Міністерство освіти і науки України

Національний університет «Одеська політехніка»

Інститут комп'ютерних систем

Кафедра інформаційних систем

Лабораторна робота №6

З дисципліни: "Інтелектуальний аналіз даних"

Тема: " Дослідження радіально-базисних мереж"

Підготувала студентка

Групи АІ – 212

Козуб К.О

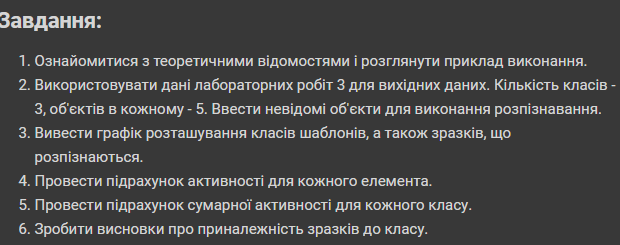
Перевірили:

Антощук С.Г

Кошутіна Д.В

Одеса 2023

Мета: Вивчити структуру і принцип роботи радіально-базисних мереж



<https://colab.research.google.com/drive/17Rhx_Wu5kA_upjjni5-yLJ5BA5XVMWVJ?usp=sharing>

Хід роботи

Результат:

template layer:

8.530086426835482e-20

3.9214800009045494e-14

3.6359916207800256e-22

9.293052930248007e-15

5.651501597411007e-25

4.937269850390672e-11

2.047527392588059e-17

9.832178985238457e-15

5.667336497023763e-11

1.0435492281397301e-14

1.8805488618553097e-15

2.9269141327145914e-11

3.8551803343303357e-13

2.4343488046889167e-12

8.636982965065227e-15

output layer:

4.8507938604322085e-14

1.0606635162068492e-10

3.2099525697094785e-11

point: [0.5134025 0.65252277]

in class 2

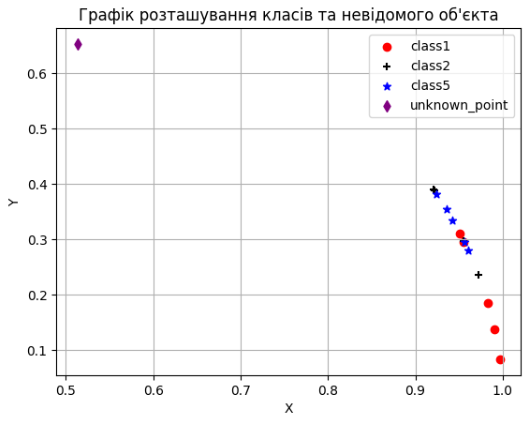


Рисунок 1 - Графік розташування класів та невідомого об'єкта

Висновок: На лабораторній роботі 6. Я вивчила структуру і принцип роботи радіально-базисних мереж

Додаток А

import math

import numpy.random as nprandom

import numpy as np

import matplotlib.pyplot as plt

# Функція для обчислення активності

def func\_active(sigma, W01, x):

    temp\_val = []

    for i in range(W01.shape[0]):

        # Обчислюємо відстань між шаблонами та невідомим об'єктом

        val = ((W01[i, 0] - x[0])\*\*2 + (W01[i, 1] - x[1])\*\*2) / sigma\*\*2

        # Розраховуємо активність на основі відстані

        temp\_val.append(math.exp(-val))

    return temp\_val

# Клас для радіальних базисних функцій (RBF)

class my\_RBF:

    def \_\_init\_\_(self, W01, sigma, n\_classes):

        self.W01 = W01  # Шаблони

        self.sigma = sigma  # Параметр sigma

        self.n\_classes = n\_classes  # Кількість класів

        self.temp\_layer = np.zeros(W01.shape[0])  # Тимчасовий шар активності шаблонів

        self.output\_layer = np.zeros(n\_classes)  # Вихідний шар

    def rbf(self, point):

        # Вхідний шар

        input\_layer = point

        # Обчислення активності шаблонів

        self.temp\_layer = func\_active(self.sigma, self.W01, input\_layer)

        # Обчислення вихідного шару для кожного класу

        for i in range(self.n\_classes):

            i1 = int(5 \* i)

            # Використовуємо активність шаблонів для розрахунку вихідного шару

            self.output\_layer[i] = sum(self.temp\_layer[i1:i1 + 5])

# Дані для класів class1, class2 та class5

class1 = np.array([(0.98277696, 0.18479567),

                  (0.95077496, 0.30988221),

                  (0.99059557, 0.13682259),

                  (0.95544248, 0.29517735),

                  (0.99658631, 0.08255745)])

class2 = np.array([(0.92198211, 0.38723248),

                  (0.97191304, 0.23534029),

                  (0.95526586, 0.29574843),

                  (0.92131686, 0.38881262),

                  (0.95507881, 0.29635192)])

class5 = np.array([(0.9602459,  0.27915552),

                  (0.92446161, 0.38127513),

                  (0.94264749, 0.33378993),

                  (0.93538826, 0.35362239),

                  (0.95567104, 0.29443651)])

# Задаємо діапазон для генерації випадкових координат

x\_min, x\_max = 0.0, 1.0

y\_min, y\_max = 0.0, 1.0

# Генеруємо випадковий невідомий об'єкт

point = np.array([nprandom.uniform(x\_min, x\_max), nprandom.uniform(y\_min, y\_max)])

# Переініціалізуємо мережі під мої дані

sigma = 0.1

n\_classes = 3

W01 = np.vstack((class1, class2, class5))

rbf1 = my\_RBF(W01, sigma, n\_classes)

# Обчислення активності та класифікація невідомого об'єкта

rbf1.rbf(point)

# Виведення результатів активності

temp\_layer = rbf1.temp\_layer

output\_layer = rbf1.output\_layer

print("\n template layer:")

for i in range(len(temp\_layer)):

    print("\n", temp\_layer[i])

print("\n output layer:")

for i in range(len(output\_layer)):

    print("\n", output\_layer[i])

# Виведення приналежності невідомого об'єкта до класу

point\_label = rbf1.output\_layer.argmax() + 1

print("\n point: ", point)

print('in class', point\_label)

plt.scatter(class1[:, 0], class1[:, 1], label="class1", c="red")

plt.scatter(class2[:, 0], class2[:, 1], label="class2", c="black", marker = "+")

plt.scatter(class5[:, 0], class5[:, 1], label="class5", c="blue", marker = "\*")

plt.scatter(point[0], point[1], label="unknown\_point", c="purple", marker="d")

plt.xlabel("X")

plt.ylabel("Y")

plt.legend()

plt.title("Графік розташування класів та невідомого об'єкта")

plt.grid(True)

plt.show()