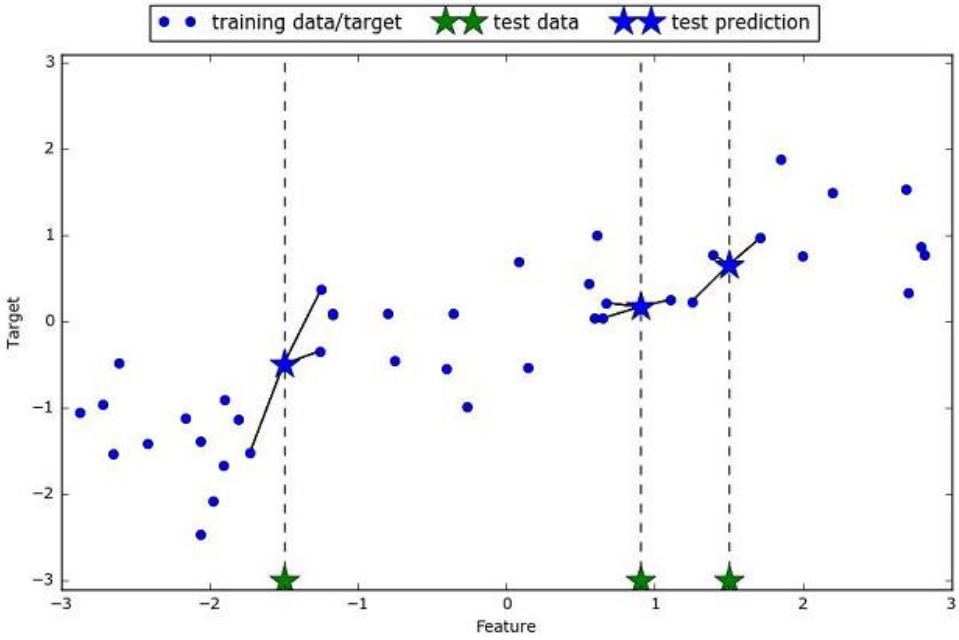


Рис. 4.6. Прогнози, отримані за допомогою регресійної моделі трьох найближчих сусідів

Алгоритм регресії k найближчих сусідів реалізований в класі **KNeighborsRegressor**. Він використовується так само, як і KNeighborsClassifier:

```
from sklearn.neighbors import KNeighborsRegressor
# розбиваємо набір даних на навчальну і тестову вибірки
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
# створюємо екземпляр моделі і встановлюємо кількість сусідів рівним 3
knnreg = KNeighborsRegressor(n_neighbors=3)
# налаштовуємо модель з використанням навчальних даних
knnreg. fit(X_train, y_train)
```

Прогнози для тестового набору:

```
print("Прогнози для тестового набору: {}" . format(knnreg. predict(X_test)))
```

Крім того, можна оцінити якість моделі за допомогою методу **score**, який для регресійних моделей повертає значення R^2. R^2, також відомий як коефіцієнт детермінації, є показником якості регресійній моделі і приймає значення від 0 до 1. Значення 1 відповідає ідеальній прогнозуючій здатності, а значення 0 відповідає константі моделі, яка лише прогнозує середнє значення відповідей в навчальному наборі y_train.

```
print("R^2 на тестовій виборці: {}" . format(knnreg. score(X_test, y_test)))
```

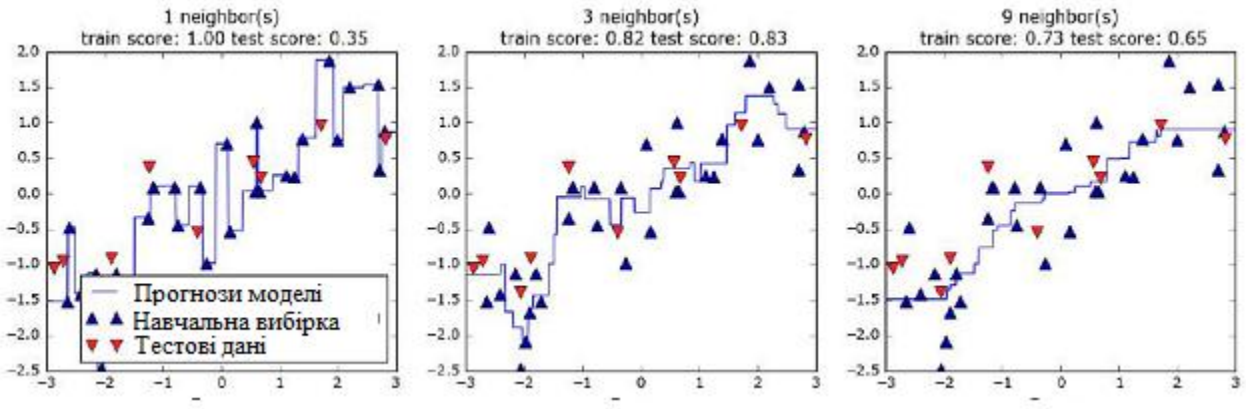


Рис.4.7. Порівняння прогнозів, отриманих за допомогою регресії найближчих сусідів для різних значень n_neighbors

Як видно на рисунку 4.7, при використанні лише одного сусіда кожна точка навчального набору має очевидний вплив на прогнози, і спрогнозовані значення проходять крізь всі точки даних. Це призводить до дуже нестійких прогнозів. Збільшення числа сусідів призводить до отримання більш згладжених прогнозів, але при цьому знижується правильність підгонки до навчальних даних.

В класифікаторі **KNeighbors** є два важливі параметри: *кількість сусідів і міра відстані між точками даних*. На практиці використання невеликого числа сусідів (наприклад, 3-5) часто працює добре, але, звичайно, можна налаштувати цей параметр. За замовчуванням використовується евклідова відстань, яке добре працює в багатьох ситуаціях.

Однією з переваг методу найближчих сусідів є те, що цю модель дуже легко інтерпретувати і, як правило, цей метод дає прийнятну якість без необхідності використання великої кількості налаштувань параметрів. Як правило, побудова моделі найближчих сусідів відбувається дуже швидко, але, коли навчальна вибірка дуже велика (з точки зору кількості характеристик або кількості спостережень) отримання прогнозів може зайняти значний час. При використанні алгоритму найближчих сусідів важливо виконати попередню обробку даних. Окрім того, цей метод не дуже добре працює, коли мова йде про вибірки даних з великою кількістю ознак (сотні і більше).

ЛІНІЙНІ МОДЕЛІ
ЛІНІЙНІ МОДЕЛІ ДЛЯ КЛАСИФІКАЦІЇ ТА РЕГРЕСІЇ

Лінійні моделі являють собою клас моделей, які широко використовуються на практиці і дають прогноз, використовуючи лінійну функцію (linear function) вхідних ознак. Для регресії загальна прогнозна формула лінійної моделі виглядає наступним чином:

$$\hat{y} = w[0] * x[0] + w[1]*x[1] + ... + w[p]*x[p] + b$$

x[0] - x[p] позначають ознаки (в даному прикладі кількість ознак дорівнює p+1) для окремої точки даних, w і b - параметри моделі, які оцінюються в ході навчання, $\hat{y}$ - прогноз, який видається моделлю. Для набору даних з однією ознакою ця формула має вигляд: $\hat{y} = w[0]*x[0]+b$. Ця формула - рівняння прямої. Тут x [0] є нахилом, а b - зміщенням по осі y. Коли використовується кілька ознак, регресійне рівняння містить параметри нахилу для кожної ознаки. Як варіант, прогноз можна представити у вигляді зваженої суми вхідних ознак, де ваги (які можуть бути негативними) задаються елементами w (рис.4.8).

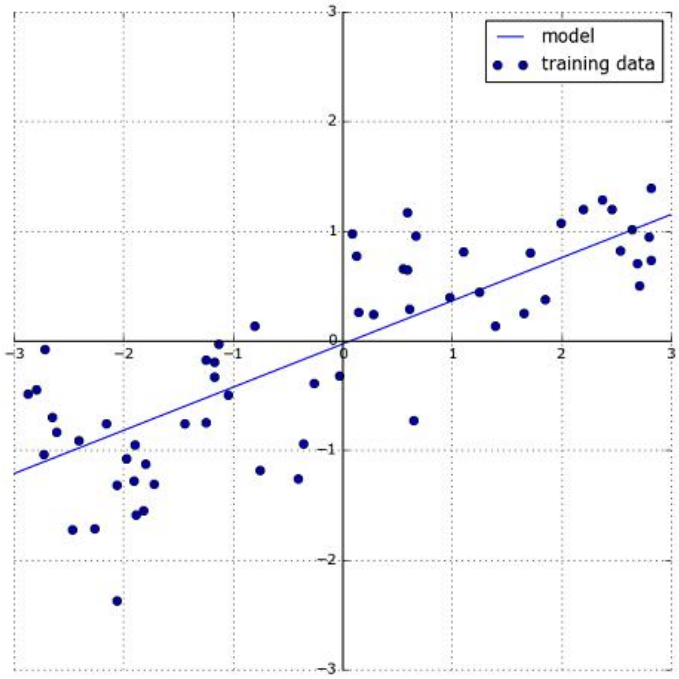


Рис. 4.8. Прогнозування за допомогою лінійної моделі

Лінійні моделі для регресії можна охарактеризувати як регресивні моделі, в яких прогнозом є пряма лінія для однієї ознаки, площина для двох ознак, або гіперплощина для більшої кількості вимірів (тобто, коли використовується багато ознак). Якщо прогнози, отримані за допомогою прямої лінії, порівняти з прогнозами KNeighborsRegressor, використання лінії регресії для отримання прогнозів здається дуже жорстким припущенням. Може здатися, що всі дрібні деталі даних не враховуються. У певному сенсі це вірно через припущення, що цільова змінна y є лінійною комбінацією ознак. Але для наборів даних з великою кількістю ознак лінійні моделі можуть бути дуже корисні. Зокрема, якщо кількість ознак перевищує кількість точок даних для навчання, будь-яку цільову зміну y можна змоделювати (на навчальній вибірці) у вигляді лінійної функції.

Існують різні види лінійних моделей для регресії. Різниця між цими моделями полягає в способі оцінювання параметрів моделі *w* і *b* по навчальних даних і контролі складності моделі.

Лінійна регресія (метод найменших квадратів)

Лінійна регресія або звичайний метод найменших квадратів (ordinary least squares, OLS) - це найпростіший і найбільш традиційний метод регресії. Лінійна регресія знаходить параметри *w* і *b*, які мінімізують средньоквадратичну помилку (mean squared error) між спрогнозованим і фактичним значенням *y* в навчальному наборі. Значення помилки дорівнює сумі квадратів різниць між спрогнозованим і фактичним значеннями. Метод лінійної регресії є простим, що є перевагою, але в той же час у нього немає інструментів, що дозволяють контролювати складність моделі.

```
from sklearn.linear_model import LinearRegression
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
lr = LinearRegression().fit(X_train, y_train)
```

Параметри «нахилу» (*w*), також називають вагами або коефіцієнтами (coefficients), зберігаються в атрибуті **coef_**, тоді як зсув (offset) або константа (intercept), що позначається як b, зберігається в атрибуті **intercept_**:

```
print("lr.coef_: {}". format(lr. coef_))
print("lr.intercept_: {}". format(lr. intercept_))
```

Правильність моделі на навчальному та тестовому наборах:

```
print("Правильність на навчальному наборі: {:.2f}" . format(lr. score(X_train, y_train)))
print("Правильність на тестовому наборі: {:.2f}" . format(lr. score(X_test, y_test)))
```

Якщо спостерігається дуже велика невідповідність між правильністю на навчальному наборі і правильністю на тестовому наборі (наприклад, 0.92 та 0.68), це є явною ознакою перенавчання і тоді необхідно знайти модель, яка дозволить контролювати складність. Одна з найбільш часто використовуваних альтернатив стандартної лінійної регресії – гребнева регресія.

Гребнева регресія

Гребнева регресія також є лінійною моделлю регресії, тому її формула аналогічна тій, що використовується в звичайному методі найменших квадратів. У гребневій регресії коефіцієнти (*w*) вибираються не тільки з точки зору того, наскільки добре вони дозволяють прогнозувати на навчальних даних, вони ще підлаштовуються до додаткових обмежень. Наприклад, потрібно, щоб величина коефіцієнтів була якомога меншою. Іншими словами, все елементи *w* повинні бути близькими до нуля. Це означає, що кожна ознака повинна мати якомога менший вплив на результат (тобто кожна ознака повинна мати невеликий регресійний коефіцієнт) і в той же час він повинна як і раніше мати гарну прогнозну здатність. Це обмеження є прикладом регуляризації (Regularization).

Регуляризація означає явне обмеження моделі для запобігання перенавчання. Регуляризація, що використовується в гребневій регресії, відома як *L2 регулярізація*. Гребнева регресія реалізована в класі **linear_model.Ridge**. Ridge - модель з більш суворим обмеженням, тому менше ймовірність перенавчання.

```
from sklearn.linear_model import Ridge
ridge = Ridge(). fit(X_train, y_train)
print("Правильність на навчальній вибірці: {:.2f}" . format(ridge. score(X_train, y_train)))
print("Правильність на тестовій вибірці: {:.2f}" . format(ridge. score(X_test, y_test)))
```

Модель Ridge дозволяє знайти компроміс між простотою моделі (отриманням коефіцієнтів, близьких до нуля) і якістю її роботи на навчальному наборі. Компроміс між простотою моделі і якістю роботи на навчальному наборі може бути заданий за допомогою параметра **alph**a. Оптимальне значення alpha залежить від конкретного набору даних. Збільшення alpha змушує коефіцієнти стискатися до близьких до нуля значень, що знижує якість роботи моделі на навчальному наборі, але може поліпшити її узагальнюючу здатність.

Збільшення alpha змушує коефіцієнти стискатися до близьких до нуля значень, що знижує якість роботи моделі на навчальному наборі, але може поліпшити її узагальнюючу здатність.

```
ridge10 = Ridge(alpha=10). fit(X_train, y_train)
```

При дуже малих значеннях alpha обмеження на коефіцієнти практично не накладається і в кінцевому підсумку отримується модель, що нагадує лінійну регресію.

```
ridge01 = Ridge(alpha=0.1). fit(X_train, y_train)
```

Лассо регресія

Альтернативою Ridge як методу регуляризації лінійної регресії є Lasso. Як і гребнева регресія, лассо також стискає коефіцієнти до близьких до нуля значень, але трохи іншим способом, що називається *L1 регулярізацією*. Результат L1 регуляризації полягає в тому, що при використанні лассо деякі коефіцієнти стають рівними точно нулю. Виходить, що деякі ознаки повністю виключаються з моделі. Це можна розглядати як один з видів автоматичного відбору ознак. Отримання нульових значень для деяких коефіцієнтів часто спрощує інтерпретацію моделі і може виявити найбільш важливі ознаки моделі.

```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha=0.01, max_iter=100000). fit(X_train, y_train)
```

ЛІНІЙНІ МОДЕЛІ ДЛЯ КЛАСИФІКАЦІЇ

Лінійні моделі також широко використовуються для задачі класифікації. Розглянемо для початку бінарну классифікацію. У цьому випадку прогноз отримують із використанням наступної формули:

$$y\hat{} = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b > 0$$

Формула дуже схожа на формулу лінійної регресії, але замість того, щоб просто повернути зважену суму ознак, для прогнозованого значення задається поріг, рівний нулю. Якщо функція менше нуля, ми прогнозується клас -1, якщо вона більше нуля, прогнозується клас +1. Це прогнозне правило є загальним для всіх лінійних моделей класифікації. Є багато різних способів знайти коефіцієнти (*w*) і константу (*b*). Для лінійних моделей регресії вихідна змінна *y* є лінійною функцією ознак: лінією, площиною або гиперплощиною (для великої кількості вимірювань). Для лінійних моделей класифікації границя прийняття рішень (decision boundary) є лінійною функцією аргументу. Іншими словами, бінарний лінійний класифікатор - це класифікатор, який розділяє два класи за допомогою лінії, площини або гипер площини.

Існує багато алгоритмів навчання лінійних моделей. Два критерії задають відмінності між алгоритмами:

• Вимірювані метрики;

• Факт використання регуляризації і вид регуляризації, якщо вона використовується.

Двома найбільш поширеними алгоритмами лінійної класифікації є ***логістична регресія (logistic regression)***, реалізована в класі **linear_model.LogisticRegression**, і лінійний метод опорних векторів (linear support vector machines) або ***лінійний SVM***, реалізований в класі **svm.LinearSVC** (SVC розшифровується як support vector classifier - класифікатор опорних векторів) (рис.4.9). Незважаючи на назву, логістична регресія є

алгоритмом класифікації, а не алгоритмом регресії, і його не слід плутати з лінійної регресією.

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
logreg = LogisticRegression(). fit(X_train, y_train)
svc= LinearSVC(). fit(X_train, y_train)
```

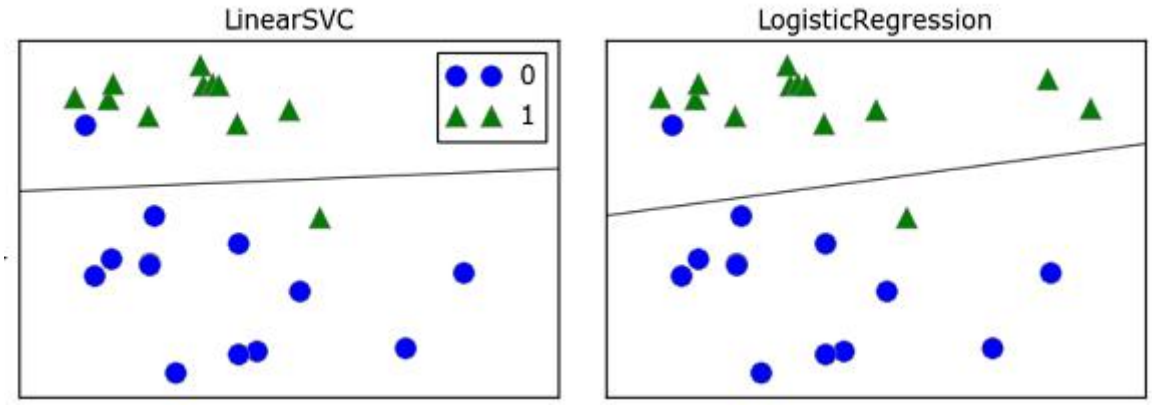


Рис.4.9. Приклад границь прийняття рішень лінійного SVM і логістичної регресії

Границі прийняття рішення представлені у вигляді прямих ліній, що відокремлюють область значень, класифікованих як клас 1 (у верхній частині графіка) від області значень, класифікованих як клас 0 (у нижній частині графіка). Іншими словами, будь-яка нова точка даних, яка лежить вище лінії буде віднесена відповідної моделлю до класу 1, тоді як будь-яка точка, що лежить нижче лінії, буде віднесена до класу 0. Обидві моделі мають схожі границі прийняття рішень. За замовчуванням обидві моделі використовують L2 регуляризацію, той же самий метод, який використовується в гребневій регресії.

Для LogisticRegression і LinearSVC параметр, який визначає ступінь регуляризації, називається C, і більш високі значення C відповідають меншій регуляризації. Коли ви використовується *високе значення параметра C*, LogisticRegression і LinearSVC намагаються підігнати модель до навчальних даних якнайкраще, тоді як при *низьких значеннях параметра C* моделі роблять більший акцент на пошуку вектора коефіцієнтів (*w*), близького до нуля. Існує ще одна цікава деталь, пов'язана з роботою параметра C. Використання низьких значень C призводить до того, що алгоритми намагаються підлаштуватися під «більшість» точок даних, тоді як використання більш високих значень C підкреслює важливість того, щоб кожна окрема точка даних була класифікована правильно.

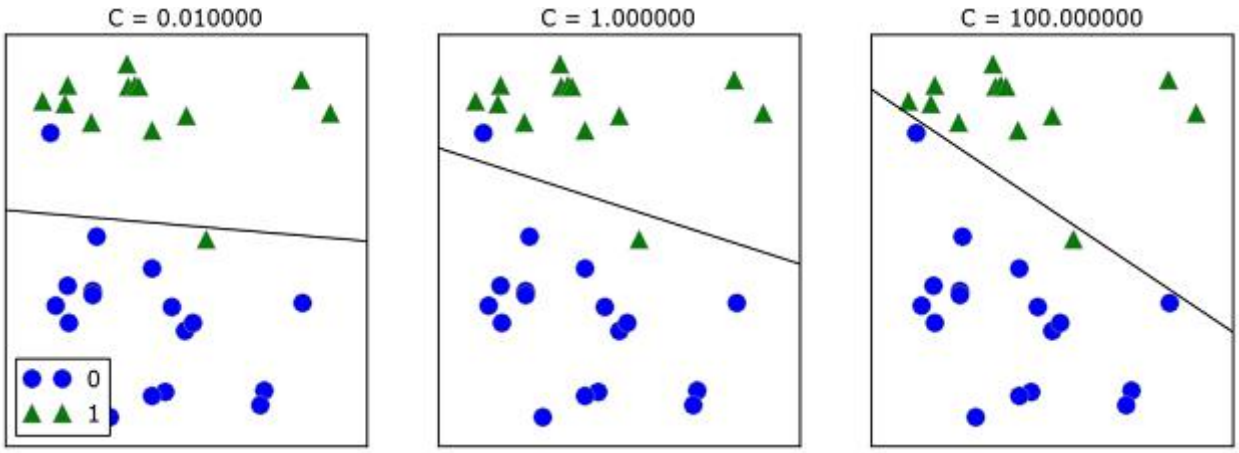


Рис.4.10. Границі прийняття рішень лінійного SVM з різними значеннями параметра C

Оскільки LogisticRegression за замовчуванням використовує L2 регуляризацію, результат схожий на результат, отриманий при використанні моделі Ridge. Якщо потрібно отримати більш інтерпретабельну модель, може допомогти L1 регуляризація, оскільки вона обмежує модель використанням лише кількох ознак.

```
lr_l1 = LogisticRegression(C=C, penalty="l1" ). fit(X_train, y_train)
```

Загальнорозповсюджений підхід, що дозволяє поширити алгоритм бінарної класифікації на випадок мультіклассовой класифікації називається підходом «один проти інших »(one-vs-rest) (рис.4.11). В підході «один проти інших» для кожного класу будується бінарна модель, яка намагається відокремити цей клас від всіх інших, в результаті чого кількість моделей визначається кількістю класів. Для отримання прогнозу точка тестового набору подається на всі бінарні класифікатори. Класифікатор, який видає по своєму класу найбільше значення, «перемагає», і мітка цього класу повертається в якості прогнозу. Використовуючи бінарний класифікатор для кожного класу, отримують один вектор коефіцієнтів (w) і одну константу (b) по кожному класу. Клас, який отримує найбільше значення згідно з наведеною нижче формулою, прогнозується для вхідних даних:

$$w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b.$$

```
linear_svm = LinearSVC(). fit(X, y)
```

Кожен рядок **coef_** містить вектор коефіцієнтів для кожного з класів, а кожен стовпець містить значення коефіцієнта для конкретної ознаки. Атрибут **intercept_** тепер є одновимірним масивом, в якому записані константи класів.

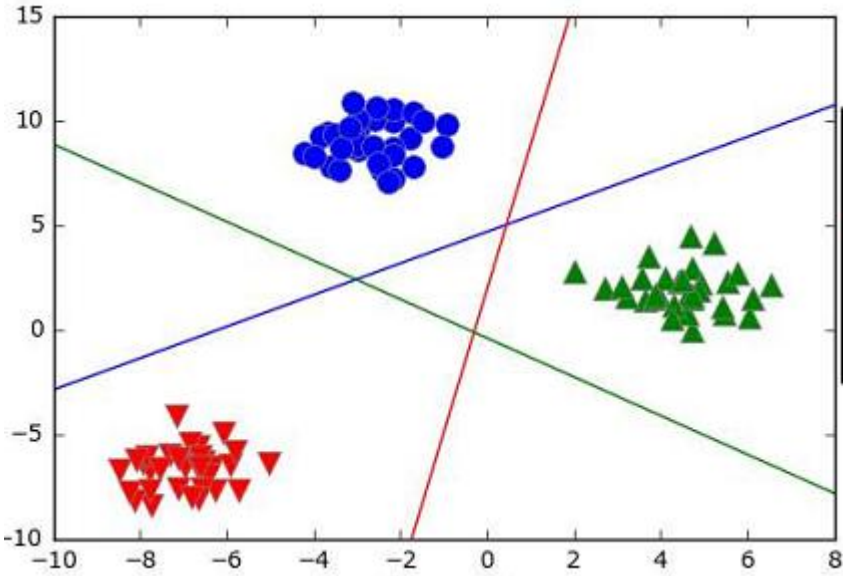


Рис. 4.11. Границі прийняття рішень, отримані за допомогою трьох бінарних класифікаторів в рамках підходу «один проти інших»

НАЇВНИЙ БАЄСІВСЬКИЙ КЛАСИФІКАТОР

Наївні байєсовські класифікатори представляють собою сімейство класифікаторів, які вони оцінюють параметри, розглядаючи кожну ознаку окремо і за кожною ознакою збирають прості статистики класів. У scikit-learn реалізовані три види наївних байесовских класифікаторів: **GaussianNB**, **BernoulliNB** і **MultinomialNB**. BernoulliNB і MultinomialNB в основному використовуються для класифікації текстових даних.

ДЕРЕВА РІШЕНЬ

Дерева рішень є моделями, які широко використовуються для вирішення завдань класифікації і регресії. По суті вони задають питання і вибудовують ієрархію правил «якщо ... то», яка найкраще приводить до рішення. Щоб побудувати дерево, алгоритм перебирає всі можливі тести і знаходить той, який є найбільш інформативним з точки зору прогнозування значень цільової змінної (рис. 4.12-4.14).

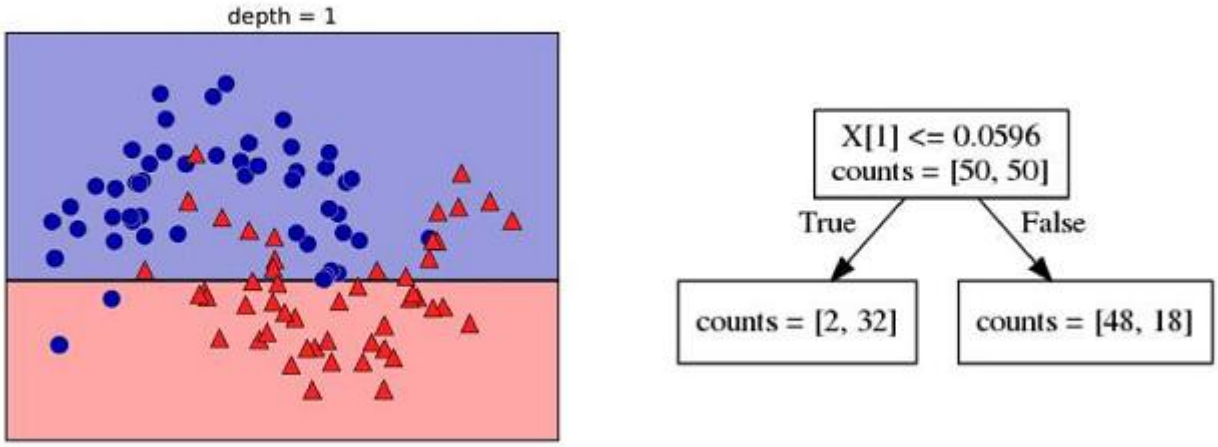


Рис.4.12. Границя прийняття рішення, отримана за допомогою дерева глибиною 1 (ліворуч) і відповідне дерево рішень (праворуч)

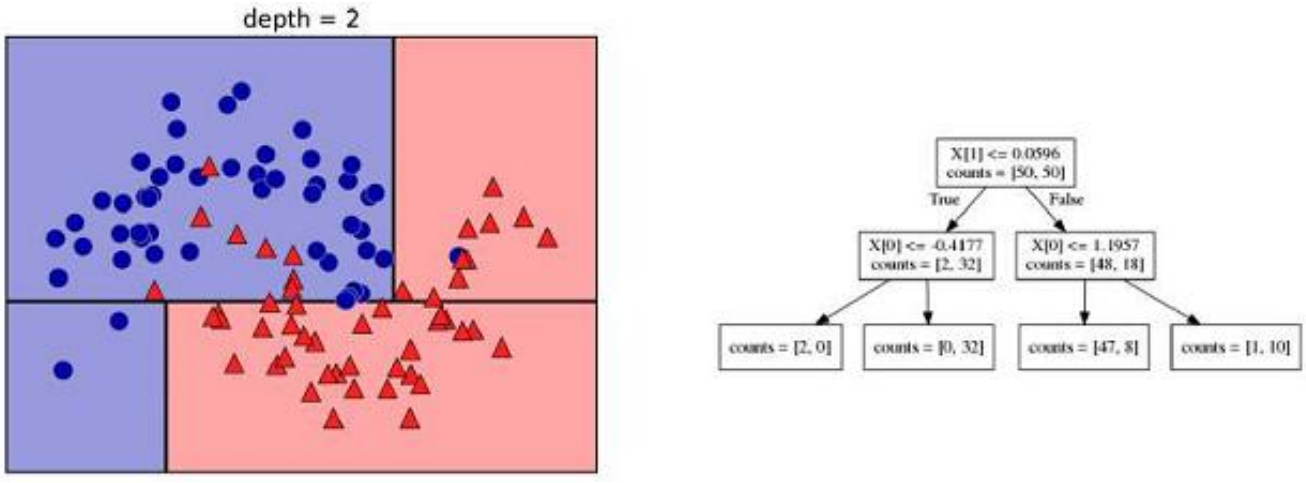


Рис.4.13. Границя прийняття рішення, отримана за допомогою дерева глибиною 2 (ліворуч) і відповідне дерево рішень (праворуч)

Цей рекурсивний процес будує в результаті бінарне дерево рішень, в якому кожен вузол відповідає певному тесту. Крім того, можна інтерпретувати тест як розбиття частини даних, які розглядається в даному випадку уздовж однієї осі. Це дозволяє скласти уявлення про алгоритм як способі вибудувати ієрархію розбиття. Оскільки кожен тест розглядає тільки одну ознаку, області, що виходять в результаті розбиття, завжди мають границі, паралельні осям. Рекурсивне розбиття даних повторюється до тих пір, поки всі точкики даних в кожній області розбиття (кожному листі дерева рішень) не належатимуть одному і тому ж значенню цільової змінної (класу або кількісному значенню). Лист дерева, який містить точки даних, що відносяться до одного і того ж значення цільової змінної, називається чистим (pure).

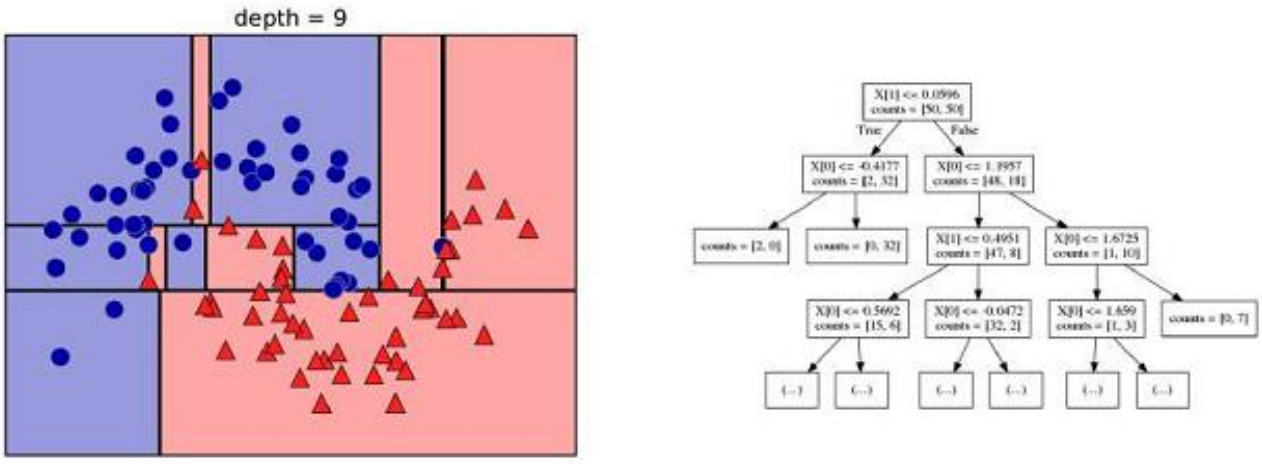


Рис.4.14. Границя прийняття рішень, отримана за допомогою дерева глибиною 9 (зліва) і фрагмент відповідного дерева (праворуч)

Прогноз для нової точки даних одержують у такий спосіб: спочатку з'ясовують, в якій області розбиття простору ознак знаходиться дана точка, а потім визначають клас, до якого належить більшість точок в цій області (або єдиний клас в області, якщо лист є чистим). Область може бути знайдена за допомогою обходу дерева, починаючи з кореневого вузла і шляхом переміщення вліво або вправо, в залежності від того, чи виконується тест чи ні. Крім того, можна використовувати дерева для вирішення завдань регресії, використовуючи такий самий підхід. Для отримання прогнозу ми обходимо

дерево на основі тестів в кожному вузлі і знаходимо лист, в який потрапляє нова точка даних. Виходом для цієї точки даних буде значення цільової змінної, усереднене по всім навчальним точкам в цьому листі.

Можливі критерії попереднього обрізання дерева для уникнення перенавчання включають в себе обмеження максимальної глибини дерева, обмеження максимальної кількості листя або мінімальної кількості спостережень в вузлі, необхідної для розбиття.

У бібліотеці scikit-learn дерева рішень реалізовані в класах **DecisionTreeRegressor** і **DecisionTreeClassifier**.

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=0)
tree. fit(X_train, y_train)
```

Якщо не обмежити глибину, дерево може бути як завгодно глибоким і складним. Тому необрізані дерева схильні до перенавчання і погано узагальнюють результат на нові дані. Один з варіантів - зупинка процессу побудови дерева після досягнення певної глибини. Обмеження глибини дерева зменшує перенавчання. Це призводить до більш низької правильності на навчальному наборі, але покращує правильність на тестовому наборі.

```
tree = DecisionTreeClassifier(max_depth=4, random_state=0)
tree. fit(X_train, y_train)
```

АНСАМБЛІ ДЕРЕВ РІШЕНЬ

Ансамблі (ensembles) - це методи, які поєднують в собі набір моделей машинного навчання, щоб у підсумку отримати більш потужну модель. Існує багато моделей машинного навчання, які належать до цієї категорії, але є дві ансамблеві моделі, які довели свою ефективність на різних наборах даних для

задач класифікації і регресії, обидві використовують дерева рішень в якості будівельних блоків: **випадковий ліс дерев рішень** і **градієнтний бустінг дерев рішень**.

Основним недоліком дерев рішень є їх схильність до перенавчання. **Випадковий ліс** є одним із способів вирішення цієї проблеми. По суті випадковий ліс - це набір дерев рішень, де кожне дерево трохи відрізняється від інших. Ідея випадкового лісу полягає в тому, що кожне дерево може досить добре прогнозувати, але швидше за все перенавчати на частині даних. Якщо побудувати багато дерев, які добре працюють і перенавчають з різним ступенем, можна зменшити перенавчання шляхом усереднення їх результатів.
Для реалізації такої стратегії потрібно побудувати велику кількість дерев рішень. Кожне дерево має на прийнятному рівні прогнозувати цільову змінну і має відрізнятися від інших дерев. Випадкові ліс отримав свою назву через те, що в процесі побудови дерев внесена випадковість, покликана забезпечити унікальність кожного дерева. Існує дві техніки, що дозволяють отримати рандомізовані дерева в рамках випадкового лісу: спочатку вибираються точки даних (спостереження), які будуть використовуватися для побудови дерева, а потім відбираються ознаки в кожному розбитті. Для побудови моделі випадкових лісів необхідно визначитися з кількістю дерев (параметр **n_estimators** для RandomForestRegressor або RandomForestClassifier). Ці дерева будуть побудовані абсолютно незалежно один від одного, і алгоритм буде випадковим чином відбирати ознаки для побудови кожного дерева, щоб отримати несхожі один на одного дерева.

Щоб дати прогноз для випадкового лісу, алгоритм спочатку дає прогноз для кожного дерева в лісі. Для регресії можна усереднити ці результати, щоб отримати остаточний прогноз. Для класифікації використовується стратегія «м'якого голосування». Це означає, що кожен алгоритм дає «м'який» прогноз, обчислюючи ймовірності для кожного класу. Ці ймовірності усереднюються по всьому деревам і прогнозується клас з найбільшою ймовірністю (рис.4.15).

```
from sklearn.ensemble import RandomForestClassifier
forest = RandomForestClassifier(n_estimators=5, random_state=2)
forest. fit(X_train, y_train)
```

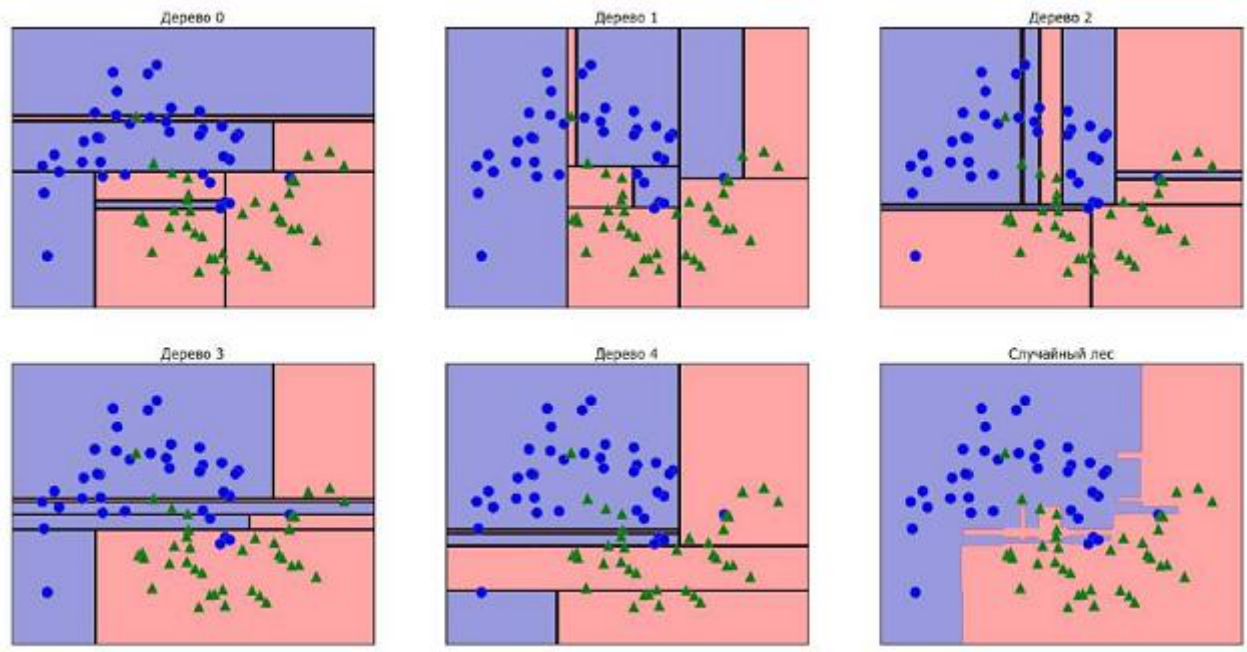


Рис.4.15. Границі прийняття рішень, знайдені п'ятьма рандомізованими деревами рішень, і границя прийняття рішень, отримана шляхом усереднення їх прогнозованих ймовірностей

Градієнтний бустінг дерев рішень – ще один ансамблевий метод, який об'єднує в собі безліч дерев для створення більш потужної моделі. Незважаючи на слово «регресія» в назві, ці моделі можна використовувати для регресії і класифікації. На відміну від випадкового лісу, градієнтний бустінг будує послідовність дерев, в якій кожне дерево намагається виправити помилки попереднього. За замовчуванням в градієнтному бустінзі дерев регресії відсутня випадковість, замість цього використовується сувора попередня обрізка. У градієнтному бустінзі дерев часто використовуються дерева невеликої глибини, від одного до п'яти рівнів, що робить модель менше з точки зору пам'яті і прискорює обчислення прогнозів. Основна ідея градиентного бустінга полягає в об'єднанні безлічі простих моделей, дерев невеликої глибини. Кожне дерево може дати хороші прогнози тільки для частини даних і

таким чином для ітеративного поліпшення якості додається все більша кількість дерев. На відміну від випадкового лісу він, як правило, трохи більше чутливий до налаштування параметрів, однак при правильно заданих параметрах може дати більш високе значення правильності. Крім попереднього обрізання і числа дерев в ансамблі, ще один важливий параметр градиентного бустінга - це **learning_rate**, який контролює, наскільки сильно кожне дерево буде намагатися виправити помилки попередніх дерев. Більш висока швидкість навчання означає, що кожне дерево може внести більш сильні коригування і це дозволяє отримати більш складну модель. Додавання більшої кількості дерев в ансамбль, здійснюване за рахунок збільшення значення **n_estimators**, також збільшує складність моделі, оскільки модель має більше шансів виправити помилки на навчальному наборі.

```
from sklearn.ensemble import GradientBoostingClassifier
gbrt = GradientBoostingClassifier(random_state=0)
gbrt. fit(X_train, y_train)
```

ЯДЕРНИЙ МЕТОД ОПОРНИХ ВЕКТОРІВ

Ядерний метод опорних векторів (SVM) - це розширення методу опорних векторів, воно дозволяє отримувати більш складні моделі, які не зводяться до побудови простих гіперплощин в просторі.

В ході навчання SVM обчислює важливість кожної точки навчальних даних з точки зору визначення границі прийняття рішення між двома класами. Зазвичай лише частина точок навчального набору важлива для визначення границі прийняття рішень: точки, які лежать на границі між класами. Вони називаються опорними векторамі (support vectors) і дали свою назву машині опорних векторів SVM. Щоб отримати прогноз для нової точки, вимірюється відстань до кожного опорного вектора. Класифікаційне рішення приймається,

виходячи з відстаней до опорних векторів, а також важливості опорних векторів, отриманих в процесі навчання (зберігаються в атрибуті **dual_coef_** класу SVC) (рис. 4.16).

Відстань між точками даних вимірюється за допомогою гаусівського ядра:

$k_{rbf}(x_1, x_2) = \exp(-\gamma \| x_1 - x_2\|^2)$

x_1, x_2 – точки даних, $\| x_1 - x_2\|$ - евклідова відстань, γ - параметр, який регулює ширину гаусівського ядра.

```
from sklearn.svm import SVC
svm = SVC(kernel='rbf' , C=10, gamma=0.1). fit(X, y)
```

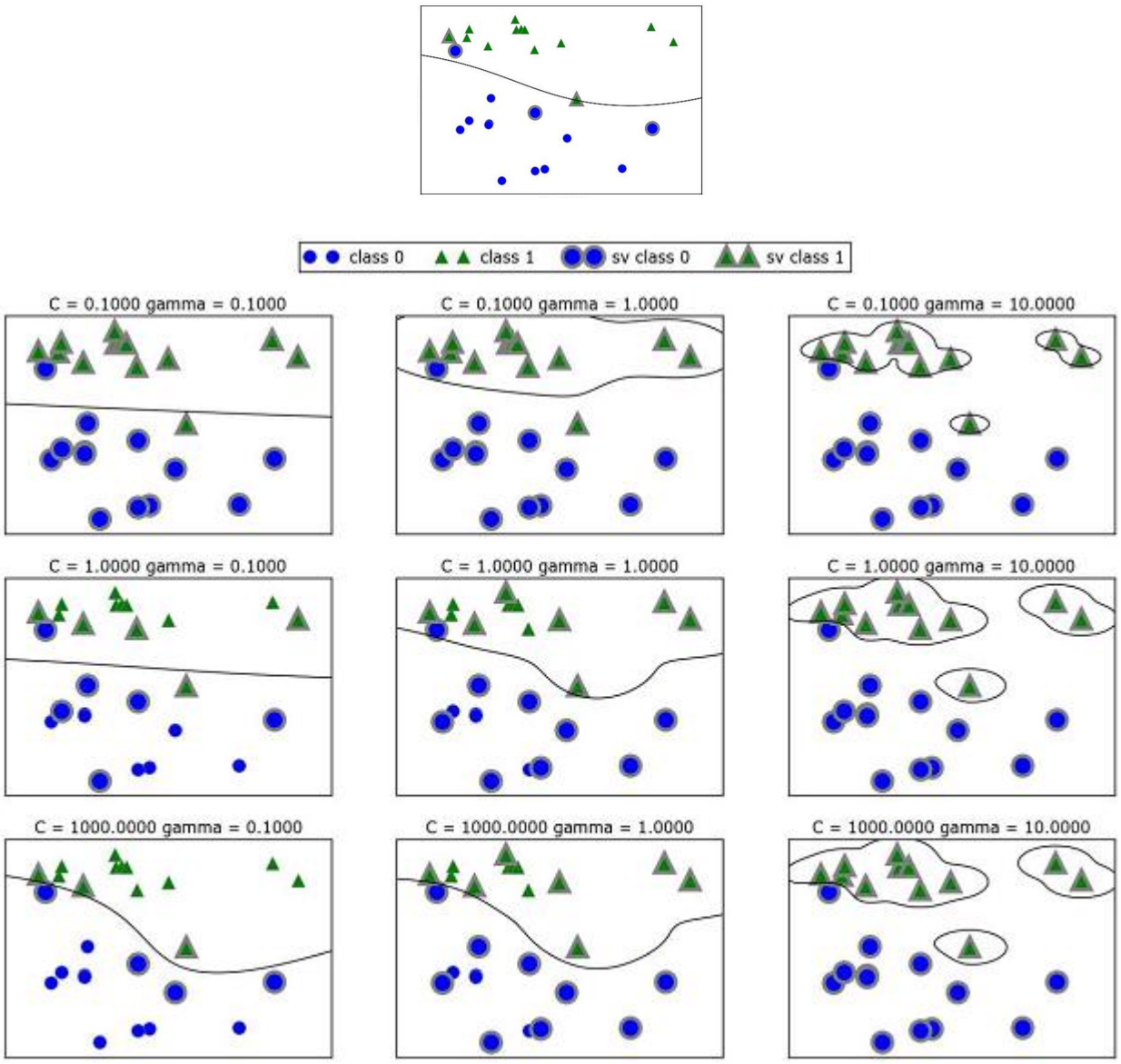


Рис.4.16. Границя прийняття рішень і опорні вектори,
знайдені SVM за допомогою ядра RBF при різних параметрах C та gamma

SVM вимагає ретельної попередньої обробки даних і налаштування параметрів. Загальнопоширений метод масштабування для ядерного SVM полягає в масштабуванні даних так, щоб всі ознаки приймали значення від 0 до 1. Важливими параметрами ядерного SVM є параметр регуляризації C, тип ядра, а також параметри, які визначаються ядром.

НЕЙРОННІ МЕРЕЖІ

Багатошарові персептрони (MLP) також називають простими нейронними мережами прямого поширення. MLP можна розглядати як узагальнення лінійних моделей, яке перш ніж прийти до прийняття рішення виконує кілька етапів обробки даних. Згадаймо, що в лінійної регресії прогноз отримують за допомогою наступної формули:

$$\hat{y} = w[0] * x[0] + w[1] * x[1] + ... + w[p] * x[p] + b,$$

де $\hat{y}$ це зважена сума вхідних ознак $x[0]$ ... $x[p]$. Вхідні ознаки зважені за обчисленими під час навчання коефіцієнтами $w[0]$ ... $w[p]$. Можна представити їх графічно, як показано на рис.4.17.

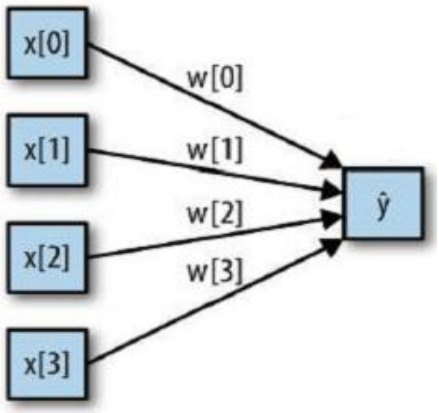


Рис.4.17. Візуалізація логістичної регресії, в якій вхідні ознаки і прогнози показані в вигляді вузлів, а коефіцієнти - у вигляді сполучень між вузлами

Тут кожен вузол, показаний зліва, є вхідною ознакою, з'єднувальні лінії - коефіцієнти, а вузол справа – це вихід, який є зваженою сумою входів. У MLP

процес обчислення зважених сум повторюється кілька разів. Спочатку обчислюються приховані елементи (hidden units), які являють собою проміжний етап обробки. Вони знову об'єднуються за допомогою зважених сум для отримання кінцевого результату (рис.4.18).

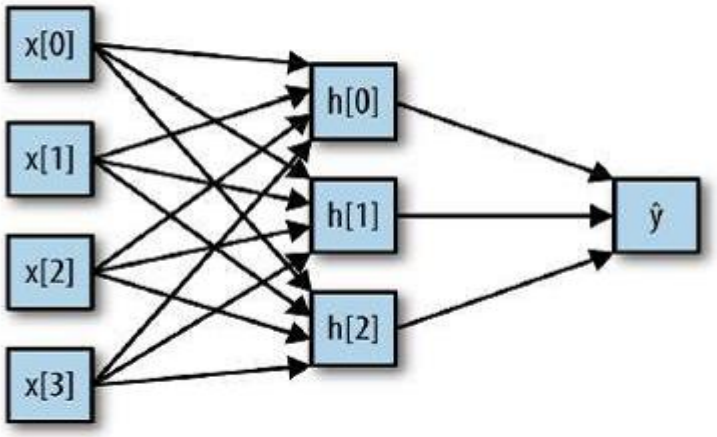


Рис.4.18. Ілюстрація багатошарового персептрона з одним прихованим шаром

У цій моделі набагато більше обчислюваних коефіцієнтів (які називаються вагами): коефіцієнт між кожним входом і кожним прихованим елементом (які утворюють прихований шар hiddenlayer) і коефіцієнт між кожним елементом прихованого шару і виходом (рис.4.19).

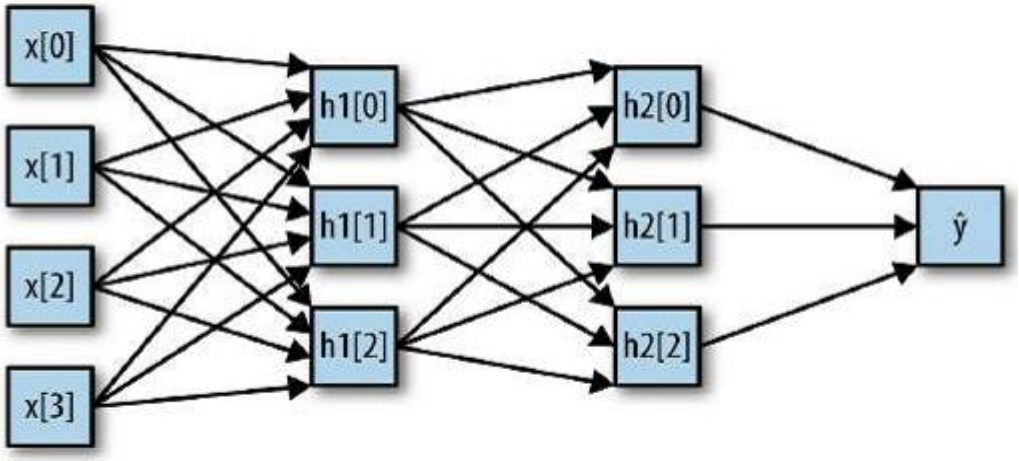


Рис.4.19. Багатошаровий персептрон з двома прихованими шарами

Коли відбувається обчислення зваженої суми входів для кожного елемента прихованого шару, до неї застосовується ***функція активації*** -

зазвичай використовуються нелінійні функції випрямлений лінейний елемент (rectified linear unit або *relu*) або гіперболічний тангенс (hyperbolic tangent або *tanh*). Як результат отримаються виходи нейронів прихованого шару. Ці проміжні виходи можуть вважатися нелінійними перетвореннями і комбінаціями первинних входів. Вони стають входами вихідного шару. Знову обчислюється зважена суму входів, застосовується функція активації і отримуються підсумкові значення цільової змінної. Функції активації relu і tanh показані на рис. 4.20. Relu відсікає значення нижче нуля, в той час як tanh приймає значення від -1 до 1 (відповідно для мінімального і максимального значень входів).

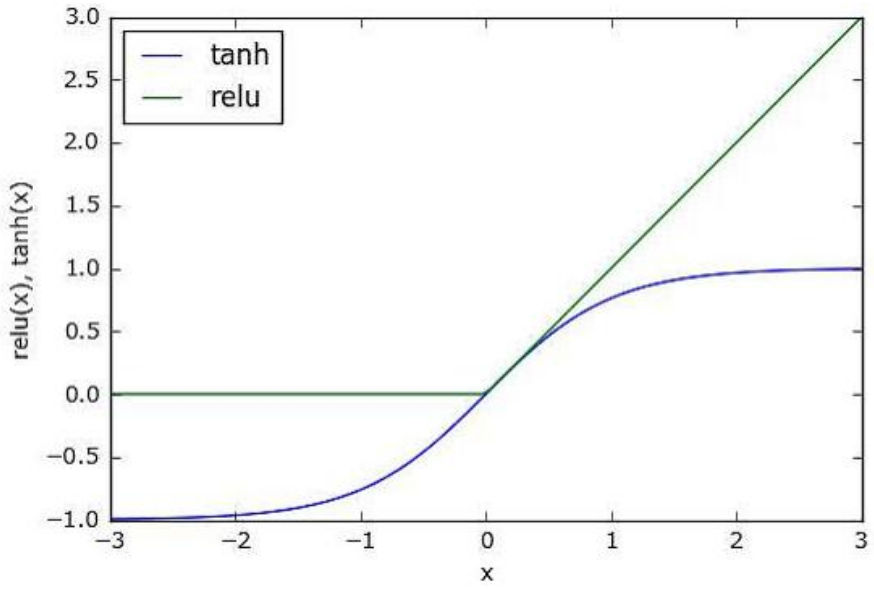


Рис.4.20. Функція активації гіперболічний тангенс
і функція активації випрямленої лінійного елемента

$h[0] = \tanh(w[0,0] * x[0] + w[1,0] * x[1] + w[2,0] * x[2] + w[3,0] * x[3])$

$h[1] = \tanh(w[0,0] * x[0] + w[1,0] * x[1] + w[2,0] * x[2] + w[3,0] * x[3])$

$h[2] = \tanh(w[0,0] * x[0] + w[1,0] * x[1] + w[2,0] * x[2] + w[3,0] * x[3])$

$\hat{y} = v[0] * h[0] + v[1] * h[1] + v[2] * h[2]$

Тут w - ваги між входом x і прихованим шаром h, а v – вагові коефіцієнти між прихованим шаром h і виходом $\hat{y}$. Ваги v і w обчислюються за даними, x є вхідними ознаками, $\hat{y}$ - обчислений вихід, а h - проміжні обчислення.

Важливий параметр, який повинен задати користувач - кількість вузлів в прихованому шарі (рис.4.21). Побудова великих нейронних мереж, що складаються з багатьох шарів обчислень, ввела термін «глибоке навчання» («**deep learning**»).

```
from sklearn.neural_network import MLPClassifier
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y, random_state=42)
mlp = MLPClassifier(solver='lbfgs' , random_state=0, hidden_layer_sizes=[10])
mlp. fit(X_train, y_train)

# приклад використання двох прихованих шарів по 10 елементів в кожному, з функцією tanh
mlp = MLPClassifier(solver='lbfgs' , activation='tanh' , random_state=0,
hidden_layer_sizes=[10, 10])
mlp. fit(X_train, y_train)
```

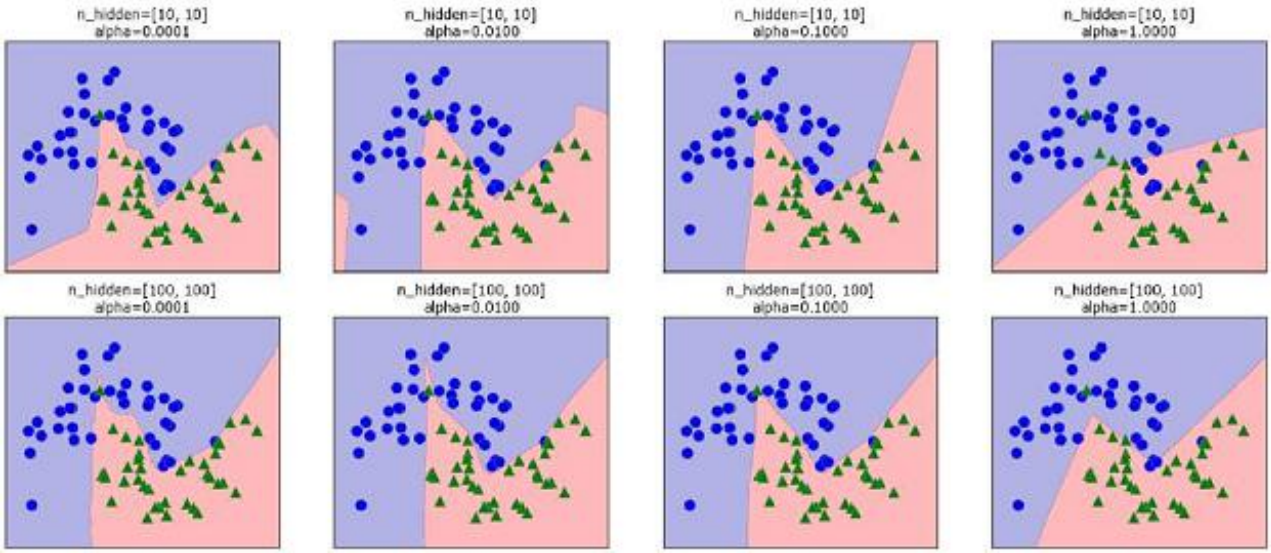


Рис.4.21. Границі прийняття рішень для різної кількості прихованих елементів і різних значень параметра alpha

Важливою властивістю нейронних мереж є те, що їх ваги задаються випадковим чином перед початком навчання і випадкова ініціалізація впливає

на процес навчання моделі. Це означає, що навіть при використанні одних і тих же параметрів можна отримати дуже різні моделі, задаючи різні стартові значення генератора псевдовипадкових чисел.

Нейронні мережі також як і SVM вимагають того, щоб все вхідні ознаки були виміряні в одному і тому ж масштабі, в ідеалі вони повинні мати нульове середнє і дисперсію 1.

```
# обчислюємо середнє значення для кожної ознаки навчального набору
mean_on_train = X_train. mean(axis=0)
# обчислюємо стандартне відхилення для кожної ознаки навчального набору
std_on_train = X_train. std(axis=0)
# віднімаємо середнє і потім помножуємо на зворотну величину стандартного відхилення
# mean=0 и std=1
X_train_scaled = (X_train - mean_on_train) / std_on_train
# використовуємо те саме перетворення (середнє і стандартне відхилення навчального набору) для тестового набору
X_test_scaled = (X_test - mean_on_train) / std_on_train
mlp = MLPClassifier(random_state=0)
mlp. fit(X_train_scaled, y_train)
```

Одна з базових переваг нейронних мереж полягає в тому, що вони здатні обробляти інформацію, що міститься у великих обсягах даних, і будувати неймовірно складні моделі. При наявності достатнього часу обчислень, даних і ретельного налаштування параметрів нейронні мережі часто перевершують інші алгоритми машинного навчання. Але нейронні мережі, особливо великі нейронні мережі, як правило, вимагають тривалого часу навчання. Також вони вимагають попередньої обробки даних.

Перший графік на рис. 4.22 відповідає синтетичному двукласовому набору даних з двома ознаками. Перша ознака (вісь x) приймає значення в діапазоні від 10 до 15. Друга ознака (вісь y) приймає значення приблизно в діапазоні від 1 до 9. Наступні чотири графіка показують чотири різні способи перетворення даних, які дають більш стандартні діапазони значень.

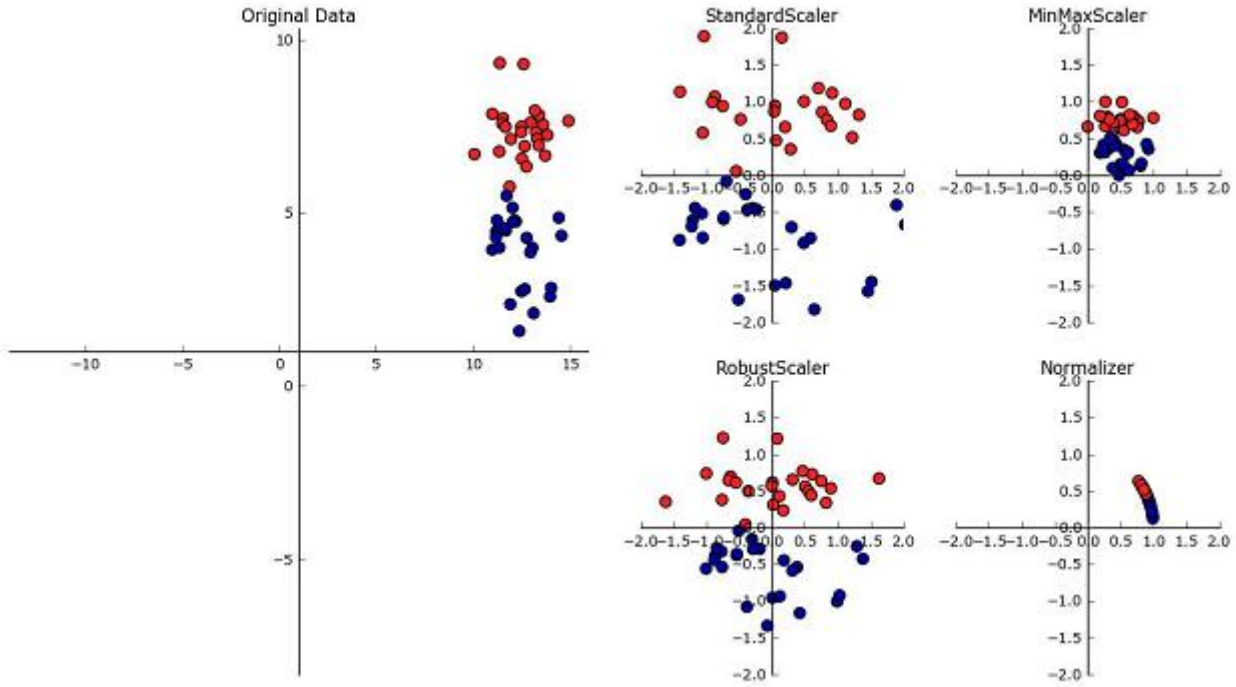


Рис.4.22. Різні способи масштабування і попередньої обробки даних

Застосування

StandardScaler в scikit-learn гарантує, що для кожної ознаки середнє дорівнюватиме 0, а дисперсія буде дорівнювати 1, в результаті чого всі ознаки матимуть один і той самий масштаб. Однак це масштабування не гарантує отримання якихось конкретних мінімальних і максимальних значень ознак.

RobustScaler аналогічний StandardScaler в тому плані, що в результаті його застосування ознаки матимуть один і той же масштаб. Однак RobustScaler замість середнього та дисперсії використовує медіану і квартилі. Це дозволяє RobustScaler ігнорувати точки даних, які сильно відрізняються від інших

(наприклад, помилки вимірювань), які ще називаються викидами (outliers) і можуть стати проблемою для інших методів масштабування.

MinMaxScaler перетворює дані таким чином, щоб всі ознаки знаходилися строго в діапазоні від 0 до 1. Для двовимірного набору даних це означає, що всі дані містяться в прямокутнику, утвореному віссю x з діапазоном значень від 0 і 1 і віссю y з діапазоном значень від 0 і 1.

Normalizer здійснює зовсім інший вид масштабування. Він масштабує кожну точку даних таким чином, щоб вектор ознак мав евклидову довжину 1. Іншими словами, він проектує точку даних на коло з радіусом 1 (або сферу в разі великого числа вимірювань). Вектор множиться на інверсію своєї довжини. Подібна нормалізація використовується тоді, коли важливим є напрямок (але не довжина) вектора ознак.

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler. fit(X_train)
X_train_scaled = scaler. transform(X_train)
```

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler. fit(X). transform(X)
```

МЕТОДИ МАШИННОГО НАВЧАННЯ БЕЗ ВЧИТЕЛЯ

Машинне навчання без вчителя включає в себе всі види машинного навчання, коли відповідь невідома і відсутній учитель, який вказує відповідь алгоритму. У машинному навчанні без учителя є лише вхідні дані і алгоритму необхідно витягти знання з цих даних. Головна проблема машинного навчання без учителя - оцінка корисності інформації, отриманої алгоритмом. Алгоритми машинного навчання без вчителя, як правило, застосовуються до даних, які не

містять ніяких міток, таким чином, відомо, якою повиненна бути правильна відповідь. Тому дуже важко судити про якість роботи моделі. Як наслідок, алгоритми машинного навчання без вчителя часто використовуються в розвідувальних цілях, коли фахівець хоче краще вивчити самі дані.

Алгоритми **кластеризації** (clustering algorithms) розбивають дані на окремі групи схожих між собою елементів, які називаються кластерами. Мета - розділити дані таким чином, щоб точки, що знаходять в одному і тому ж кластері, були дуже схожі один з одним, а точки, що знаходяться в різних кластерах, відрізнялися одна від одної. Як і алгоритми класифікації, алгоритми кластеризації прогнозують кожній точці даних номер кластера, якому вона належить.

КЛАСТЕРИЗАЦІЯ К-СЕРЕДНІХ

Кластеризація k-середніх - один з найпростіших алгоритмів кластеризації. Спочатку вибирається число кластерів k. Після вибору значення k алгоритм k-середніх відбирає точки, які будуть представляти центри кластерів (cluster centers). Потім для кожної точки даних обчислюється його евклідова відстань до кожного центру кластера. Кожна точка призначається найближчому центру кластера. Алгоритм обчислює центроїди (*centroids*) – центри важкості кластерів. Кожен центр ваги - це вектор, елементи якого являють собою середні значення характеристик, обчислені по всіх точках кластера. Центр кластера зміщується в його центр ваги. Точки заново призначаються найближчому центру кластера. Етапи зміни центрів кластерів і перепризначення точок ітеративно повторюються до тих пір, поки границі кластерів і розташування центроїдів не перестануть змінюватися, тобто на кожній ітерації в кожен кластер будуть потрапляти одні і ті ж точки даних (рис.4.23).

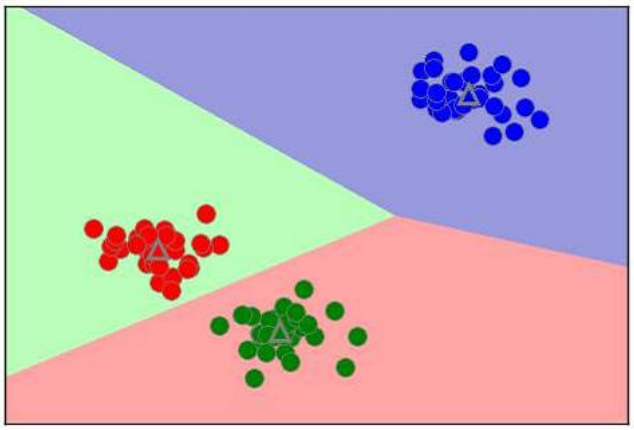


Рис. 4.23. Центри кластерів і кордони кластерів, знайдені за допомогою алгоритму k-середніх

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3)
kmeans. fit(X)
```

Під час роботи алгоритму кожній точці навчальних даних X присвоюється мітка кластера, мітки знаходяться в атрибуті **kmeans.labels_**

```
print("Приналежність до кластерів: \n{}" . format(kmeans. labels_))
```

Навіть якщо є відомою «правильна» кількість кластерів для конкретного набору даних, алгоритм k-середніх не завжди може виділити їх. Кожен кластер визначається виключно його центром, це означає, що кожен кластер має опуклу форму. В результаті цього алгоритм k-середніх може описати відносно прості форми. Крім того, алгоритм k-середніх передбачає, що всі кластери в певному сенсі мають однаковий «діаметр», він завжди проводить кордон між кластерами так, щоб вона проходила точно посередині між центрами кластерів.

АГЛОМЕРАТИВНА КЛАСТЕРИЗАЦІЯ

Алгомеративна кластеризація (agglomerative clustering) відноситься до сімейства алгоритмів кластеризації, в основі яких лежать однакові принципи:

алгоритм починає свою роботу з того, що кожну точку даних заносить в свій власний кластер і в міру виконання умов об'єднує два найбільш схожих між собою кластера до тих пір, поки не буде задоволено певний критерій зупинки (рис.4.24).

Критерій зупинки, реалізований в scikit-learn - це кількість кластерів, тому схожі між собою кластери об'єднуються до тих пір, поки не залишиться задана кількість кластерів. Є кілька критеріїв зв'язку (linkage), які задають точний спосіб вимірювання «найбільшої схожості кластерів». В основі цих критеріїв лежить відстань між двома існуючими кластерами.

У scikit-learn реалізовані наступні три критерії:

Ward - метод за замовчуванням (метод Варда) вибирає і об'єднує два кластера так, щоб приріст дисперсії всередині кластерів був мінімальним. Часто цей критерій призводить до отримання кластерів однакового розміру.

Average - метод average (метод середнього зв'язку) об'єднує два кластери, які мають найменше середнє значення всіх відстаней, виміряних між точками двох кластерів.

Complete - метод complete (метод повного зв'язку або метод максимальної зв'язку) об'єднує два кластери, які мають найменшу відстань між двома їх найбільш віддаленими точками.

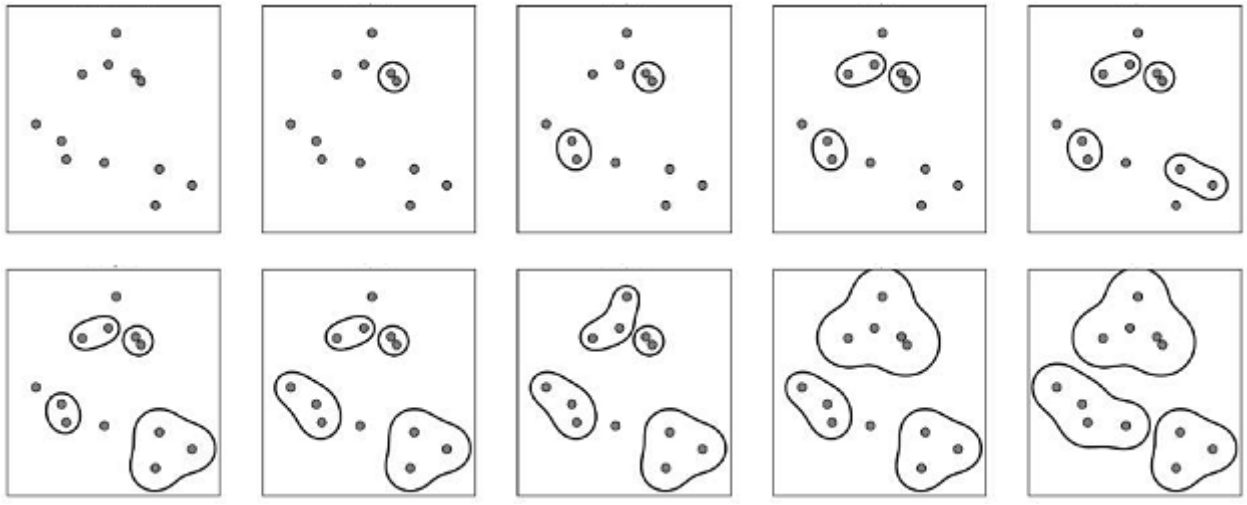


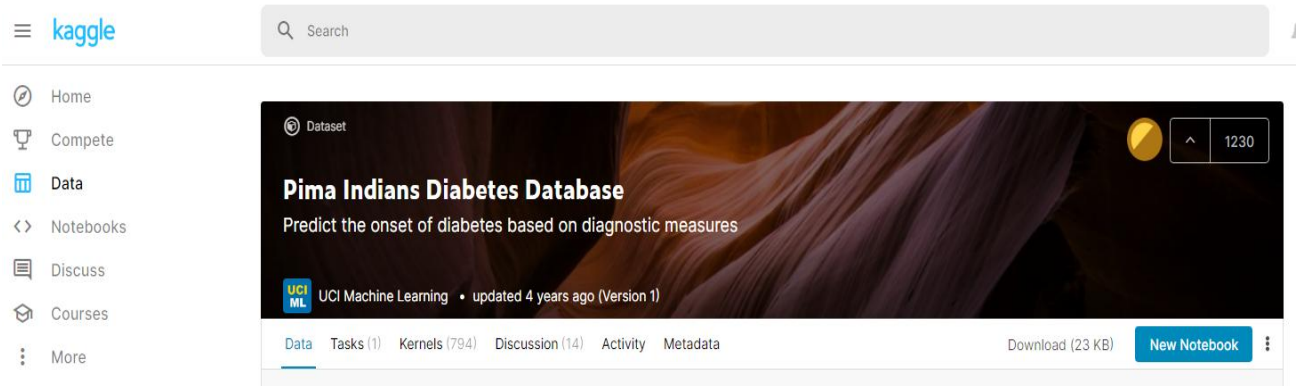
Рис.4.24. Алгоритм агломеративної кластеризації ітеративно об'єднує два найближчих кластера

```
from sklearn.cluster import AgglomerativeClustering
agg = AgglomerativeClustering(n_clusters=3)
assignment = agg. fit_predict(X)
```

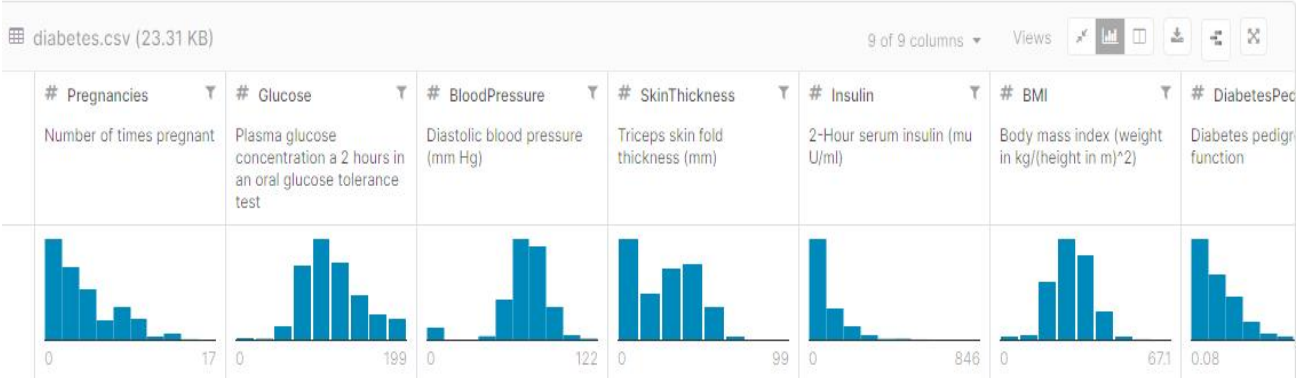
РЕАЛІЗАЦІЯ МАШИННОГО НАВЧАННЯ НА МОВІ PYTHON НА ПРИКЛАДІ ПРОГНОЗУВАННЯ ДІАБЕТУ У ВАГІТНИХ

Завантажити файл з даними за посиланням

https://www.kaggle.com/uciml/pima-indians-diabetes-database#diabetes.csv



Дані містять виміри в популяції американських індіанців племені Піма, серед яких відзначається висока вірогідність діабету. Це один з типових наборів даних для тестування різних алгоритмів машинного навчання. Набір даних складається з декількох медичних прогнозних (незалежних) змінних та однієї цільової (залежної) змінної, що є результатом (мітка класу). Незалежні змінні включають кількість вагітностей у пацієнтки, індекс маси тіла, рівень інсуліну, вік тощо.



Набір містить дані 767 вагітних жінок, здорових і хворих на діабет. Кожен ряд відповідає окремій пацієнтці, кожен стовпець – певний показник.

Змінні:
Pregnancies – кількість вагітностей
Glucose – концентрація глюкози в плазмі при крові (через 2 години після глюкозного тесту);
BloodPressure – діастолічний артеріальний тиск (мм рт.ст.);
SkinThickness – товщина складки шкіри трицепса (мм);
Insulin – рівень інсуліну у сироватці крові через 2 години після глюкозного тесту (мкг/мл);
BMI – індекс маси тіла (вага у кг/(зріст у м)^2);
DiabetesPedigreeFunction – індекс зустрічаємості діабету в родині;
Age – Вік (років);
Outcome Class variable (0 or 1) – 268 випадків з 768 мають 1, інші 0. Якщо 1, то у пацієнтки діагностували діабет. Якщо 0, то пацієнтка ніколи не страждала на діабет.

Імпортувати необхідні бібліотеки (наприклад, pandas, numpy)

 # використано в цьому прикладі

```
import pandas as pd
from pandas.plotting import scatter_matrix
import numpy as np
import os
import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler as Scaler
from sklearn import model_selection
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeRegressor
from sklearn.neural_network import MLPClassifier
```

```
#встановити шлях до папки з даними
dataset_path = 'diabetfolder/'
data_path = os.path.join(dataset_path, 'diabetes.csv')
dataset = pd.read_csv(data_path)
#розмірність даних
dataset.shape

#подивитися початок таблиці даних
dataset.head()
```

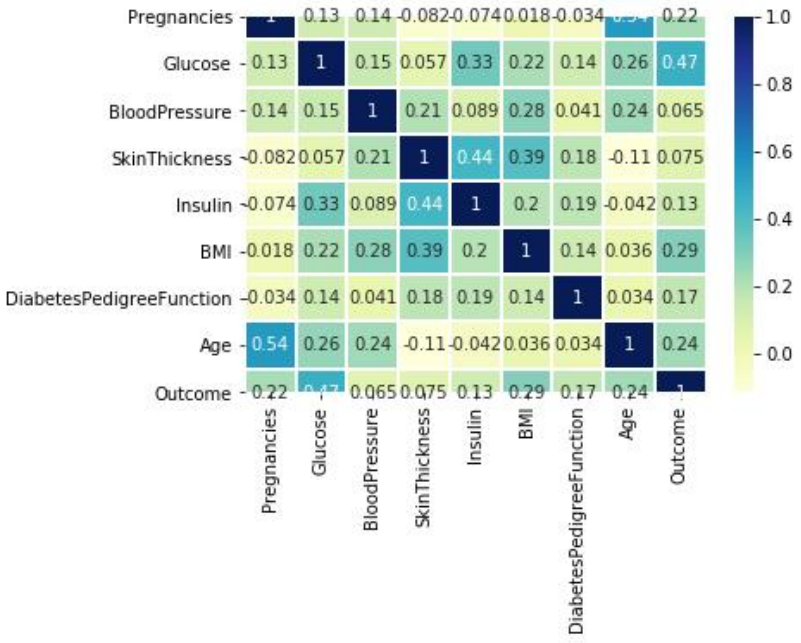
Результат: (768, 9)

Результат:

	Pregnancies	Glucose	BloodPressure	...	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	...	0.627	50	1
1	1	85	66	...	0.351	31	0
2	8	183	64	...	0.672	32	1
3	1	89	66	...	0.167	21	0
4	0	137	40	...	2.288	33	1

З метою розуміння кореляції між різними характеристиками необхідно проаналізувати кореляційну матрицю. Значення у кореляційній матриці варіюються від -1 до 1, чим ближче значення до 1, тим краще співвідношення між двома характеристиками.

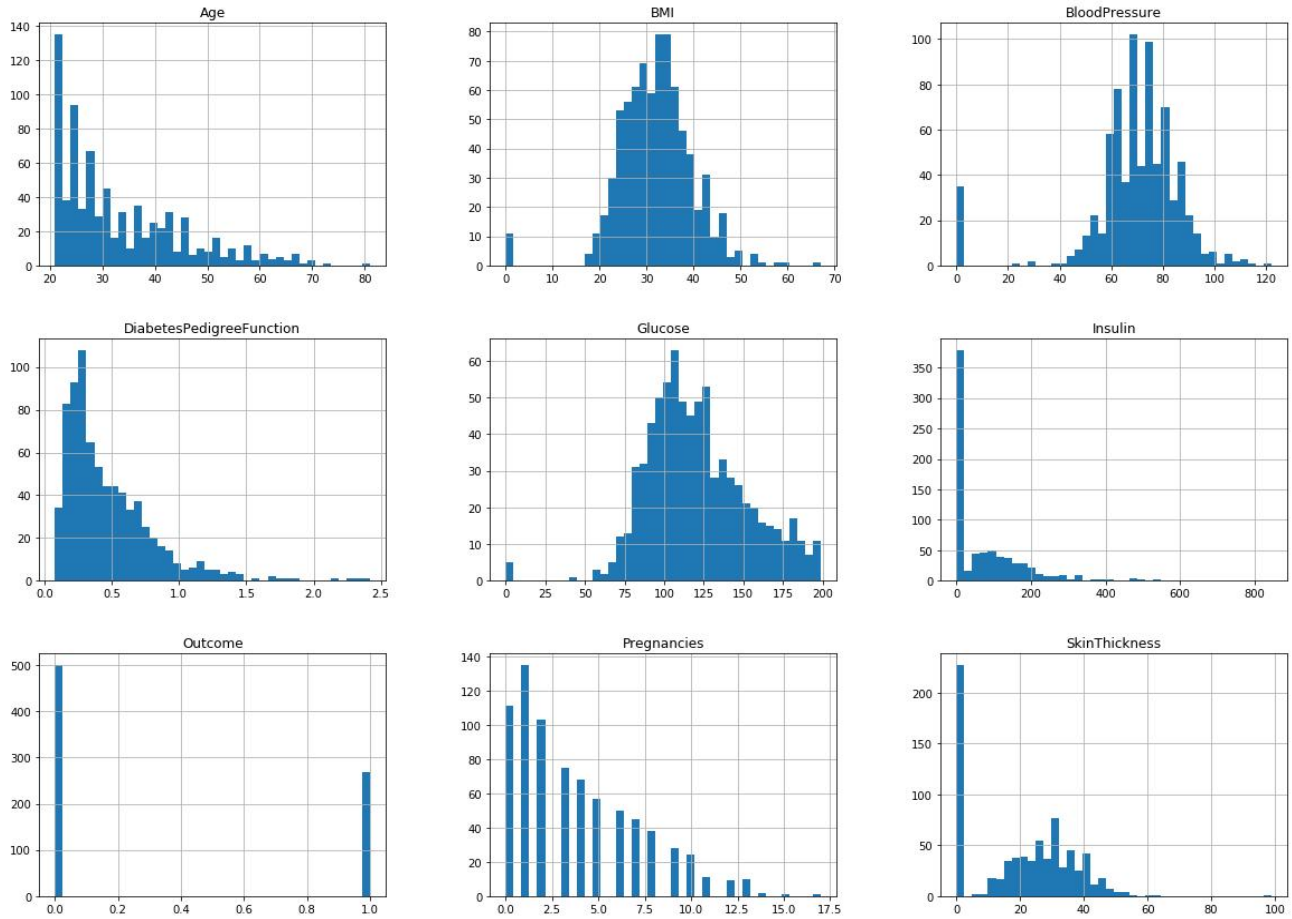
```
#Кореляційна матриця
corr = dataset.corr()
sns.heatmap(corr, annot=True, cmap="YlGnBu", linewidths=1)
```



#гістограми розподілу для ознак

dataset.hist(bins=40, figsize=(20, 15))

plt.show()



За допомогою візуалізації гістограм розподілу ознак можна проаналізувати дані та виявити відсутні або некоректні дані (наприклад, індекс маси тіла або тиск = 0), які необхідні замінити на середні значення або медіану.

#Згадаємо змінні - Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome

 # Медіана для BloodPressure
median_bloodpr = dataset['BloodPressure'].median()
#Замінимо нульові значення на медіану
dataset['BloodPressure'] = dataset['BloodPressure'].replace(to_replace=0, value=median_bloodpr)

 # Медіана для SkinThickness
median_bmi = dataset['SkinThickness'].median()

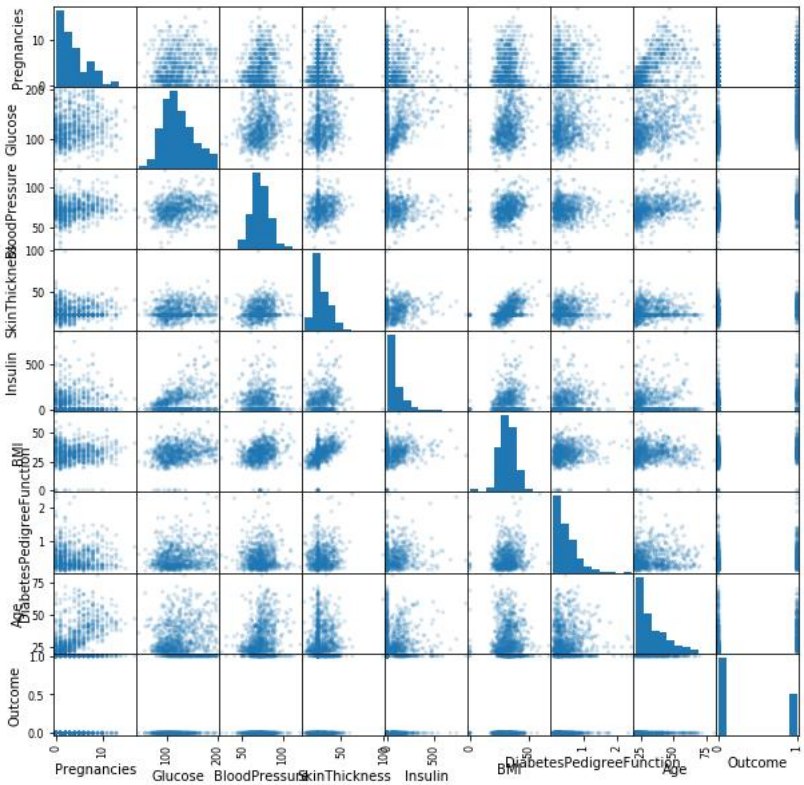
```
#Замінимо нульові значення на медіану
dataset['SkinThickness'] = dataset['SkinThickness'].replace(to_replace=0,
value=median_bmi)

# Медіана для SkinThickness
median_bmi = dataset['SkinThickness'].median()
#Замінимо нульові значення на медіану
dataset['SkinThickness'] = dataset['SkinThickness'].replace(to_replace=0,
value=median_bmi)

# Медіана для Glucose
median_bmi = dataset['Glucose'].median()
#Замінимо нульові значення на медіану
dataset['Glucose'] = dataset['Glucose'].replace(to_replace=0, value=median_bmi)

# Медіана для Age
median_bmi = dataset['Age'].median()
#Замінимо нульові значення на медіану
dataset['Age'] = dataset['Age'].replace(to_replace=0, value=median_bmi)

#парні діаграми розсіювання ознак
scatter_matrix(dataset, alpha=0.2, figsize=(10, 10))
plt.show()
```



```
#Розбиття даних на навчальну та тестову вибірки (80 та 20%)
train_set, test_set = train_test_split(dataset, test_size=0.2, random_state=42)
#Відокремлення міток класу для навчальної та тестової вибірок
train_set_labels = train_set["Outcome"].copy()
train_set = train_set.drop("Outcome", axis=1)
test_set_labels = test_set["Outcome"].copy()
test_set = test_set.drop("Outcome", axis=1)

#Масштабування даних
scaler = Scaler()
scaler.fit(train_set)
train_set_scaled = scaler.transform(train_set)
test_set_scaled = scaler.transform(test_set)

# Створюємо масив з назвами класифікаторів, які застосуємо
array_of_models = []
#1
array_of_models.append(('SVC', SVC()))
#2
array_of_models.append(('LSVC', LinearSVC()))
#3
array_of_models.append(('KNN', KNeighborsClassifier()))
#4
array_of_models.append(('LR', LogisticRegression()))
#5
array_of_models.append(('NB', GaussianNB()))
#6
array_of_models.append(('RFC', RandomForestClassifier()))
#7
array_of_models.append(('DTR', DecisionTreeRegressor()))
#8
array_of_models.append(('GBC', GradientBoostingClassifier()))
#9
array_of_models.append(('MLP', MLPClassifier()))

# Увага! Для простоти коду всі класифікатори перебираються у циклі
Але параметри їх не прописані і беруться за замовченням. Вони не будуть кращими для цієї задачі!
```

```
results = []
names = []
X = train_set_scaled
Y = train_set_labels
for name, model in array_of_models:

#Кросвалідація. Розбиваємо дані на 5 частин
   kfold = model_selection.KFold(n_splits=5, random_state=32)
   cv_results = model_selection.cross_val_score(model, X, Y, cv=kfold,
scoring='accuracy')
   results.append(cv_results)
   names.append(name)
   res = "%s: %f" % (name, cv_results.mean())
   print(res)
```

```
#Результат оцінювання точності прогнозування на тестовому
наборі

SVC: 0.752419
LSVC: 0.760576
KNN: 0.736199
LR: 0.749154
NB: 0.737745
RFC: 0.737785
DTR: 0.701906
MLP: 0.768733
```

#Кожен з класифікаторів має свої налаштування, від яких залежить точність класифікації на певному наборі даних. Гратковий пошук ***GridSearchCV*** *допомогає задати різні значення параметрів, прорахувати всі варіанти і повернути найкращий набір параметрів*

#Приклад граткового пошуку для SVM

```
   param_grid = {
   'C': [1.0, 2.0, 5.0, 10.0, 25.0, 50.0],
   'kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
   'shrinking': [True, False],
   'gamma': ['auto', 1, 0.7, 0.5, 0.1],
   'coef0': [0.0, 0.1, 0.2, 0.3, 0.4, 0.5]}
model_svc = SVC()
grid_search=GridSearchCV(model_svc, param_grid, cv=5, scoring='accuracy')
grid_search.fit(train_set_scaled, train_set_labels)
grid_search.best_score_
```

Результат: 0.7703583061889251

grid_search.best_params_

Результат: {'C': 10.0, 'coef0': 0.5, 'gamma': 0.5, 'kernel': 'poly', 'shrinking': True}

```
#Застосуємо тепер найкращі параметри для навчання моделі на всіх даних
svc = grid_search.best_estimator_
X = np.append(train_set_scaled, test_set_scaled, axis=0)
Y = np.append(train_set_labels, test_set_labels, axis=0)
svc.fit(X, Y)
#Нова пацієнтка – створюємо масив з її показниками Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age, Outcome
#масштабуємо ці дані та зроблю прогноз
patient_undifined = pd.DataFrame([[2, 110, 85, 23, 0, 36, 0.4, 24]])
patient_undifined_scaled = scaler.transform(patient_undifined)
prediction = svc.predict(patient_undifined_scaled)
print(prediction)
```

Результат – [0] – з вірогідністю 0.77 діабета не має

ЛІТЕРАТУРА

1. Андреас Мюллер, Сара Гвидо. Введение в машинное обучение с помощью Python, 2017, 393 стр.
2. Duda R. O. Pattern Classification / R. O. Duda, P. E. Hart, D. G. Stork. — Wiley, 2001. — 654 p.
3. Marques de Sá J. P. Pattern recognition: concepts, methods, and applications / Marques de Sá J. P. — Springer, 2001. — 318 p.
4. P. Kim. MATLAB Deep Learning: With Machine Learning, Neural Networks and Artificial Intelligence. Apress. 151 p.
5. M. Swamynathan, Mastering Machine Learning with Python in Six Steps, DOI 10.1007/978-1-4842-2866-1
6. M. Paluszek and S. Thomas, MATLAB Machine Learning, DOI 10.1007/978-1-4842-2250-8
7. T. Hastie et al., The Elements of Statistical Learning, Second Edition, Springer Science+Business Media, 2009, 745 p.
8. David G. Kleinbaum, Mitchel Klein. Logistic Regression. Springer Science+Business Media, 2010, 702 p.
9. Nikhil Ketkar. Deep Learning with Python: A Hands-on Introduction. Apres, 2017, 226 p.
10. Aurélien Géron. Hands-On Machine Learning with ScikitLearn and TensorFlow. O'Reilly Media, 2017, 751 p.
11. Аллен Б.Дауни. Think DSP. Цифровая обработка сигналов на Python, 2017, 160 стр.