Міністерство освіти і науки України

Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

Лабораторна робота №3

З дисципліни: "Інтелектуальний аналіз даних"

Тема: " Побудова поверхонь, які поділяють об'єкти різних класів у системах розпізнавання"

Підготувала студентка

Групи АІ – 212

Козуб К.О

Перевірили:

Антощук С.Г

Кошутіна Д.В

Одеса 2023

Мета роботи: Розробка і реалізація алгоритму пошуку кордонів між існуючими класами на основі використання відомих геометричних мір близькості.

<https://colab.research.google.com/drive/1WmYu1XkT89VYQqwkldXJeGD0Q-sqAA6l?usp=sharing>

Хід роботи

**Завдання:**

1. Розробити алгоритм побудови лінійних розділяючих площин між класами.
2. Виконати програмну реалізацію розробленого алгоритму, налагодити програму.

**Код:**

# Імпорт необхідних бібліотек

from google.colab import drive

import matplotlib.pyplot as plt

import numpy as np

import os

# Монтування Google Drive

drive.mount('/content/gdrive/')

# Зміна поточного робочого каталогу на Google Drive

%cd /content/gdrive/MyDrive/IA2023

# Оголошення словника із даними для класів

class\_dict = {

'I': np.load('class1.npy'),

'II': np.load('class2.npy'),

'V': np.load('class5.npy'),

}

# Оголошення словника із маркерами та кольорами для кожного класу

markers\_dict = {

'I': ('brown', '\*'),

'II': ('grey', '+'),

'V': ('green', '.'),

}

# Створення графіку для кожного класу

for key, class\_ in class\_dict.items():

plt.scatter(class\_[:, 0], class\_[:, 1], c=markers\_dict[key][0], marker=markers\_dict[key][1], label=key)

# Додавання заголовка та налаштувань графіку

plt.title('Initial data (MaxAbsScaled)')

plt.grid()

plt.legend()

plt.show()

**Результат:**

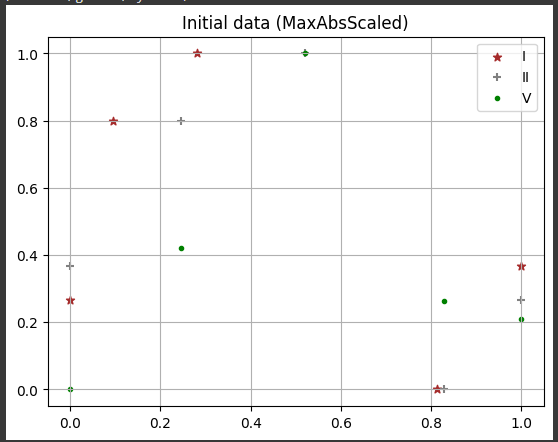
****

Рисунок 1 - Початкові дані (MaxAbsScaled)

**Код:**

Визначення барицентів

def get\_centroids(class\_dict: dict):

  centroids\_dict = {key: (np.mean(class\_, axis=0)) for key, class\_ in class\_dict.items()}

  return centroids\_dict

**Код:**

Визначення середин відрізків

def get\_points\_centers(centroids\_dict: dict):

    # Ініціалізуємо порожні словники для збереження центрів та оброблених центрів

    points\_centers = {}  # Тут будемо зберігати центри між точками

    processed\_centers = []  # Тут відслідковуємо оброблені центри

    # Перебираємо центроїди в словнику `centroids\_dict`

    for key, point in centroids\_dict.items():

        for other\_key, other\_point in centroids\_dict.items():

            if other\_key != key:

                # Обчислюємо центр між двома центроїдами

                center = (point[0] + other\_point[0]) / 2, (point[1] + other\_point[1]) / 2

                # Перевіряємо, чи цей центр вже був оброблений

                if center not in processed\_centers:

                    processed\_centers.append(center)

                    # Перевіряємо, чи існує центр для ключа `key` в словнику `points\_centers`

                    # Якщо так, то ми встановлюємо ключ `key` на значення іншого ключа `other\_key`

                    if key in points\_centers:

                      key = other\_key

                    # Додаємо центр до словника `points\_centers` з ключем `key`

                    points\_centers[key] = center

    return points\_centers

**Код:**

Візуалізація результатів

# Отримуємо центроїди для кожного класу з використанням функції get\_centroids

centroids = get\_centroids(class\_dict)

print("Центроїди:", centroids)

# Отримуємо центри між точками для кожного класу з використанням функції get\_points\_centers

points\_centers = get\_points\_centers(centroids)

print("Центри між точками:", points\_centers)

# Створюємо графіки для кожного класу та позначаємо центроїди та центри між точками

for key, class\_ in class\_dict.items():

    # Візуалізуємо дані класу з використанням раніше визначених маркерів та кольорів

    plt.scatter(class\_[:, 0], class\_[:, 1], c=markers\_dict[key][0], marker=markers\_dict[key][1], label=key)

    # Візуалізуємо центроїд класу за допомогою квадратного маркера

    centroid = centroids[key]

    plt.scatter(centroid[0], centroid[1], c=markers\_dict[key][0], marker='s')

    # Візуалізуємо центр між точками класу за допомогою квадратного маркера та сірого кольору

    point\_center = points\_centers[key]

    plt.scatter(point\_center[0], point\_center[1], c='gray', marker='s')

# Обчислюємо координати центроїдів для побудови відрізків між ними

centroid\_coords = [centroid for centroid in centroids.values()]

# Побудова відрізків між центроїдами за допомогою сірих пунктирних ліній

for i in range(len(centroid\_coords)):

    plt.plot([centroid\_coords[i][0], centroid\_coords[(i + 1) % len(centroid\_coords)][0]],

             [centroid\_coords[i][1], centroid\_coords[(i + 1) % len(centroid\_coords)][1]],

             c='gray', linestyle=':')

# Налаштування заголовку та інших параметрів графіку

plt.title('Initial data (MaxAbsScaled)')

plt.grid()

plt.legend()

plt.show()

**Результат:**

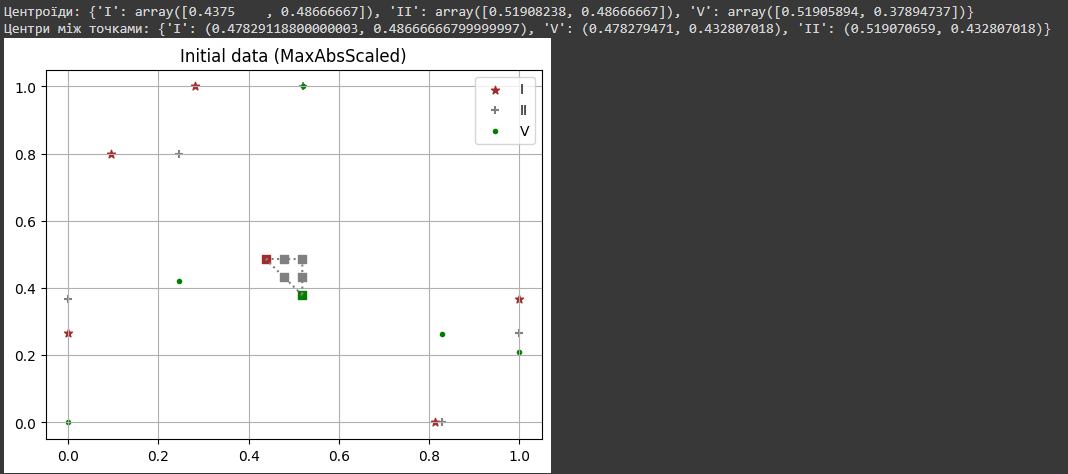
****

Рисунок 2

**Код:**

Support vector machine SVM

from sklearn.svm import SVC

# Створення комбінованого набору даних і міток для SVM

data = np.concatenate([class\_ for class\_ in class\_dict.values()])

# Заповнення мітками на кожен об'єкт класу

labels = np.concatenate((np.zeros(class\_dict['I'].shape[0]),

                         np.ones(class\_dict['II'].shape[0]),

                         2 \* np.ones(class\_dict['V'].shape[0])))

# Створення класифікатора SVM

clf = SVC(kernel='linear')

clf.fit(data, labels)

**Результат:**



**Код:**

def show\_boundaries():

  xx, yy = np.meshgrid(np.linspace(0, 1.1, 100),

                      np.linspace(0, 1.1, 100))

  # Прогназування класу для кожної точки на поверхні

  Z = clf.predict(np.c\_[xx.ravel(), yy.ravel()])

  Z = Z.reshape(xx.shape)

  # Відображення розділючих площин

  plt.contourf(xx, yy, Z)

  plt.contour(xx, yy, Z, colors='black', linewidths=1)

**Код:**

Візуалізація границь

show\_boundaries()

for key, class\_ in class\_dict.items():

  plt.scatter(class\_[:, 0], class\_[:, 1], c=markers\_dict[key][0], marker=markers\_dict[key][1], label=key)

plt.title('Розділяючі площини')

plt.grid()

plt.legend()

plt.show()

**Результат:**

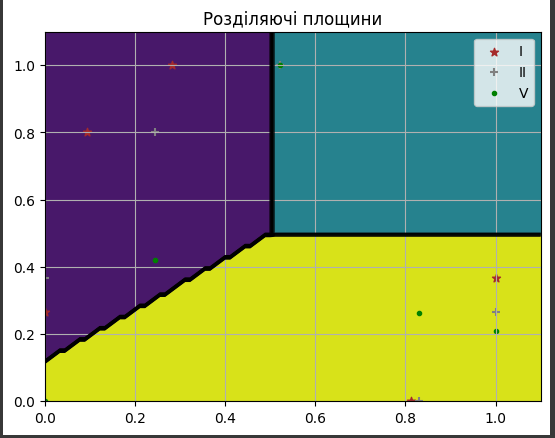


Рисунок 3 - Розділяючі площини

**Код:**

Функція для генерації невідомих об'єктів

import random

def generate\_objects(amount):

  obj\_list = [(random.randint(0, 100) / 100, random.randint(0, 100) / 100) for \_ in range(amount)]

  return obj\_list

**Код:**

Контрольне розпізнавання декількох невідомих об'єктів і візуалізація

unknown\_objects = generate\_objects(10)

predictions = clf.predict(unknown\_objects)

show\_boundaries()

# Відображення результатів розподілення

class\_keys = list(class\_dict.keys())

added\_labels = []

for obj, result in zip(unknown\_objects, predictions):

  if result == 0:

    key = class\_keys[0]

  elif result == 1:

    key = class\_keys[1]

  else:

    key = class\_keys[2]

  if key not in added\_labels:

    added\_labels.append(key)

    label = key

  else:

    label = None

  plt.scatter(obj[0], obj[1], c=markers\_dict[key][0], marker=markers\_dict[key][1], label=label)

plt.title("Результат розпізнавання декількох невідомих об'єктів")

plt.grid()

plt.legend()

plt.show()

**Результат:**

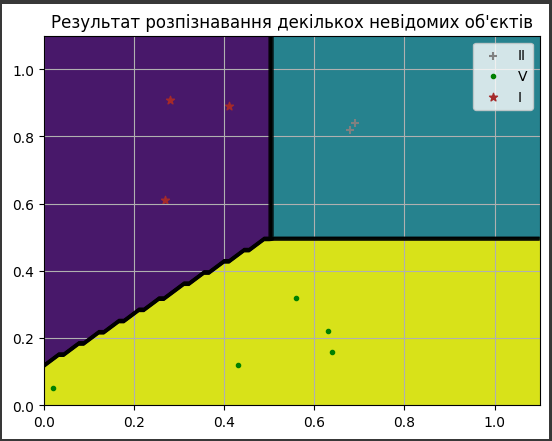


Рисунок 4 - Результат розпізнавання декількох невідомих об'єктів

Висновок: В ході виконання лабораторної роботи був створений та впроваджений алгоритм, який дозволяє визначати кордони між класами.