МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра системного аналізу та управління

МАШИННЕ НАВЧАННЯ

Методичні рекомендації до виконання практичних робіт
для здобувачів ступеня магістра
освітньо-професійної програми «Системний аналіз»
зі спеціальності 124 Системний аналіз

Дніпро
НТУ «ДП»
2024

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Машинне навчання [Електронний ресурс] : методичні рекомендації до виконання практичних робіт для здобувачів ступеня магістра освітньо-професійної програми «Системний аналіз» зі спеціальності 124 Системний аналіз / уклад.: Т.А. Желдак, О.Б. Владико, А.В. Малієнко, Д.М. Гаранжа ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2024. – 48 с.

Укладачі:
Т.А. Желдак, канд. техн. наук, доц., зав. кафедри;
О.Б. Владико, канд. техн. наук, доц.;
А.В. Малієнко, канд. техн. наук, доц.;
Д.М. Гаранжа, ст. викл.

Затверджено науково-методичною комісією зі спеціальності 124 Системний аналіз (протокол № 3 від 10.05.2024) за поданням кафедри системного аналізу та управління (протокол № 5 від 10.05.2024).

Уміщено тематику практичних робіт, перелік основних теоретичних питань, короткі теоретичні відомості. Описано методики обчислень і наведено схеми та приклади розв’язування типових задач. Подано методичні поради до виконання індивідуальних завдань. Сформульовано питання для самоконтролю й критерії оцінювання індивідуальних робіт. Рекомендації орієнтовано на активізацію виконавчого етапу навчальної діяльності студентів.

Відповідальний за випуск завідувач кафедри системного аналізу та управління, канд. техн. наук, доц. Т.А. Желдак.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Зміст

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Вступ

Машинне навчання – це один з методів функціонування штучного інтелекту, що вивчає методи побудови алгоритмів, здатних навчатися.

Мета дисципліни – сформувати у здобувачів навички застосування та використання сучасних технологій обробки даних для вирішення задач класифікації, регресійного аналізу, прогнозування та ухвалення рішень.

Очікувані результати навчання за дисципліною:

– розробляти та застосовувати методи, алгоритми та інструменти для навчання нейронних мереж;

– застосовувати знання машинного навчання з вчителем та без нього в системах підтримки прийняття рішень та повсякденній практиці;

– знати математичну основу, ідею та алгоритми класифікації об’єктів у дійсному, категорійному та змішаному просторах;

– використовувати методи добування знань, кластеризації та класифікації;

– створювати ефективні алгоритми для обчислювальних задач системного аналізу та систем підтримки прийняття рішень;

– застосовувати знання машинного навчання в умовах слабо структурованих даних різної природи;

– використовувати алгоритми побудови асоціативних та класифікуючих правил;

– будувати алгоритми AdaBoost для класифікації та навчання нейронних мереж.

Практичні роботи виконуються кожним студентом за варіантами згідно з порядковим номером у журналі поточного контролю викладача. В практичних роботах наводяться всі необхідні формули з розшифруванням прийнятих позначень, їхні значення і результати обчислень. Всі розрахунки здійснюються у середовищі Python.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Практична робота №1

Тема: **Завдання лінійної бінарної класифікації.**

Мета роботи: у середовищі Python, створити алгоритм бінарної класифікації.

Поставлена мета досягається вирішенням наступних задач:

1. розглянути алгоритми бінарної класифікації.

Очікувані результати навчання:

- ДРН 3. Знати математичну основу, ідею та алгоритми класифікації об’єктів у дійсному, категорійному та змішаному просторах;

Теоретичні відомості

Розв'язання простого завдання бінарної класифікації. Розпізнавання образів (pattern recognition) — це розділ теорії штучного інтелекту (artificial intelligence), що вивчає методи класифікації об’єктів. За традицією об’єкт, що піддається класифікації, називається образом (pattern). Образом може бути цифрова фотографія (розпізнавання зображень), буква або цифра (розпізнавання символів), запис мови (розпізнавання мови) тощо.

Прецедент — це об’єкт, приналежність якого до заданого класу визначена заздалегідь. Прецедентом може бути, наприклад, набір ознак пацієнта із відомим діагнозом, з яким слід порівнювати набір ознак людини, діагноз якої ще невідомий.

Класифікатор, або вирішальне правило (decision rule) — це функція, яка ставить у відповідність вектору ознак образу клас, до якого він належить. Задачу розпізнавання образів можна розділити на ряд підзадач.

1. Генерування ознак (feature generation) — вимірювання або обчислення числових ознак, що характеризують об’єкт.

2. Вибір ознак (feature selection) — визначення найбільш інформативних ознак для класифікації (в цей набір можуть входити не лише первинні ознаки, але й функції від них).

3. Побудова класифікатора (classifier construction) — конструювання вирішального правила, на підставі якого здійснюється класифікація.

4. Оцінка якості класифікації (classifier estimation) — обчислення показників правильності класифікації (точність, чутливість, специфічність, помилки першого та другого роду). Введемо позначення і сформулюємо математичну постановку задачі класифікації.

Простий приклад розв'язання задачі бінарної класифікації образів, що лінійно розділяються. Це дозволить вам краще зрозуміти специфіку вирішення таких завдань.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Розглянемо наступний алгоритм бінарної класифікації:

$$a(x, w) = sign(\langle w, x \rangle) = \begin{cases} -1, x \in C_1 \\ +1, x \in C_2 \end{cases}$$

де $w = [w_0, w_1, \ldots, w_n]^T$ - вектор настроюваних (у процесі навчання) параметрів. Як шукати вектор w - це ключове питання таких завдань. На цьому занятті ми діятимемо банально, прямолінійно, що називається «в лоб» для знаходження значень компонент вектору w. Але спочатку поставимо таке завдання.

Припустимо, ми хочемо створити класифікатор в медицині, який би відрізняв два види клітин які отримані за допомогою метода сканування з довжиною хвилі 575 нм і діаметром зонда 0,05 мкм рис 1. Ці клітини є образами, що піддаються класифікації, а їхня сукупність утворює простір образів, який розбито на два класи: РМЗ і ФАМ. Така класифікація називається бінарною. В результаті були отримані дані табл.1

У підсумку, наша навчальна вибірка складається з наступних спостережень:

Таблиця 1 – Навчальна вибірка.

№	Ширина, нм	Довжина, нм	Клітини	№	Ширина, нм	Довжина, нм	Клітини
1	10	50	ФАМ	6	40	40	РМЗ
2	20	30	РМЗ	7	30	45	РМЗ
3	25	30	РМЗ	8	20	45	ФАМ
4	20	60	ФАМ	9	40	30	РМЗ
5	15	70	ФАМ	10	7	35	ФАМ

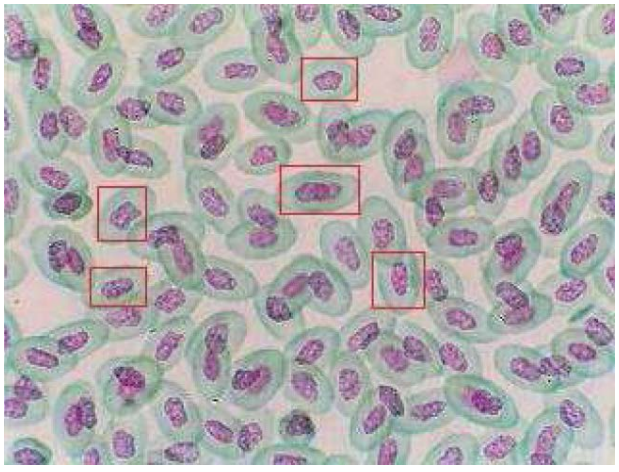

Рис. 1 – Ядра клітин після забарвлення.

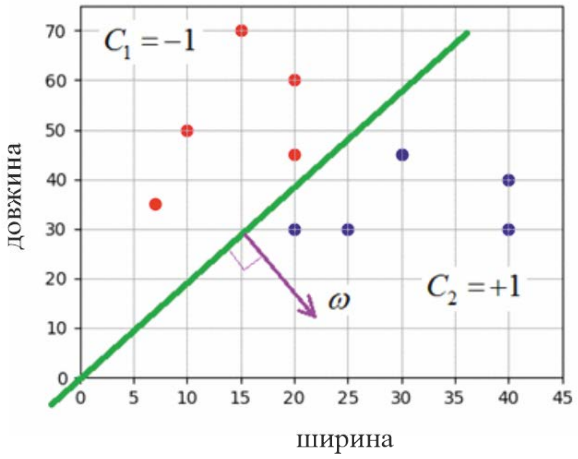

Рис. 2 – Графік розділення образів клітин на площині

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Тут червоними точками відзначені ФАМ, а синіми – РМЗ. Причому, ФАМ буде відповідати цільове значення -1, а РМЗ - +1. Також видно, що навчальна вибірка утворює лінійно-роздільні класи і лінію, що розділяє, можна провести через початок координат.

Тому рівняння лінії шукатимемо у вигляді: $w_1x_1 + w_2x_2 = 0$

Спростимо вираз, перепишемо його, так: $x_2 = -\frac{w_1}{w_2} x_1$

Якщо прийняти w_2 = -1, то залишиться тільки один параметр, що підбирається: $x_2 = w_1x_1$

А загальний вектор: $w = [w_1, -1]^T$

Фактично тут w_1 – кутовий коефіцієнт прямий і нам потрібно його знайти.

Приблизно в середині 1950-х років це завдання вирішував Френк Розенблатт. Він сформулював критерій якості для лінії, що розділяє, як число невірних класифікацій:

$$Q(a, X^l) = \sum_{i=1}^{l} [a(x_i) \neq y_i]$$

Тут квадратні дужки – це індикатор помилки, вони переводять булеві величини *True* і *False* у значення 1 та 0 (нотація Айзерсона):

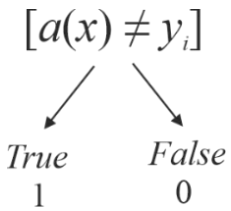

Рис. 3 – Нотація Айзерсона

У разі, якщо відповідь вирішального правила $a(x)$ не збігається з цільовим y, то формується значення *True*, тобто 1.1, навпаки, при збігу, будемо отримувати *False* і значення 0. У результаті функціонал $Q(a, X^l)$ буде підраховувати кількість помилок класифікації.

Однак, при вирішенні завдань бінарної класифікації, коли цільові відповіді вибираються з множини y O {-1, +1}, той самий показник якості можна обчислити і так: $Q(a, X^l) = \sum_{i=1}^{l} [y_i \cdot a(x_i) < 0]$

Тут величина $y_i \cdot a(x_i)$ прийматиме негативні значення при помилкових класифікаціях і позитивні – при правильних. Дійсно, якщо знаки величин y_i , $a(x_i)$ збігаються, отже, ми чітко віднесли образ до потрібного класу і добуток буде позитивним. Інакше знаки будуть відрізнятися, і добуток буде меншим за нуль.

Такий добуток часто використовують при розробці алгоритмів бінарної класифікації. Позначають його великою літерою M: $M_i = a(x_i) \cdot y_i$ і називають відступом (у перекладі з англ. *margin*).

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Ця величина може показувати не тільки ознаку правильної класифікації, а й наскільки далеко відстоює образ від площини, що розділяє. У нашому випадку ми могли б вимірювати цю відстань, якщо прибрати знакову функцію з моделі: $a(x) = \langle w, x \rangle$ і $M_i = \langle w, x \rangle \cdot y_i$.

Але поки що нам це не потрібно і ми залишимо функцію sign(). Головне тут запам'ятати, як обчислюється *margin* та що він означає. Надалі ми його активно використовуватимемо.

Отже, критерій якості для вирішення поставленого завдання сформульовано: $Q(a, X^l) = \sum_{i=1}^{l} [M_i < 0] \rightarrow \min$

Як нам тепер ним скористатися, щоб знайти коефіцієнт w_1?

Вочевидь, що суто математично мінімізувати цей функціонал неможливо, т.к. він являє собою шматково-безперервну функцію, що не диференціюється.

Тому рішення буде алгоритмічним, подібно до того, що запропонував Френк Розенблат. Воно є у вигляді псевдокоду, реалізація мовою Python:

Вхід: вибірка X^l, крок навчання η, максимальна кількість ітерацій N.

Вихід: вектор ваг $w = [w_1, w_2]^T$

1) ініціалізація $w = [0, -1]^T$

2) повторювати N разів

3) по черзі обирати x_i, y_i з навчальної вибірки X^l

4) якщо $M_i = sign(\langle w, x_i \rangle) \cdot y_i < 0$, то

5) коригувати вагу: $w_1 = w_1 + \eta \cdot y_i$

6) обчислюємо показник якості $Q(a, X^l) = \sum_{i=1}^{l} [M_i < 0]$

7) якщо $Q(a, X^l) = 0$, то перериваємо цикл (рішення знайдено)

Приклад. Програма в Python для лінійної бінарної класифікації.

1. Вхідні дані.

```
import numpy as np
import matplotlib.pyplot as plt

x_train = np.array([[10, 50], [20, 30], [25, 30], [20, 60], [15, 70], [40, 40], [30, 45], [20, 45], [40, 30], [7, 35]])
y_train = np.array([-1, 1, 1, -1, -1, 1, 1, -1, 1, -1])
```

2. Основні розрахунки

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

```
n_train = len(x_train)            # розмір навчальної вибірки
w = [0, -1]                       # початкове значення вектору w
a = lambda x: np.sign(x[0]*w[0] + x[1]*w[1])   # вирішальне правило
N = 50                            # максимальна кількість ітерацій
L = 0.1                           # крок зміни ваги
e = 0.1

last_error_index = -1     # індекс останнього помилкового спостереження

for n in range(N):
    for i in range(n_train):      # перебір за спостереженнями
        if y_train[i]*a(x_train[i]) < 0:   # якщо помилка класифікації
            w[0] = w[0] + L * y_train[i]   # то коригування ваги w0
            last_error_index = i

    Q = sum([1 for i in range(n_train) if y_train[i]*a(x_train[i]) < 0])
    if Q == 0:     # показник якості класифікації (число помилок)
        break      # зупинити, якщо все правильно класифікуємо
if last_error_index > -1:
   w[0] = w[0] + e * y_train[last_error_index]
```

3. Виведення результатів

```
print(w)
line_x = list(range(max(x_train[:, 0])))
line_y = [w[0]*x for x in line_x]

x_0 = x_train[y_train == 1]              # формування точок для 1-го
x_1 = x_train[y_train == -1]             # і 2-го класів
plt.scatter(x_0[:, 0], x_0[:, 1], color='red')
plt.scatter(x_1[:, 0], x_1[:, 1], color='blue')
plt.plot(line_x, line_y, color='green')
plt.xlim([0, 45])
plt.ylim([0, 75])
plt.ylabel("довжина")
plt.xlabel("ширина")
plt.grid(True)
plt.show()
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Завдання.

У середовищі Python, вирішити завдання лінійної бінарної класифікації для свого варіанту. Побудувати графіки, пояснити отримані результати.

Варіанти до практичної роботи

Таблиця 1.

Номер варіанту	**X_1\\Y_1**		**X_2\\Y_2**		**X_3\\Y_3**		**X_4\\Y_4**		**X_5\\Y_5**	
1	4	35	22	4	1	60	11	45	4	19
2	12	15	13	27	15	25	7	16	14	30
3	5	49	2	22	20	26	15	17	9	12
4	8	33	13	15	2	12	12	19	20	23
5	4	23	20	5	14	17	3	27	3	32
6	12	19	12	6	15	22	7	26	12	31
7	18	28	6	9	4	29	4	3	21	26
8	8	32	17	15	1	27	11	13	21	19
9	12	24	16	7	5	27	12	26	16	28
10	17	34	9	16	24	9	11	12	3	11
11	4	21	15	1	24	12	15	6	3	28
12	19	12	19	8	9	10	4	10	13	7
13	13	34	9	25	5	27	10	14	20	35
14	3	26	13	33	12	30	11	8	5	31
15	15	28	4	7	12	23	15	8	9	32
16	9	22	22	31	6	8	20	7	9	3
17	10	15	18	9	20	7	8	30	19	2
18	19	33	11	16	15	6	2	31	9	25
19	6	11	15	7	2	5	12	23	13	6
20	19	4	23	3	25	29	8	19	12	15
21	11	28	6	30	1	29	11	22	10	6
22	5	9	13	21	12	20	1	19	7	14
23	5	31	6	22	2	24	17	26	19	11
24	6	5	16	24	24	22	18	10	9	14
25	18	3	8	32	18	13	8	31	11	31
26	20	3	17	11	2	25	18	8	9	32
19	6	11	15	7	2	5	12	23	13	6
20	19	4	23	3	25	29	8	19	12	15

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Номер варіанту	**X_1\\Y_1**		**X_2\\Y_2**		**X_3\\Y_3**		**X_4\\Y_4**		**X_5\\Y_5**	
21	11	28	6	30	1	29	11	22	10	6
22	5	9	13	21	12	20	1	19	7	14
23	5	31	6	22	2	24	17	26	19	11
24	6	5	16	24	24	22	18	10	9	14
25	18	3	8	32	18	13	8	31	11	31
26	20	3	17	11	2	25	18	8	9	32
27	9	1	24	4	14	22	14	25	19	33
28	9	8	9	7	24	11	1	15	3	32
29	15	15	2	13	8	22	18	2	15	31
30	15	21	4	16	0	19	5	28	10	23

Продовження таблиці 1.

Номер варіанту	**X_6\\Y_6**		**X_7\\Y_7**		**X_8\\Y_8**		**X_9\\Y_9**		**X_{10}\\Y_{10}**	
1	23	10	13	18	4	46	15	35	4	64
2	10	48	25	34	13	28	15	19	12	27
3	24	63	8	14	22	34	6	11	20	22
4	7	28	6	8	13	34	12	26	4	15
5	5	30	14	2	17	1	12	21	11	5
6	21	20	2	31	5	67	17	33	22	6
7	15	31	8	65	6	29	0	7	6	6
8	20	18	0	16	18	31	9	1	15	15
9	28	6	19	11	12	25	16	2	45	7
10	15	8	7	28	8	27	8	12	14	31
11	17	22	17	24	7	4	2	12	13	1
12	1	8	13	16	20	4	5	1	18	8
13	22	30	4	12	7	4	6	22	5	25
14	8	34	15	16	3	4	8	10	16	33
15	6	27	3	32	15	5	4	12	12	67
16	3	27	20	20	12	29	3	31	2	31
17	20	7	10	4	2	23	12	17	17	9
18	11	12	9	22	10	19	13	25	8	16
19	20	22	18	9	13	19	0	31	16	7
20	4	5	10	11	8	22	11	6	7	3

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Номер варіанту	**X_6\\Y_6**		**X_7\\Y_7**		**X_8\\Y_8**		**X_9\\Y_9**		**X_{10}\\Y_{10}**	
21	20	11	11	25	2	24	10	5	21	30
22	22	17	20	32	9	11	1	30	15	21
23	6	10	22	18	16	13	13	14	1	22
24	8	12	3	15	8	26	1	0	0	24
25	5	30	6	35	14	8	3	25	4	32
26	19	4	6	5	14	12	4	33	21	16
27	12	30	10	16	2	32	12	8	13	4
28	3	3	4	35	8	7	6	22	13	67
29	20	2	3	12	4	6	8	28	13	13
30	18	5	22	12	12	34	7	20	6	16

Таблиця 2.

Номер варіанту	1	2	3	4	5	6	7	8	9	10
1	-1	1	1	-1	-1	1	1	-1	1	-1
2	1	-1	1	1	-1	1	1	1	-1	1
3	1	1	-1	1	1	-1	1	1	1	-1
4	-1	1	1	-1	1	1	-1	1	1	1
5	-1	-1	1	1	-1	1	1	-1	1	1
6	1	-1	-1	1	1	-1	1	1	-1	1
7	1	1	-1	-1	1	1	-1	1	1	-1
8	-1	1	1	-1	-1	1	1	-1	1	1
9	-1	-1	1	1	-1	-1	1	1	-1	1
10	1	-1	-1	1	1	-1	-1	1	1	-1
11	-1	1	-1	-1	1	1	-1	-1	1	1
12	1	-1	1	-1	-1	1	1	-1	-1	1
13	1	1	-1	1	-1	-1	1	1	-1	-1
14	-1	1	1	-1	1	-1	-1	1	1	-1
15	-1	1	1	1	-1	1	-1	-1	1	1
16	1	-1	1	1	1	-1	1	-1	-1	1
17	1	1	-1	1	1	1	-1	1	-1	-1
18	-1	1	1	-1	1	1	1	-1	1	-1
19	1	-1	1	1	-1	1	1	1	-1	1
20	-1	1	-1	1	1	-1	1	1	1	-1

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Номер варіанту	1	2	3	4	5	6	7	8	9	10
21	1	-1	1	-1	1	1	-1	1	1	1
22	-1	1	-1	1	-1	1	1	-1	1	1
23	1	-1	1	-1	1	-1	1	1	-1	1
24	-1	1	-1	1	-1	1	-1	1	1	-1
25	-1	1	1	-1	1	-1	1	-1	1	1
26	1	-1	1	1	-1	1	-1	1	-1	1
27	1	1	-1	1	1	1	1	-1	1	-1
28	-1	1	1	-1	1	-1	1	1	-1	1
29	1	-1	1	1	1	1	-1	1	1	-1
30	1	1	-1	1	-1	1	1	-1	1	1

Додаткове завдання.

У середовищі Python, створити алгоритм байєсовського класифікатора Бернулі. Побудувати графіки, пояснити отримані результати.

Джерела для самостійного опрацювання.

1. Аналіз алгоритмів машинного навчання: https://ami-ejournal.cdu.edu.ua/article/view/3707/3987.
2. обробка сигналів в біомедичних електронних системах: https://ela.kpi.ua/server/api/core/bitstreams/b1880b6c-1111-4f9f-b6a7-daec958a0a32/content.
3. https://www.guru99.com/uk/machine-learning-tutorial.html.

Питання для самоконтролю

1. Назвіть задачі класифікації.
2. Які види класифікацій ви знаєте?
3. У чому полягає ідея лінійного пошуку? Сформулюйте умову припинення перегляду елементів.
4. Які існують алгоритми машинного навчання? У чому їх відмінність?
5. Сформулюйте алгоритм бінарного пошуку.
6. Як проходить оцінка якості кластеризації?
5. На які підзадачі розбивається задача класифікації?
6. У якому випадку може використовуватися бінарний пошук? У чому його основна ідея?
7. Що таке чутливість, специфічність і точність бінарної класифікації?

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Практична робота № 2

Тема: **Реалізація гаусівського байєсівського класифікатора**

Мета: у середовищі Python, створити та вирішити алгоритм гаусівського байєсовського класифікатора.

1. розглянути алгоритми гаусівського байєсівського класифікатора.

Очікувані результати навчання:

- ДРН 1. Розробляти та застосовувати методи, алгоритми та інструменти для навчання нейронних мереж.

Теоретичні відомості. Наївний метод Байєса – це набір методів класифікації, заснованих на теоремі Байєса. **Теорема Байєса** – одна з основних теорем елементарної теорії ймовірностей, яка дозволяє визначити ймовірність якої-небудь події за умови, що сталася інша статистично взаємозалежна з нею подія. Іншими словами, за формулою Байєса можна більш точно перерахувати ймовірність, взявши до уваги раніше відому інформацію і дані нових спостережень.

Формула Байєса: $P(A|B) = \frac{P(B|A)\times \mathrm{P}(A)}{\mathrm{P}(B)}$,

де, $P(A)$ – апріорна ймовірність гіпотези A; $P(A|B)$ – ймовірність гіпотези A при настанні події B (апостеріорна ймовірність); $P(B|A)$ – ймовірність настання події B при істинності гіпотези A; $P(B)$ – ймовірність настання події B.

Байєсовський класифікатор – широкий клас алгоритмів класифікації, заснований на принципі максимуму апостеріорної ймовірності. Для класифікованого об’єкта обчислюються функції правдоподібності кожного з класів, за ними обчислюються апостеріорні ймовірності класів.

Властивості наївної класифікації:

1. Використання всіх змінних і визначення всіх залежностей між ними.

2. Наявність двох припущень щодо змінних:

- усі змінні є однаково важливими;

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

Хоча наївний байєсовський класифікатор є хорошим класифікатором, значення прогнозованих ймовірностей не завжди досить точні.

Тому на практиці врахувати підступні зв'язки між ознаками і побудувати не наївний, а повноцінний класифікатор Байєса?

У разі багатовимірного розподілу гаусівського це зробити відносно просто.

Як ви вже здогадалися, для цього нам достатньо по навчальній вибірці побудувати оцінки математичного очікування та коваріаційних матриць окремо

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

для кожного класу:

$\hat{\mu}_y, \hat{\Sigma}_y, y \in Y$

Формули для обчислення цих оцінок прості і загалом очевидні:

$$\hat{\mu}_y = \frac{1}{l_y}\sum_{i:y_i=y} x_i$$

$$\Sigma_y = \frac{1}{l_y}\sum_{i:y_i=y}(x_i - \hat{\mu}_y)\cdot(x_i - \hat{\mu}_y)^T$$

Існує теорема, яка доводить, що ці оцінки відповідають оцінкам максимальної правдоподібності для n-вимірних гаусівських розподілів. Тобто ці оцінки адекватні і, загалом, нічого кращого тут ми придумати не можемо.

Звичайно, щоб можна було довіряти цим оцінкам, кількість об'єктів кожного класу має бути якнайбільше. Інакше просто не зберемо достатньої статистики. Вважається, що мінімум це 100 об'єктів одного конкретного класу. Але, бажано мати від 1000 і більше.

Щоб усе це краще зрозуміти, давайте застосуємо цей підхід (гаусівський байєсівський класифікатор) до змодельованого завдання бінарної класифікації, в

якій навчальна вибірка згенерована на основі нормального двомірного розподілу, з наступними параметрами:

$$r_1 = 0{,}8;\ \sigma_{x1}^2 = 1{,}0;\ \mu_{y1} = [0, -3]^T;\ \sum_{y1} = \begin{pmatrix} \sigma_{x1}^2 & \sigma_{x1}^2 \cdot r_1 \\ \sigma_{x1}^2 \cdot r_1 & \sigma_{x1}^2 \end{pmatrix}$$

$$r_2 = 0{,}7;\ \sigma_{x2}^2 = 2{,}0;\ \mu_{y2} = [0, -3]^T;\ \sum_{y1} = \begin{pmatrix} \sigma_{x2}^2 & \sigma_{x2}^2 \cdot r_2 \\ \sigma_{x2}^2 \cdot r_2 & \sigma_{x2}^2 \end{pmatrix}$$

Усього ми генеруємо за $N = 1000$ образів вибірки кожного класу.

За цими даними обчислюємо оцінки МО та коваріаційних матриць: $\hat{\mu}_y, \hat{\Sigma}_y, y$ O Y.

А потім застосовуємо алгоритм гаусівського байсівського класифікатора:

$$a(x) = \arg\max_{y \in Y}\left(\ln\left(\lambda_y \hat{P}(y)\right) - \frac{1}{2}(x - \hat{\mu}_y)^T \cdot \hat{\Sigma}_y^{-1}(x - \hat{\mu}_y) - \frac{1}{2}\ln \det\hat{\Sigma}_y\right)$$

(Тут від умовної багатовимірної ПРВ було взято натуральний логарифм).

Програма на Python. Приклад реалізації наївного гаусівського байєсівського класифікатора

1. Вхідні дані.

```
import numpy as np
import matplotlib.pyplot as plt
x_train = np.array([[10, 50], [20, 30], [25, 30], [20, 60], [15, 70], [40, 40], [30, 45], [20, 45], [40, 30], [7, 35]])
y_train = np.array([-1, 1, 1, -1, -1, 1, 1, -1, 1, -1])
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

2. Основні розрахунки

```
mw1, ml1 = np.mean(x_train[y_train == 1], axis=0)
mw_1, ml_1 = np.mean(x_train[y_train == -1], axis=0)

# формула для обчислення дисперсії тут трохи інша 1/N*sum(...)
sw1, sl1 = np.var(x_train[y_train == 1], axis=0)
sw_1, sl_1 = np.var(x_train[y_train == -1], axis=0)
print('МО: ', mw1, ml1, mw_1, ml_1)
print('Дисперсії:', sw1, sl1, sw_1, sl_1)
x = [40, 10]  # довжина, ширина
a_1 = lambda x: -(x[0] - ml_1) ** 2 / (2 * sl_1) - (x[1] - mw_1) ** 2 / (2 * sw_1)
a1 = lambda x: -(x[0] - ml1) ** 2 / (2 * sl1) - (x[1] - mw1) ** 2 / (2 * sw1)
y = np.argmax([a_1(x), a1(x)])
```

3. Виведення результатів

```
print('Номер класу (0 - ФАМ, 1 - РМЗ): ', y)
```

Приклад 2. Гаусівський байесовский класифікатор.

1. Вхідні дані.

```
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(1)
# вихідні параметри розподілу двох класів
r1 = 0.8
D1 = 1.0
mean1 = [0, -3]
V1 = [[D1, D1 * r1], [D1 * r1, D1]]
r2 = 0.7
D2 = 2.0
mean2 = [0, 3]
 V2 = [[D2, D2 * r2], [D2 * r2, D2]]
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

2. Основні розрахунки.

```
# моделювання навчальної вибірки
N = 1000
x1 = np.random.multivariate_normal(mean1, V1, N).T
x2 = np.random.multivariate_normal(mean2, V2, N).T
# обчислення оцінок МО та коваріаційних матриць
mm1 = np.mean(x1.T, axis=0)
mm2 = np.mean(x2.T, axis=0)
a = (x2.T - mm2).T
VV2 = np.array([[np.dot(a[0], a[0]) / N, np.dot(a[0], a[1]) / N],
            [np.dot(a[1], a[0]) / N, np.dot(a[1], a[1]) / N]])

# модель гаусівського байєсівського класифікатора
Py1, L1 = 0.5, 1  # ймовірність появи класів
Py2, L2 = 1 - Py1, 1  # та величини штрафів неправильної класифікації
b = lambda x, v, m, l, py: np.log(l * py) - 0.5 * (x - m) @ np.linalg.inv(v) @ (x -
m).T - 0.5 * np.log(np.linalg.det(v))
x = np.array([0, -4])  # вхідний вектор у форматі (x, y)
a = np.argmax([b(x, VV1, mm1, L1, Py1), b(x, VV2, mm2, L2, Py2)])
```

3. Виведення результатів

```
print(a)

plt.figure(figsize=(4, 4)) # виведення графіків
plt.title(f"Кореляції: r1 = {r1}, r2 = {r2}")
plt.scatter(x1[0], x1[1], s=10)
plt.scatter(x2[0], x2[1], s=10)
plt.show()
```

Завдання.

У середовищі Python, вирішити завдання з використанням алгоритму гаусівського байєсовського класифікатора для свого варіанту. Побудувати графіки, пояснити отримані результати.

Завдання для прикладу 1 взяти з лабораторної роботи 1.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Завдання для прикладу 2.

Номер варіанту	N, циклів	r1	D1	r2	D2
1	1000	0,8	1,0	0,7	2,0
2	1100	0,7	1,5	0,8	2,5
3	1200	0,6	1,8	0,5	3,0
4	1300	0,8	2,0	0,6	1,0
5	1400	0,6	1,0	0,6	2,0
6	1500	0,7	1,5	0,6	3,0
7	1600	0,8	1,8	0,5	2,5
8	1700	0,5	2,0	0,7	2,3
9	1800	0,8	1,0	0,7	2,0
10	1900	0,7	1,5	0,8	2,5
11	2000	0,6	1,8	0,5	3,0
12	2100	0,8	2,0	0,6	1,0
13	2200	0,6	1,0	0,6	2,0
14	2300	0,7	1,5	0,6	3,0
15	2400	0,8	1,8	0,5	2,5
16	2500	0,5	2,0	0,7	2,3
17	2600	0,7	1,5	0,6	3,0
18	2700	0,8	1,8	0,5	2,5
19	2800	0,5	2,0	0,7	2,3
20	2900	0,8	1,0	0,7	2,0
21	3000	0,8	1,8	0,5	2,5
22	3100	0,7	1,5	0,6	3,0
23	3200	0,8	1,8	0,5	2,5
24	3300	0,5	2,0	0,7	2,3
25	3400	0,8	1,0	0,7	2,0
26	3500	0,7	1,5	0,8	2,5
27	3600	0,6	1,8	0,5	3,0
28	3700	0,8	2,0	0,6	1,0
29	3800	0,6	1,0	0,6	2,0
30	3900	0,8	1,8	0,5	2,5

Додаткове завдання.

У середовищі Python, створити алгоритм лінійного дискримінанта Фішера. Побудувати графіки, пояснити отримані результати.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Джерела для самостійного опрацювання.

1. Інтелектуальні системи:
https://nam.kyiv.ua/files/publications/nester-kovt-fal-2-ostanna.pdf
2. https://www.guru99.com/ru/naive-bayes-classifiers.html
3. Інтелектуальні технології:
https://pdf.lib.vntu.edu.ua/books/2023/Bykov_P1_2023_229.pdf

Питання для самоконтролю

1. У чому сенс формули Байєса і що вона нам справді дає?
2. Які існують алгоритми наївної байєсівської класифікації?
3. Опишіть оптимальний байєсовський класифікатор?
4. Опишіть гаусівський байєсівський класифікатор? У чому його відмінність від других класифікаторів?
5. Опишіть алгоритм роботи гаусівського байєсівського класифікатору

Практична робота № 3

Тема: Апроксимація даних ядерним згладжуванням

Мета: у середовищі Python, провести апроксимацію даних ядерним згладжуванням.

Поставлена мета досягається послідовним вирішенням наступних задач:

Провести апроксимацію ядерним згладжуванням вхідних даних.

Очікувані результати навчання:

- ДРН 6. Застосовувати знання машинного навчання в умовах слабо структурованих даних різної природи.

Теоретичні відомості. Метричні методи – це методи, що ґрунтуються на відстанях (заходи близькості) конкретного образу *x* до образів інших класів у просторі ознак. Наприклад, у нас є результат виміру (ромб). Тоді було б логічно віднести його до того виду (групи точок), до якого він найближчий:

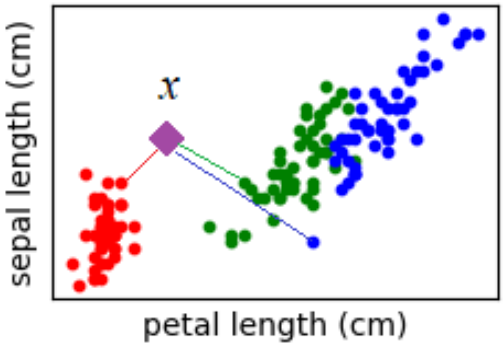

Рис. 4 Відстань *x* до трьох класів.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Але тут одразу виникають дві проблеми:

- як вимірювати відстані між об'єктами у просторі ознак;
- як визначати на основі обчислених відстаней належність об'єкта до тієї чи іншої групи точок.

На ці питання немає єдиної, загальної відповіді. Тому існує безліч різних метричних методів класифікації.

Метрики у просторі ознак. Почнемо з способу визначення відстані між двома будь-якими векторами в n-мірному просторі ознак $\mathbb{R}^n$. Припустимо, ми маємо вектори об'єктів розмічених даних:

$x_i = [x_i^1, x_i^2, \ldots, x_i^n]^T$, $i = 1, 2, \ldots, l$

та нові результати вимірювань, які слід класифікувати:

$x_i = [x^1, x^2, \ldots, x^n]^T$

Тоді, перше, що спадає на думку, взяти евклідову відстань між двома точками:

$$\rho(x, x_i) = \sqrt{\sum_{j=1}^{n}\left(x^j - x_i^j\right)^2} = \left(\sum_{j=1}^{n}\left|x^j - x_i^j\right|^2\right)^{1/2}$$

І така метрика справді застосовується на практиці.

У найпростішому випадку ми можемо узагальнити евклідову метрику, таким чином:

$$\rho(x, x_i) = \left(\sum_{j=1}^{n} w_j \cdot \left|x^j - x_i^j\right|^p\right)^{1/p}$$

Тут $p > 0$ - параметр метрики; $w_1, w_2, \ldots, w_n$ - ваги ознак.

Такий загальний спосіб обчислення відстаней називається метрикою Мінковського.

Навіщо потрібні ваги у цій формулі? Справа в тому, що ознаки можуть мати різний масштаб. Наприклад, вага золота в грамах і його вартість у рублях – це два абсолютно різні діапазони значень. Щоб їх однаково враховувати щодо близькості векторів, слід нормувати через підбір вагових коефіцієнтів.

Інший параметр p задає поведінку метрики у різних напрямках. Це найпростіше показати на рисунках еквідистантних кривих (рис. 5):

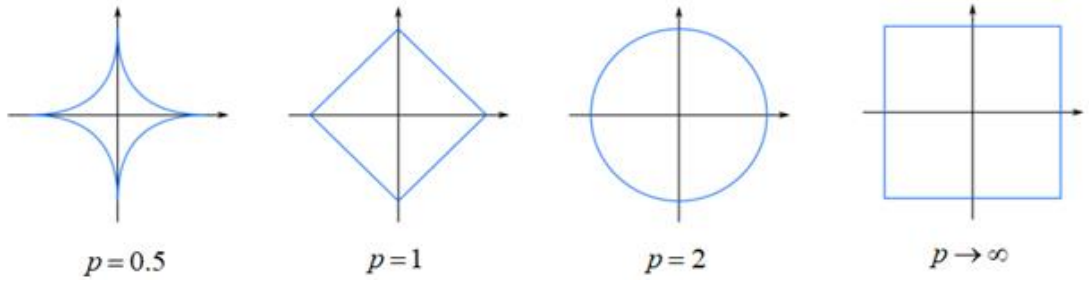

Рис. 5 – Еквідистантні криві

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Як бачимо, при $p = 1$ маємо аналог суми різниць модулів координат, при $p = 2$ – класична евклідова відстань, а якщо $p \to \infty$, то коло вироджується у квадрат. Найчастіше практично можна побачити метрику з модулями і квадратами (при $p = 1$ і 2).

Узагальнений метричний класифікатор. Після того, як введена метрика, ми можемо будувати алгоритм класифікації на основі критерію близькості деякого об'єкта $x \in X$ до розмічених образів $\{x_i\}$ (її можна сприймати як навчальну вибірку). Для цього зручно усі об'єкти вибірки впорядкувати за зростанням відстані щодо x спільно з цільовими значеннями:

$\rho(x, x^{(1)}) \le \rho(x, x^{(2)}) \le \ldots \le \rho(x, x^{(l)})$

$y^{(1)}, y^{(2)}, \ldots, y^{(l)}$

(Верхній індекс - новий порядок спостережень у вибірці). Тоді математично класифікатор у найзагальнішому вигляді можна записати, таким чином:

$$a(x, X^l) = \arg\max_{y \in Y} \sum_{i=1}^{l} [y^{(i)} = y] \cdot w(i, x)$$

де $w(i, x)$ - вага (ступінь важливості) i-го об'єкта по відношенню до x, пов'язаний з метрикою $\rho(x_i, x)$. Величина $w(i, x)$ невід'ємна та не зростає при збільшенні i.

Фактично, тут, ми обчислюємо загальну вагу $\sum_i w(i, x)$ для кожного класу та вибираємо клас із найбільшою сумою.

Методи парзенівського вікна та потенційних функцій. Ми з вами розглянули найпростіший алгоритм метричної класифікації найближчих сусідів. І відзначили суттєву нестачу його вагових коефіцієнтів $w(i, x)$.

Вони приймають лише два значення:

$$w(i, x) = [i \le k] = \begin{cases} 1, i \le k \\ 0, i > k \end{cases}$$

Я нагадаю, що індекс i означає i-й образ у впорядкованій за зростанням відстаней послідовності щодо вектору, що класифікується x:

$\rho(x, x^{(1)}) \le \rho(x, x^{(2)}) \le \ldots \le \rho(x, x^{(l)})$

$y^{(1)}, y^{(2)}, \ldots, y^{(l)}$

Тобто формула $w(i, x) = [i \le k]$ просто виділяє k найближчих сусідів для вектору x. Але, мабуть, було б логічно враховувати сусідні об'єкти з урахуванням відстані $\rho(x, x^{(i)})$ до них: чим вона більша, тим менша вага повинна мати дане спостереження.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

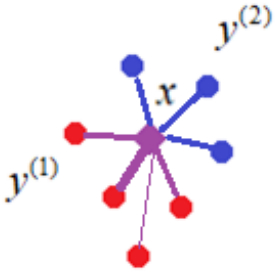

Рис. 6 – Відстані до кластерів

(Близькі сусіди несуть більший внесок, ніж далекі). Для реалізації цієї ідеї ваги $w(i, x)$ пропонується вибирати з використанням деякої функції, що не зростає $K(r)$ (її ще називають ядром): $w(i, x) = K\left(\frac{\rho(x, x^{(i)})}{h}\right)$

Причому часто її вибирають обмеженою (фінітною) на інтервалі [0; 1] та симетричною щодо нуля (парної): $K(r) \neq 0, \quad r \in [0;1]$

Такі ядра отримали назву **вікно Парзена чи парзенівські вікна**. Для вікон параметр h у формулі ваги визначає область охоплення об'єктів щодо вектору x (з центром у точці x). Наприклад, якщо $h = 1$, то максимальна ненульова відстань дорівнюватиме одиниці. Якщо h взяти рівним 4 то відстань (охоплення) також збільшиться в 4 рази. Залишається питання, яку функцію ядра взяти? Часто на практиці використовують такі варіанти:

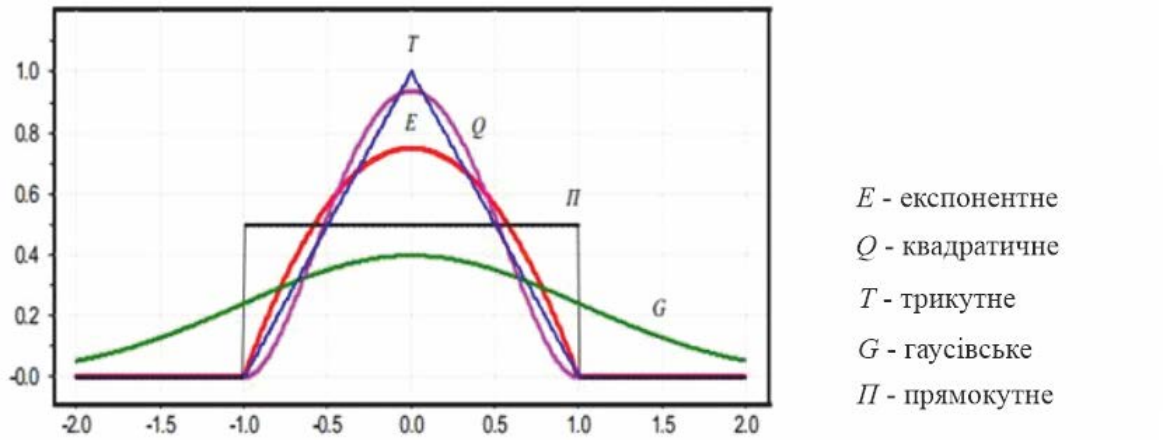

Рис. 7 – Види парзенівськіх вікон

В результаті ми отримуємо наступний алгоритм класифікації за допомогою парзенівського вікна:

$$a(x^{-}, X^{l}, h, K) = \arg\max_{y \in Y} \sum_{i=1}^{l} [y_i = y] \cdot K\left(\frac{\rho(x, x_i)}{h}\right)$$

Тут $[| r | \leq 1]$ - індикатор того, що $r \leq 1$ тобто повертається 1, якщо це так і 0 – в іншому випадку.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Тут ми можемо не сортувати спостереження за відстанями, т.к. функція відстані використовується у самому ядрі.

Давайте припустимо, що ми маємо деякий набір розмічених даних $\{x_i, y_i\}_{i=1}^{l}$, пов'язаних деякою залежністю (не обов'язково строго функціональною): $y_i = f(x_i)$, $i = 1, 2, \ldots, l$. У таких завданнях модель $a(x)$ має вловити природу взаємозв'язку між входами x_i та виходами y_i. Раніше ми з вами вирішували це завдання шляхом параметричної оптимізації.

Таблиця 2 – Формули ядра

Ядро $K(r)$	**Формула**
експонентне	$E(r) = (3/4)\,(1 - r^2) \cdot [\,\lvert r \rvert \le 1]$
квадратичне	$Q(r) = (15/16)\,(1 - r^2)^2 \cdot [\,\lvert r \rvert \le 1]$
трикутне	$T(r) = (1 - \lvert r \rvert) \cdot [\,\lvert r \rvert \le 1]$
гаусівське	$G(r) = \frac{1}{\sqrt{2\pi}} exp\left(-\frac{r^2}{2}\right)$
прямокутне	$П(r) = (1/2) \cdot [\,\lvert r \rvert \le 1]$

Тобто передбачали залежності у вигляді деякої функції: $a\,(x,\,\theta) = g(x,\,\theta)$ та знаходили вектор параметрів θ так, щоб забезпечити мінімум функціоналу якості.

Якщо функціонал має вигляд:

$$Q(a, X^l) = \sum_{i=1}^{l} w_i \cdot (y_i - a(x_i, \theta))^2 \to min_\theta$$

Ваги перед y_1, y_2 можна виразити через трикутне ядро:

$K(r) = |1 - r| \cdot [r \le 1]$

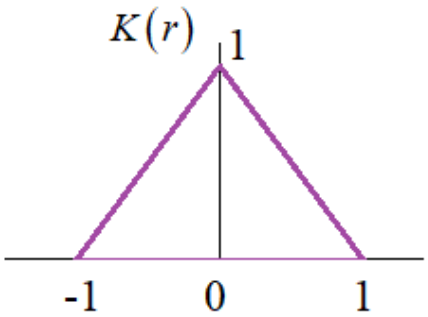

Рис. 8 – Трикутне ядро

Отримаємо:

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

$$y = y_1 \cdot K\left(\frac{\rho(x,x_1)}{h}\right) + y_2 \cdot K\left(\frac{\rho(x,x_2)}{h}\right)$$

Або, узагальнюючи на довільну кількість відліків, маємо:

$$y = a(x, X^l) = \frac{\sum_{i=1}^{l} y_i \cdot K\left(\frac{\rho(x,x_i)}{h}\right)}{\sum_{i=1}^{l} K\left(\frac{\rho(x,x_i)}{h}\right)} = \frac{\sum_{i=1}^{l} y_i \cdot w_i(x)}{\sum_{i=1}^{l} w_i(x)}$$

В останній рівності введено позначення вагових коефіцієнтів:

$$w_i(x) = K\left(\frac{\rho(x,x_i)}{h}\right)$$

Також зверніть увагу, ми тут у формулу додали нормування за вагами, тому що при більшій кількості відліків і різних функцій ядра, нам необхідно зберегти масштаб вихідних значень.

Отримана формула відома під назвою формули **ядерного згладжування Надарая-Ватсона**.

Її можна вивести суворішими математичними методами, наприклад, з критерію мінімуму квадрата помилки, при константній параметричній функції:

$$a(x, X^l) = \sum_{i=1}^{l} w_i(x) \cdot (\alpha - y_i)^2 \rightarrow \min_\alpha$$

Диференціюючи цей вираз по α і прирівнюючи результат нулю, отримаємо:

$$\frac{\partial Q(\alpha; X^l)}{\partial \alpha} = 2 \cdot \sum_{i=1}^{l} w_i(x) \cdot (\alpha - y_i) = 0$$

Звідки

$$\alpha = y = a(x) = \frac{\sum_{i=1}^{l} y_i \cdot w_i(x)}{\sum_{i=1}^{l} w_i(x)}$$

Цей простий висновок показує, який критерій якості мінімізується формулою Надар-Ватсона.

Експериментальні дані тут генеруються за такою формулою:

$y_i = \sin(x_i) + \varepsilon_i, \quad i = 1, 2, \ldots$

де ε_i - гаусівські випадкові величини з нульовим середнім та дисперсією 0,5.

Самі ж величини x_i О [0; 10] із кроком 0,1.

Потім ми визначимо метрику як модуль відстані:

$\rho(x, x_i) = | x - x_i |, \; i = 1, 2, \ldots$

А як ядро візьмемо гаусівську криву:

$K(r) = \exp(-2r^2)$

Як функція ядра $K(r)$ можна використовувати будь-які парзенівські вікна, а метрика вибирається виходячи з знань про експериментальних даних. Це може бути і модуль, як у розглянутій задачі, і евклідова відстань та багато інших відомих метрик.

Програма на Python. Апроксимація даних ядерним згладжуванням

1. Вхідні дані.

```
import numpy as np
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

```
import matplotlib.pyplot as plt
x = np.arange(0, 10, 0.1)         # відлік для вихідного сигналу
x_est = np.arange(0, 10, 0.01) # відлік, де здійснюється відновлення функції
N = len(x)
y_sin = np.sin(x)
y = y_sin + np.random.normal(0, 0.5, N)
```

2. Основні розрахунки

```
# апроксимація ядерним згладжуванням
h = 1.0    # при вікні менше 0.1 для фінітних ядер будуть помилки

K = lambda r: np.exp(-2 * r * r)       # гаусівське ядро
# K = lambda r: np.abs(1 - r) * bool(r <= 1)       # трикутне ядро
# K = lambda r: bool(r <= 1)       # прямокутне ядро

ro = lambda xx, xi: np.abs(xx - xi)    # метрика
w = lambda xx, xi: K(ro(xx, xi) / h)    # ваги
```

3. Виведення результатів

```
plt.figure(figsize=(7, 7))
plot_number = 0
for h in [0.1, 0.3, 1, 10]:
    y_est = []
    for xx in x_est:
        ww = np.array([w(xx, xi) for xi in x])
        yy = np.dot(ww, y) / sum(ww)          # формула Надар-Ватсона
        y_est.append(yy)
    plot_number += 1
    plt.subplot(2, 2, plot_number)

    plt.scatter(x, y, color='black', s=10)
    plt.plot(x, y_sin, color='blue')
    plt.plot(x_est, y_est, color='red')
    plt.title(f"Гауссівське ядро с h = {h}")
    plt.grid()
plt.show()
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Завдання.

У середовищі Python, вирішити завдання з апроксимація даних ядерним згладжуванням для свого варіанту. Побудувати графіки, пояснити отримані результати.

Завдання для прикладу.

Номер варіанту	**Вид ядра**	**Вхідний сигнал**		
1	гаусівське	0	10	0,1
2	трикутне	0	20	0,2
3	прямокутне	0	30	0,3
4	гаусівське	0	40	0,4
5	трикутне	0	50	0,5
6	прямокутне	0	10	0,1
7	гаусівське	0	20	0,2
8	трикутне	0	30	0,3
9	прямокутне	0	40	0,4
10	гаусівське	0	50	0,5
11	трикутне	0	10	0,1
12	прямокутне	0	20	0,2
13	гаусівське	0	30	0,3
14	трикутне	0	40	0,4
15	прямокутне	0	50	0,5
16	гаусівське	0	10	0,1
17	трикутне	0	20	0,2
18	прямокутне	0	30	0,3
19	гаусівське	0	10	0,1

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Номер варіанту	**Вид ядра**	**Вхідний сигнал**		
20	трикутне	0	20	0,2
21	прямокутне	0	30	0,3
22	гаусівське	0	40	0,4
23	трикутне	0	50	0,5
24	прямокутне	0	10	0,1
25	гаусівське	0	20	0,2
26	трикутне	0	30	0,3
27	прямокутне	0	40	0,4
28	гаусівське	0	50	0,5
29	трикутне	0	10	0,1
30	прямокутне	0	20	0,2

Джерела для самостійного опрацювання.

1. Подання векторних даних на основі випадкових проекцій: https://www.researchgate.net/publication/332015426_Distributed_Representation_of_Vector_Data_based_on_Random_Projections

2.https://wiki.tntu.edu.ua/%D0%9D%D0%B5%D0%BF%D0%B0%D1%80%D0%B0%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%BD%D0%B0_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%96%D1%8F

3. Оцінки Надарая-Ватсона для спостережень із суміші: https://probability.knu.ua/tims/issues-new/100/PDF/7.pdf

Питання для самоконтролю

1. Дайте визначення матричним методам.
2. Дайте визначення метрики Мінковського?
3. Дайте визначення вікну Парзена.
4. Дайте визначення формулі ядерного згладжування Надарая-Ватсона.

Практична робота №4

Тема: **Алгоритм кластеризації Ллойда (K-середніх)**

Мета: у середовищі Python, створити алгоритм кластеризації Ллойда.

Поставлена мета досягається послідовним вирішенням наступних задач:

- розрахувати такі параметри, як інформаційний струм, потужність і величина С, що розсіюються на певному елементі;

Очікувані результати навчання:

- ДРН 3. Знати математичну основу, ідею та алгоритми класифікації об'єктів у дійсному, категорійному та змішаному просторах.
- ДРН 4. Використовувати методи добування знань, кластеризації та класифікації.

Порядок виконання роботи

1. Вивчити необхідний теоретичний матеріал та ознайомиться з прикладами.
2. Виконати програму з наведеного прикладу в середовищі Python.
3. Побудуйте графіки, поясніть отримані результати.
4. Скласти звіт про виконання роботи, який повинен містити:

− постановку задачі;
− опис та аналіз ходу розв'язання задачі;
− висновки за результатом виконаної роботи.

Теоретичні відомості.

Ідея методу *k*-середніх полягає у відшуканні такого розбиття набору з *n* точок $\{x_1, x_2, \ldots, x_n\}$ на *k* непустих кластерів, що не перетинаються, щоб була мінімальною сума квадратів відстаней від кожної точки до центра відповідного кластера.

У класичному варіанті відстані використовують евклідові, хоча можуть бути методи з іншими метриками відстаней. Тобто класичний метод *k*-середніх шукає найкращу множину кластерів $\{S_1, S_2, \ldots, S_k\}$ шляхом мінімізації цільової функції:

де c_h – центр h-го кластера.

$$\sum_{h=1}^{k} \sum_{x_i \in S_h} \|x_i - c_h\|^2 \rightarrow \min_{S_1, S_2, \ldots, S_k}$$

Мінімізація цільової функції є NP-складною задачею. Для її розв'язання запропоновано декілька алгоритмів. Перший алгоритм розроблено С. Ллойдом у 1957 році і саме він є найбільш вживаним. Алгоритм розпочинає роботу з вибору *k* -точок в якості початкових центрів кластерів.

Далі ітераційно повторюють два кроки, на яких кожну точку відносять до кластера, центр якого найближчий до неї, та після рознесення усіх точок перераховують центри кластерів.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Алгоритм Ллойда, як власне й інші алгоритми розв'язання задачі (1), має ряд недоліків. Він гарантує збіжність лише до локального оптимуму, і результати його роботи суттєво залежать від вдалого вибору початкових центрів кластерів.

Наше завдання розбити це безліч об'єктів за групами (кластерами). Для цього в алгоритмі Ллойда задається:

$K = |Y|$ - число кластерів; $Y = \{1, 2,\ldots, K\}$ - номери кластерів.

Так, ми спочатку повинні явно вказати алгоритму, на скільки груп слід розбивати об'єкти вибірки. Сам він не визначає кількість кластерів.

Визначається метрика – спосіб обчислення відстаней між довільними двома точками у просторі ознак: $\rho(x_i, x_j)$

Наприклад, часто використовують евклідову відстань:

$\rho(x_i, x_j) = \|x_i - x_j\|^2 = \sum_{k=1}^{n}(x_i^k - x_j^k)^2$

Для кожного кластера нам потрібно задати початкові центри. Це можна зробити або випадковим чином, або взяти будь-які K об'єктів і вважати їх центрами: μ_a, a O Y

Після цього запускається ітераційний процес, у якому виконуються такі дії:

– кожен об'єкт x_i відноситься до найближчого центру μ_a відповідно до обраної метрики $\rho(x_i, \mu_a)$: $a_i = \arg\max_{a \in Y} \rho(x_i, \mu_a)$, $i = 1, 2,\ldots,l$

– робиться перерахунок нових положень центрів кластерів: $\mu_a = \frac{\sum_{i=1}^{l}[a_i=a]x_i}{\sum_{i=1}^{l}[a_i=a]}$, a O Y

І так доти, доки положення центрів не змінюватимуться (або змінюються дуже незначно).

Програма на Python. Алгоритм кластеризації Ллойда (K-середніх).

1. Вхідні дані.

```
import time
import numpy as np
import matplotlib.pyplot as plt
import openpyxl

book = openpyxl.open("ML, lab4.xlsx", read_only=True)
sheet = book.active

# номер першого варіанту (1, 2), другого (2, 3) і т.д
for row in range(1, 2):
 x = [(sheet[row][0].value, …, …, sheet[row][89].value)]
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

2. Основні розрахунки

```
 M = np.mean(x, axis=0)   # обчислення середніх за кожною координатою
 D = np.var(x, axis=0)    # обчислення дисперсії за кожною координатою
 K = 3                    # число кластерів
 ma = [np.random.normal(M, np.sqrt(D / 10), 2) for n in range(K)]   # початкові центри кластерів
 # ma = [np.array(x[0]), np.array(x[1])]
 ro = lambda x_vect, m_vect: np.mean((x_vect - m_vect) ** 2)       # евклідова метрика
 COLORS = ('green', 'blue', 'brown', 'black')    # кольорів має бути не менше кластерів (>= K)
 plt.ion()
 n = 0
 while n < 10:
    X = [[] for i in range(K)]     # ініціалізація порожнього двовимірного списку для зберігання об'єктів кластерів
    for x_vect in x:
       r = [ro(x_vect, m) for m in ma]       # обчислення відстаней для поточного образу до центрів кластерів
       X[np.argmin(r)].append(x_vect)      # додавання образу до кластера з найближчим центром
    ma = [np.mean(xx, axis=0) for xx in X]    # перерахунок центрів кластерів
```

3. Виведення результатів

```
    plt.clf()
    # відображення знайдених кластерів
    for i in range(K):
       xx = np.array(X[i]).T
       plt.scatter(xx[0], xx[1], s=10, color=COLORS[i])
    # відображення центрів кластерів
    mx = [m[0] for m in ma]
    my = [m[1] for m in ma]
    plt.scatter(mx, my, s=50, color='red')
    plt.draw()
    plt.gcf().canvas.flush_events()
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

```
#    plt.savefig(f"lloyd {n+1}.png")
    time.sleep(0.2)
    n += 1
plt.ioff()

# відображення знайдених кластерів
for i in range(K):
    xx = np.array(X[i]).T
    plt.scatter(xx[0], xx[1], s=10, color=COLORS[i])

# відображення центрів кластерів
mx = [m[0] for m in ma]
my = [m[1] for m in ma]
plt.scatter(mx, my, s=50, color='red')
plt.show()
```

Завдання до теми.

Завантажити набір даних з файлу ML, lab4.xlsx, за номером варіанту. Дані представлені у вигляді таблиці xlsx. У середовищі Python, вирішити завдання з використанням алгоритму кластеризації Ллойда (*K*-середніх) для свого варіанту. Побудувати графіки, пояснити отримані результати.

Джерела для самостійного опрацювання.

1. Властивості шести контрольних наборів даних кластеризації.
http://cs.joensuu.fi/sipu/datasets/
2. Кластеризація даних з пропусками методом k-середніх.
https://core.ac.uk/download/pdf/287229915.pdf
3. К-середніх++
http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

Питання для самоконтролю

1. Назвіть різницю між методами К-середніх Ллойда.
2. Дайте визначення поняття «коротший незамкнений шлях». Як його будують?
3. Як оцінюють якість кластеризації?
4. Назвіть групи методів, застосовних для формування набору інформативних ознак.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

5. Перелічіть проблеми, які можуть виникнути під час формування набору інформативних ознак для кластеризації.

Практична робота №5

Тема: Алгоритм класифікації CART на вирішальних деревах

Мета: у середовищі Python, створити алгоритм класифікації CART на вирішальних деревах.

Поставлена мета досягається послідовним вирішенням наступних задач:

1. розглянути алгоритми класифікації CART на вирішальних деревах.

Очікувані результати навчання:

- ДРН 2. Вміння застосовувати знання машинного навчання з вчителем та без нього в системах підтримки прийняття рішень та повсякденній практиці;

- ДРН 5. Вміти створювати ефективні алгоритми для обчислювальних задач системного аналізу та систем підтримки прийняття рішень.

Порядок виконання роботи

1. Вивчити необхідний теоретичний матеріал та ознайомиться з прикладами.
2. Виконати програму з наведеного прикладу в середовищі Python.
3. Побудуйте графіки, поясніть отримані результати.
4. Скласти звіт про виконання роботи, який повинен містити:

− постановку задачі;
− опис та аналіз ходу розв’язання задачі;
− висновки за результатом виконаної роботи.

Теоретичні відомості

CART, скорочення від Classification And Regression Tree, перекладається як "Дерево Класифікації та Регресії" – алгоритм бінарного дерева рішень, вперше опублікований Бріманом та ін. в 1984. Алгоритм призначений для вирішення завдань класифікації та регресії. Існує також кілька модифікованих версій – алгоритми IndCART та DB-CART. Алгоритм IndCART є частиною пакету Ind і відрізняється від CART використанням іншого способу обробки пропущених значень, не здійснює регресійну частину алгоритму CART і має інші параметри відсікання. Алгоритм DB-CART базується на наступній ідеї: замість використовувати навчальний набір даних для визначення розбиття, використовуємо його для оцінки розподілу вхідних і вихідних значень і потім використовуємо цю оцінку, щоб визначити розбиття.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

DB відповідно означає - 'distribution based'. Стверджується, що ідея дає значне зменшення помилки класифікації, порівняно зі стандартними методами побудови дерева.

У алгоритмі CART кожен вузол дерева рішень має двох нащадків. На кожному кроці побудови дерева правило, що формується у вузлі, ділить задане безліч прикладів (навчальну вибірку) на дві частини - частина, в якій виконується правило (нащадок - right) і частина, в якій правило не виконується (нащадок - left). Для вибору оптимального правила використовується функція оцінки якості розбиття.

Навчання дерева рішень відноситься до класу навчання з учителем, тобто навчальна та тестова вибірки містять класифікований набір прикладів. Оціночна функція, що використовується алгоритмом CART, базується на інтуїтивній ідеї зменшення нечистоти (невизначеності) у вузлі. Розглянемо задачу з двома класами та вузлом, що має по 50 прикладів одного класу.

Вузол має максимальну нечистоту. Якщо буде знайдено розбиття, яке розбиває дані на дві підгрупи 40:5 прикладів в одній і 10:45 в іншій, інтуїтивно 'нечастота' зменшиться. Вона повністю зникне, коли буде знайдено розбиття, яке створить підгрупи 50:0 та 0:50.

В алгоритмі CART ідея "нечастоти" формалізована в індексі Gini.

Якщо набір даних містить дані n класів, тоді індекс Gini визначається як:

$Gini(T) = 1 - \sum_{i=1}^{n} p_i^2$

де p_i – ймовірність (відносна частота) класу i в T.

Якщо набір T розбивається на дві частини T_1 і T_2 з числом прикладів у кожному N_1 і N_2 відповідно, тоді показник якості розбиття дорівнюватиме:

$$Gini_{split}(T) = \frac{N_1}{N} \cdot Gini(T_1) + \frac{N_2}{N} \cdot Gini(T_2)$$

Найкращим вважається те розбиття, котре $Gini_{split}(T)$ мінімально.

Позначимо N – число прикладів у вузлі – предку, L, R – число прикладів відповідно у лівому та правому нащадку, l_i та r_i – число екземплярів i-го класу у лівому/правому нащадку. Тоді якість розбиття оцінюється за такою формулою:

$$Gini_{split} = \frac{L}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{l_l}{L}\right)^2\right) + \frac{R}{N}\left(1 - \sum_{i=1}^{n}\left(\frac{r_l}{R}\right)^2\right) \rightarrow \min$$

Щоб зменшити обсяг обчислень формулу можна перетворити:

$$Gini_{split} = \frac{1}{N}\left(L \cdot \left(1 - \frac{1}{L^2} \cdot \sum_{i=1}^{n} {l_l}^2\right) + R \cdot \left(1 - \frac{1}{R^2} \cdot \sum_{i=1}^{n} {r_l}^2\right)\right) \rightarrow \min$$

Так як множення на константу не відіграє ролі при мінімізації:

$$Gini_{split} = L - \frac{1}{L} \cdot \sum_{i=1}^{n} {l_l}^2 + R - \frac{1}{R} \cdot \sum_{i=1}^{n} {r_l}^2 \rightarrow \min$$

$$Gini_{split} = N - \left(\frac{1}{L} \cdot \sum_{i=1}^{n} {l_l}^2 + \frac{1}{R} \cdot \sum_{i=1}^{n} {r_l}^2\right) \rightarrow \min$$

$$\tilde{G}_{split} = \frac{1}{L} \cdot \sum_{i=1}^{n} {l_l}^2 + \frac{1}{R} \cdot \sum_{i=1}^{n} {r_l}^2 \rightarrow \max$$

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

У результаті, найкращим буде те розбиття, для якого величина $\tilde{G}_{split}$ максимальна. Рідше в алгоритмі CART використовуються інші критерії розбиття Towing, Symmetric *Gini* та ін.

Вектор предикторних змінних, що подається на вхід дерева, може містити як числові (порядкові) так і категоріальні змінні. У будь-якому випадку в кожному вузлі розбиття йде лише по одній змінній. Якщо змінна числового типу, то у вузлі формується правило виду $x \leq c$.

Де с – деякий поріг, який найчастіше вибирається як середнє арифметичне двох сусідніх упорядкованих значень змінної хі навчальної вибірки. Якщо змінна категоріального типу, то у вузлі формується правило $x_i \in V(x_i)$, де $V(x_i)$ - деяке непусте підмножина безлічі значень змінної хі у навчальній вибірці. Отже, для n значень числового атрибуту алгоритм порівнює n-1 розбиття, а категоріального $(2_{n-1}–1)$. На кожному кроці побудови дерева алгоритм послідовно порівнює всі можливі розбиття всім атрибутів і вибирає найкращий атрибут і найкраще розбиття йому.

Механізм відсікання дерева, оригінальна назва minimal cost-complexity tree pruning - найбільш серйозна відмінність алгоритму CART від інших алгоритмів побудови дерева. CART розглядає відсікання як отримання компромісу між двома проблемами: отримання дерева оптимального розміру та отримання точної оцінки ймовірності хибної класифікації.

Основна проблема відсікання – велика кількість всіх можливих відсічених піддерев для одного дерева.

Точніше, якщо бінарне дерево має $|T|$ – листів, тоді існує ~ $[1.5028369|T|]$ відсічених піддерев. І якщо дерево має хоча б 1000 листів, тоді кількість відсічених піддерев стає просто величезним.

Базова ідея методу – не розглядати всі можливі піддерев, обмежившись лише 'кращими представниками' згідно з наведеною нижче оцінкою.

Позначимо $| T |$ – число листів дерева, $R(T)$ – помилка класифікації дерева, рівна відношенню числа неправильно класифікованих прикладів до прикладів у навчальній вибірці. Визначимо $C\alpha(T)$ – повну вартість (оцінку/показник витрати-складність) дерева T як: $C\alpha(T) = R(T) + \alpha*|T|$, Де | Т | - число листів (термінальних вузлів) дерева, α - деякий параметр, що змінюється від 0 до + ∞.

Повна вартість дерева складається з двох компонентів – помилки класифікації дерева та штрафу за його складність. Якщо помилка класифікації дерева незмінна, тоді зі збільшенням повна вартість дерева збільшуватиметься.

Тоді в залежності від α менш гіллясте дерево, що дає велику помилку класифікації, може коштувати менше, ніж дає меншу помилку, але гіллясте. Визначимо T_{max} - максимальне за розміром дерево, яке потрібно обрізати.

Якщо ми зафіксуємо значення α, тоді існує найменше мінімізоване піддерево α, яке виконує такі умови:

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

$C\alpha\ (T(\alpha)) = \min T \leq T_{\max}\ C\alpha(T)$

якщо $C\alpha\ (T) = C\alpha(T(\alpha))\ then\ T(\alpha) \leq T$

Перша умова каже, що немає такого піддерева дерева $T_{\max}$, яке мало меншу вартість, ніж $T(\alpha)$ у своїй значенні α. Друга умова говорить, що якщо існує більше одного піддерева, що має цю повну вартість, тоді ми вибираємо найменше дерево.

Можна показати, що для будь-якого значення існує таке найменше поддерево, що мінімізується. Але це завдання не тривіальне. Що вона каже - що не може бути такого, коли два дерева досягають мінімуму повної вартості і вони є незрівнянними, тобто. жодне з них не є піддеревом іншого. Ми не доводитимемо цього результату.

Хоча має нескінченну кількість значень, існує кінцева кількість піддерев дерева Tmax. Можна побудувати послідовність зменшуються піддерев дерева T_{max}:

$T_1 > T_2 > T_3 > \ldots > \{t_1\}$,

де t_1 – кореневий вузол дерева) таку, що T_k – найменше мінімізоване піддерево для $\alpha \in [\alpha_k, \alpha_{k+1})$. Це важливий результат, оскільки це означає, що ми можемо отримати наступне дерево в послідовності, застосувавши відсікання до поточного дерева.

Це дозволяє розробити ефективний алгоритм пошуку найменшого піддерева, що мінімізується, при різних значеннях α. Перше дерево у цій послідовності – найменше піддерево дерева T_{max}, має таку помилку класифікації, як і T_{max}, тобто. $T_1 = T(\alpha = 0)$. Пояснення: якщо розбиття йде до тих пір, поки в кожному вузлі залишиться лише один клас, то $T_1 = T_{max}$, але так як часто застосовуються методи ранньої зупинки (prepruning), тоді може існувати дерево піддерево T_{max} має таку ж помилку класифікації.

Як ми отримуємо наступне дерево у послідовності та відповідне значення α? Позначимо T_t – гілка дерева T із кореневим вузлом t. При яких значеннях дерево $T - T_t$ буде краще, ніж T? Якщо ми відсічемо у вузлі t, тоді його внесок у повну вартість дерева $T - T_t$ стане $C\alpha(\{t\}) = R(t) + \alpha$, де $R(t) = r(t)^* p(t)$, $r(t)$ – це помилка класифікації вузла t та p(t) – пропорція випадків, які 'минули' через вузол t.

Альтернативний варіант: $R(t) = m/n$, де m – число прикладів класифікованих некоректно, а n – загальна кількість класифікованих прикладів для дерева.

Вклад T_t на повну вартість дерева T складе $C\alpha(T_t) = R(T_t) + \alpha|T_t|$,

Де

$R(T_t) = \sum_{t' \in T_t} R(t)$

Дерево $T - T_t$ буде краще, ніж T, коли $C\alpha(\{t\}) = C\alpha(T_t)$, тому що при цій величині вони мають однакову вартість, але $T - T_t$ найменше з двох.

Коли $C\alpha\ (\{t\}) = C\alpha\ (T_t)$ ми отримуємо:

$R(T_t) = \alpha|T_t| = R(t) + \alpha$

вирішуючи для α, отримуємо:

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

$\alpha = \frac{R(t)-R(T_t)}{|T_t|-1}$

Так для будь-якого вузла t в T_1 якщо ми збільшуємо α, тоді, коли

$\alpha = \frac{R(t)-R(T_{1,t})}{|T_{1,t}|-1}$

дерево, отримане відсіканням у вузлі t, краще, ніж T_1.

Основна ідея полягає в наступному: обчислимо це значення α для кожного вузла в дереві T_1, а потім виберемо "слабкі зв'язки" (їх може бути більше, ніж одна), тобто. вузли для яких величина

$g(t) = \frac{R(t)-R(T_{1,t})}{|T_{1,t}|-1}$ є найменшою.

Ми відсікаємо T_1 у цих вузлах, щоб отримати T_2 – наступне дерево у послідовності. Потім ми продовжуємо цей процес для отриманого дерева і так поки що ми не отримаємо кореневий вузол (дерево в якому лише один вузол).

Реалізація вирішальних дерев на Python за допомогою Scikit-Learn.

Отже, ми з вами загалом познайомилися з ідеєю вирішальних дерев для завдань класифікації та регресії. Залишилося дізнатися, як можна реалізувати ці підходи під час вирішення практичних завдань. У найпростішому варіанті це можна зробити мовою Python із використанням бібліотеки Scikit-Learn.

У цій бібліотеці є гілка:

```
from sklearn import tree
```

яка відповідає за побудову вирішальних дерев.

Розробники Scikit-Learn використовували алгоритм:

Classification and Regression Trees (CART)

з допомогою якого можна виконувати як класифікацію, і вирішувати завдання регресії.

Докладніше про це можна почитати на сторінці офіційної документації:

https://scikit-learn.org/stable/modules/tree.html

Побудуємо вирішальне дерево для завдання регресії.

1. Вхідні дані. Сформуємо навчальну множину у вигляді значень функції косинуса:

```
import numpy as np
import matplotlib.pyplot as plt
x=np.arange(0,np.pi, 0.1).reshape(-1, 1)
y=np.cos(x)
```

3. Основні розрахунки. З цим набором даних збудуємо вирішальне дерево з максимальною глибиною 3:

```
clf=tree.DecisionTreeRegressor(max_depth=3)
clf=clf.fit(x,y)
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Після цього виконаємо прогноз за значеннями *x*:

```
yy=clf.predict(x)
```

1. Виведемо результати та відобразимо отримані графіки:

```
#tree.plot_tree(clf)
plt.plot(x,y,label="cos(x)")
plt.plot(x,yy,label="DT Regression")
plt.grid()
plt.legend()
plt.show()
```

Додатково можна вивести структуру отриманого дерева за допомогою команди:

```
tree.plot_tree(clf)
```

За замовчуванням клас DecisionTreeRegressor використовує критерій мінімуму квадрата помилки прогнозу, як ми говорили на цьому занятті. Але, за потреби, можна скористатися й іншим критерієм – мінімум модуля помилки прогнозу. Докладно про це написано в офіційній документації (за посиланням, наведеним вище). Додатково клас DecisionTreeRegressor має інші параметри для управління побудовою дерева. Все це можна переглянути в документації.

Вирішальні дерева для завдань класифікації

Під час вирішення завдань класифікації використовується клас DecisionTreeClassifier бібліотеки Scikit-Learn:

```
tree.plot_tree(clf)
From sklearn.tree import DecisionTreeClassifier
```

Він також має багато різних вхідних параметрів для управління побудовою дерева. Про все докладно можна прочитати в офіційній документації. Тут же зазначимо, що за умовчанням використовується критерій Джині як impurity і методика усічення дерева (pruning).

Завдання до теми.

У середовищі Python, вирішити завдання з використанням алгоритму класифікації CART на вирішальних деревах для свого варіанту. Побудувати графіки, пояснити отримані результати.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Завдання для прикладу.

Номер варіанту	Глибина дерева	Вхідний сигнал		
1	3	0	1,0	0,1
2	4	0	2,0	0,2
3	5	0	3,0	0,1
4	3	0	4,0	0,2
5	4	0	5,0	0,1
6	5	0	1,0	0,2
7	3	0	2,0	0,2
8	4	0	3,0	0,1
9	5	0	4,0	0,2
10	3	0	5,0	0,1
11	4	0	1,0	0,2
12	5	0	2,0	0,1
13	3	0	3,0	0,2
14	4	0	4,0	0,2
15	5	0	5,0	0,1
16	3	0	1,0	0,2
17	4	0	2,0	0,1
18	5	0	3,0	0,2
19	3	0	1,0	0,1
20	4	0	2,0	0,2
21	5	0	3,0	0,2
22	3	0	4,0	0,1
23	4	0	5,0	0,2
24	5	0	1,0	0,1
25	4	0	2,0	0,2
26	5	0	3,0	0,1
27	3	0	4,0	0,2
28	4	0	5,0	0,2
29	5	0	1,0	0,1
30	4	0	2,0	0,2

Додаткові завдання

У середовищі Python, створити алгоритм і програму випадковий ліс для задач регресії. Побудувати графіки, пояснити отримані результати.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Джерела для самостійного опрацювання.

1. CART [Електронний ресурс]/ Режим доступу: https://machinelearningmastery.com/classification-and-regression-trees-for-machine-learning/
2. https://scikit-learn.org/stable/modules/tree.html
3. Класифікація та прогнозування https://moodle.znu.edu.ua/pluginfile.php?file=/486136/mod_resource/content/1/%D0%9B%D0%B5%D0%BA%D1%86%D1%96%D1%8F%209.pdf

Питання для самоконтролю

1. Дайте визначення методу CART.
2. Основні відмінності алгоритму CART від алгоритмів сімейства ID3
2. Дайте визначення функції оцінки якості розбиття?
3. Наведіть механізм відсікання дерева.
4. Як виконується алгоритм обробки пропущених значень?
5. Як виконується побудова дерев регресії.

Практична робота № 6

Тема: Алгоритм класифікації AdaBoost на вирішальних деревах

Мета: у середовищі Python, створити Алгоритм класифікації AdaBoost

Поставлена мета досягається послідовним вирішенням наступних задач:

1. розглянути алгоритми класифікації AdaBoost на вирішальних деревах;

Очікувані результати навчання:

- ДРН 7. Використовувати алгоритми побудови асоціативних та класифікуючих правил;
- ДРН 8. Будувати алгоритми AdaBoost для класифікації та навчання нейронних мереж.

Порядок виконання роботи

1. Вивчити необхідний теоретичний матеріал та ознайомиться з прикладами.

2. Виконати програму з наведених прикладів у середовищі Python.
3. Побудуйте графіки, поясніть отримані результати.
4. Скласти звіт про виконання роботи, який повинен містити:

– постановку задачі;
– опис та аналіз ходу розв'язання задачі;
– висновки за результатом виконаної роботи.

Теоретичні відомості. Суть цього у тому, що сильний класифікатор створюють з урахуванням слабких: кожна нова модель навчається на помилках попередньої. Тобто щоразу додаються дедалі нові моделі, які намагаються виправити помилки своїх попередників. Так триває до того часу, поки прогнози стануть безпомилковими чи досягне ліміт кількість моделей (рис. 1). Кожна наступна модель навчається на попередніх помилках, так можна сильно підвищити точність прогнозів.

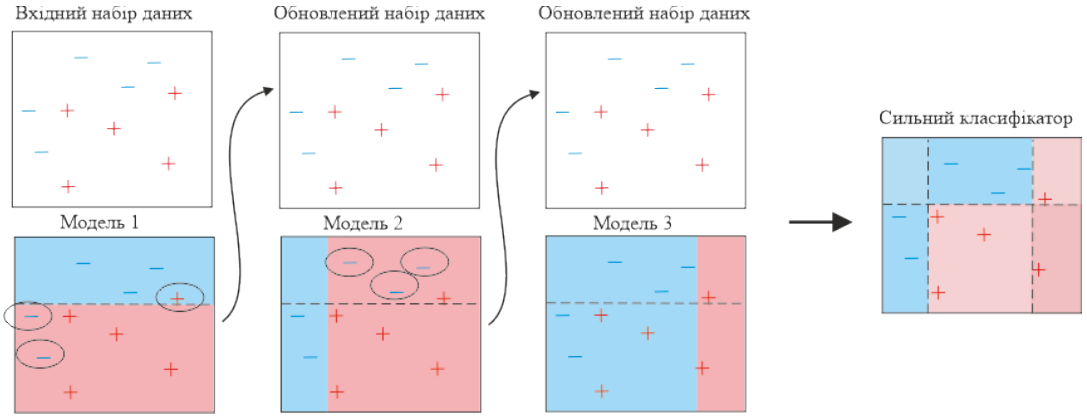

Рис. 1. Алгоритм класифікації AdaBoost

AdaBoost – частковий випадок градієнтного бустинга, у якому в 49 якості функції втрат використовується експоненційна. AdaBoost являється першим теоретично дослідженим варіантом бустинга, для якого вперше була доведена основна теорема із аналітично виведеними оптимальними b_m на кожному кроці. Початкова версія алгоритму повертала тільки номер класу у бінарній класифікації; пізніше він був узагальнений та тепер являється можливість отримати ймовірність належності відповідному класу.

Розглянемо конкретний приклад - алгоритм AdaBoost (скорочення від Adaptive Boosting), який і запропонували Фройнд з Шапіро стосовно завдань бінарної класифікації з мітками класів Y = {-1, +1}.

Для апроксимації порогового функціоналу якості було запропоновано використовувати експоненційну функцію втрат:

$$Q_T = \sum_{i=1}^{l} [M_i < 0] \le \sum_{i=1}^{l} exp\,(-M_i) = \tilde{Q}_T,$$

Якщо розписати цей вираз, то отримаємо вираз:

$$Q_T \le \tilde{Q}_T = \sum_{i=1}^{l} exp\,(-y_i \cdot \sum_{t=1}^{T} a_t\, b_t(x_i))$$

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Тепер дивіться. Відповідно до першої евристики, ми шукатимемо найкращий алгоритм $b_T(x)$ і вага a_T при фіксованих попередніх алгоритмах $b_1(x),\ldots,b_{T-1}(x)$ та вагах $a_1,\ldots,a_{T-1}$. Тому перепишемо функціонал якості $\tilde{Q}_T$, наступним чином:

$$Q_T \le \tilde{Q}_T = \sum_{i=1}^{l} \underbrace{exp(-y_i \cdot \sum_{t=1}^{T-1} a_t\, b_t(x_i))}_{w_i} \cdot \exp(-y_i \cdot \alpha_T \cdot b_T(x_i))$$

Обидві формули – те саме, але на другий ми явно (окремо) винесли $b_T(x)$, a_T. І перед другою експонентою з'являється множник (вага) w_t, тому що попередні алгоритми та коефіцієнти зафіксовані.

Давайте проаналізуємо та подивимося, який сенс мають ці ваги. По-перше, вони обчислюються для кожного об'єкта $\{x_i\}$ навчальної вибірки, тобто це фактично ваги об'єктів. По-друге, множник буде тим більше, чим неактивнішим буде відступ (margin). Тобто, множник зростає для об'єктів, що погано класифікуються.

В результаті, поточний алгоритм $b_T(x)$ більшою мірою при навчанні враховуватиме ці погано класифіковані образи і налаштовуватиметься на них. Це і є реалізація принципу бустінгу – послідовно навчати алгоритми на об'єктах, на яких попередня композиція показала найгірші результати.

Далі, за алгоритмом, ваги $\{w_i\}$ нормуються під час пошуку кожного наступного алгоритму за формулою:

$$\tilde{w}_i = \frac{w_i}{\sum_{j=1}^{l} w_j},\ i = 1, 2,\ldots,l$$

Тобто ми робимо так, щоб у сумі нормовані ваги давали одиницю:

$$\sum_{i=1}^{l} \tilde{w}_t = 1$$

Завдяки цьому вони не будуть йти в нуль або в нескінченність, а приймати цілком розумні значення. Це важливо під час реалізації алгоритму на комп'ютері. У результаті кожного поточного алгоритму $b(x)$ ми можемо підрахувати частку вірних і неправильних класифікацій:

$$N(b) = \sum_{i=1}^{l} \tilde{w}_i \cdot [b(x_i) \ne y_i]$$
$$P(b) = \sum_{i=1}^{l} \tilde{w}_i \cdot [b(x_i) = y_i] = 1 - N(b)$$

Очевидно, найкращий поточний алгоритм має мінімізувати частку невірних класифікацій:

$$b_T = \arg\max_{b \in B} N(b)$$

(Це відповідає обраному показнику якості). І далі, Фройнд з Шапіро довели, що найкраще (оптимальне) значення вагового коефіцієнта при експоненційній функції втрат і знайденого алгоритму $b_T(x)$ можна обчислити за простою формулою:

$$\alpha_t = \frac{1}{2} ln \frac{1 - N(b_t)}{N(b_t)}$$

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Все це і є основою алгоритму AdaBoost. Фактично, ми кожної ітерації (від 1 до T) навчаємо поточний алгоритм $b_j(x)$ на зважених об'єктах навчальної вибірки. Саме навчання виконується будь-яким відомим нам способом. А потім для нього обчислюється коефіцієнт a_j.

Зараз ми в деталях подивимося на роботу цього алгоритму, коли як алгоритми $\{b_j(x)\}$ виступають вирішальні дерева. Але спочатку я наведу псевдокод загального алгоритму AdaBoost:

Вхід: навчальна вибірка X_l і параметр T (число алгоритмів композиції)

Вихід: набір базових алгоритмів $b_1(x),\ldots, b_T(x)$ з вагами $\alpha_1,\ldots, \alpha_T$

1: початкова ініціалізація ваг об'єктів: $w_i = 1 / l$, $i = 1,\ldots,l$,

2: для всіх $t = 1,\ldots,T$

3: знайти найкращий поточний алгоритм за правилом:

$b_t = arg \max_b N(b)$

4: обчислити оптимальний коефіцієнт:

$\alpha_t = \frac{1}{2} ln \frac{1-N(b_t)}{N(b_t)}$

5: оновити ваги об'єктів:

$w_i = w_i \cdot \exp(-\alpha_t y_i b_t(x_i))$; $i = 1,\ldots,l$

6: нормувати ваги об'єктів:

$w_{sum} = \sum_{i=1}^{l} w_j$; $w_i = w_i / w_{sum}$; $i = 1,\ldots,l$

Реалізація алгоритму AdaBoost на Python з використанням вирішальних дерев як базових алгоритмів $\{b_t(x)\}$.

1. Вхідні дані.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeClassifier
import openpyxl

def get_grid(data):
   x_min, x_max = data[:, 0].min() - 10, data[:, 0].max() + 10
   y_min, y_max = data[:, 1].min() - 10, data[:, 1].max() + 10
   return np.meshgrid(np.arange(x_min, x_max, 1), np.arange(y_min, y_max, 1))
book = openpyxl.open("ML, lab6.xlsx", read_only=True)
sheet = book.active
# номер першого варіанту (1, 2), другого (2, 3) і т.д
for row in range(1, 2):
 t = [(sheet[row][0].value, …, …, sheet[row][89].value)]
n1 = len(t[0])
n2 = len(t[1])
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

2. Основні розрахунки

```
train_data = np.r_[t[0], t[1]]
train_labels = np.r_[np.ones(n1) * -1, np.ones(n2)]

# x, y = train_data[:, 0], train_data[:, 1]
# plt.scatter(x[train_labels == -1], y[train_labels == -1])
# plt.scatter(x[train_labels == 1], y[train_labels == 1])
# plt.show()
XN = len(train_data)    # довжина навчальної вибірки
T = 1                # число алгоритмів у композиції
max_depth = 2          # максимальна глибина вирішальних дерев
w = np.ones(XN) / XN    # початкові значення ваги для об'єктів вибірки
algs = []             # список з отриманих алгоритмів
alfa = []             # список з обчислених ваг для композиції

for n in range(T):
    # створюємо та навчаємо вирішальне дерево з вагами об'єктів w
    algs.append(DecisionTreeClassifier(criterion='gini', max_depth=max_depth))
    algs[n].fit(train_data, train_labels, sample_weight=w)

    predicted = algs[n].predict(train_data) # формуємо прогнози отриманого дерева за навчальною вибіркою
    N = np.sum(np.abs(train_labels - predicted) / 2) / XN   # обчислюємо частку невірних класифікацій
    alfa.append( 0.5 * np.log((1 - N) / N) if N != 0 else np.log((1-1e-8) / 1e-8) ) # обчислюємо вагу для поточного алгоритму

    # перераховуємо ваги об'єктів вибірки
    w = w * np.exp(-1 * alfa[n] * train_labels * predicted)
    w = w / np.sum(w)

# обчислюємо число помилок класифікації на основі отриманої композиції
predicted = alfa[0] * algs[0].predict(train_data)
for n in range(1, T):
    predicted += alfa[n] * algs[n].predict(train_data)

N = np.sum(np.abs(train_labels - np.sign(predicted)) / 2)
```

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

3. Виведення результатів

```
    print(f"Кількість помилок на навчальній вибірці: {N} при композиції {T} вирішальних дерев")

  # Відображаємо отримані результати класифікації
  xx, yy = get_grid(train_data)
  predicted = alfa[0] * algs[0].predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
  for n in range(1, T):
      predicted += alfa[n] * algs[n].predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

  plt.pcolormesh(xx, yy, predicted, cmap='spring', shading='auto')
  plt.scatter(train_data[:, 0], train_data[:, 1], c=train_labels, s=5000 * w, cmap='spring', edgecolors='black', linewidth=1.5)
  plt.show()
```

Завдання до теми.

Завантажити набір даних з файлу ML, lab6.xlsx, за номером варіанту. Дані представлені у вигляді таблиці xlsx. У середовищі Python, вирішити завдання з використанням алгоритму класифікації AdaBoost на вирішальних деревах для свого варіанту. Побудувати графіки, пояснити отримані результати.

Джерела для самостійного опрацювання.

1. . Gradient Boosting
https://machinelearningmastery.com/gradient-boosting-algorithm-machine-learning/
2. Guide on AdaBoost Algorithm
https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/
3. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html
4. AdaBoost, Clearly Explained
https://www.youtube.com/watch?v=LsK-xG1cLYA

5. Understanding the AdaBoost Algorithm
https://medium.com/@datasciencewizards/understanding-the-adaboost-algorithm-2e9344d83d9b

Питання для самоконтролю

1. Що таке бегінг та метод випадкових підмножин? Що таке випадковий ліс, як він навчається і як він будує прогнози?
2. У чому ідея розкладання помилки на зміщення та розкид? Як бегінг змінює зміщення та розкид однієї моделі?
3. Опишіть ідею градієнтного бустингу для середньоквадратичної помилки. Запишіть завдання для навчання чергової базової моделі.
4. Опишіть ідею градієнтного бустингу довільної функції втрат. Запишіть завдання для навчання чергової базової моделі.
5. Якими способами можна подолати перенавчання моделі:
    1. у випадку (незалежно від моделі);
    2. якщо модель лінійна;
    3. якщо модель – градієнтний бустинг.

Оцінювання результатів навчання

Оцінювання практичних занять та дисциплінарні результати

Подаються критерії оцінювання у вигляді переліку припущених недоліків, що знижують оцінку якості виконання цього практичного завдання.

Об’єктивна оцінка результатів розв’язання задач можлива (як і будь-яке інше вимірювання) лише при їх зіставленні з еталонними – зразками правильних та повних рішень (відповідей).

Оцінювання результатів практичних занять здійснюється наступним чином:

1 питання – 2,5 бала;

2 питання – 2,5 бала;

3 питання – 2,5 бала;

4 питання – 2,5 бала.

Критеріями визначення оцінок приймається:

«Відмінно» – 9-10 балів;

«Добре» – 7,5-9 бали;

«Задовільно» – 6-7,5 бали;

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

«Незадовільно» – до 6 балів.

При остаточній оцінці результатів виконання завдання необхідно враховувати здатність студента:

- диференціювати, інтегрувати та уніфікувати знання;
- застосовувати правила, методи, принципи, закони у конкретних ситуаціях;
- інтерпретувати схеми, графіки, діаграми;
- аналізувати й оцінювати факти, події та спрогнозувати очікувані результати від прийнятих рішень;
- викладати матеріал на папері послідовно, з дотриманням вимог чинних стандартів.

Вимоги до звіту

Звіт з виконання практичної роботи має містити:

– обкладинку в корпоративному стилі НТУ «Дніпровська політехніка» із зазначенням групи, ПІ Б студента, викладача та теми роботи;

– мету роботи і повний текст завдання з уточненням варіанту системи, що розглядається;

– виконання завдання з побудовою конструктивної та потокової функціональних діаграм системи, описом її функціонування, аналізом переваг та недоліків системи і висновками щодо шляхів її покращення.

Звіт розміщується на сайті дистанційної освіти у відповідному розділі із завданням і оцінюється виходячи з 100 балів в ході усного захисту.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Рекомендовані джерела інформації

1. Murphy, K. P. (2012). Machine Learning: a Probabilistic Perspective. MIT Press, Cambridge, MA, USA
2. Farabet, C., LeCun, Y., Kavukcuoglu, K., Culurciello, E., Martini, B., Akselrod, P., and Talay, S. (2011). Large-scale FPGA-based convolutional networks. In R. Bekkerman, M. Bilenko, and J. Langford, editors, Scaling up Machine Learning: Parallel and Distributed Approaches. Cambridge University Press.
3. Kevin P. Murphy «Machine Learning: A Probabilistic Perspective», 2012.
4. Кононова К. Ю. Машинне навчання: методи та моделі: підручник для бакалаврів, магістрів та докторів філософії спеціальності 051 «Економіка» / К. Ю. Кононова. – Харків: ХНУ імені В. Н. Каразіна, 2020. – 301 с.
5. Черняк О. І. Інтелектуальний аналіз даних: підручник / О. І. Черняк, П. В. Захарченко; Київський національний університет імені Т. Шевченка. К.: Знання, 2010. 834 с.

Желдак Тімур Анатолійович, доцент, канд. техн. наук, зав кафедри, zheldak.t.a@nmu.one
Владико Олександр Борисович, доцент, канд. техн. наук, доцент, vladyko.o.b@nmu.one
Малієнко Андрій Вікторович, доцент, канд. техн. наук, доцент, malienko.a.v@nmu.one
Гаранжа Дмитро Миколайович - старший викладач, haranzha.d.m@nmu.one

Навчальне видання

Желдак Тимур Анатолійович
Владико Олександр Борисович
Малієнко Андрій Вікторович
Гаранжа Дмитро Миколайович

МАШИННЕ НАВЧАННЯ

Методичні рекомендації до виконання практичних робіт
для здобувачів ступеня магістра
освітньо-професійної програми «Системний аналіз»
зі спеціальності 124 Системний аналіз

Видано в авторській редакції.

Електронний ресурс.
Підписано до видання 11.06.2024. Авт. арк. 3,6.

Національний технічний університет «Дніпровська політехніка».
49005, м. Дніпро, просп. Дмитра Яворницького, 19.