КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Кафедра інтелектуальних та інформаційних систем

Лабораторна робота № 4

з дисципліни

“Нейромережні технології та їх застосування”

Виконав студент
групи КН-31
Пашковський Павло Володимирович

Київ-2021

Контрольні питання

1. **Що таке перцептрон? Архітектура персептрона.**

   Персептрон – математична або комп'ютерна модель сприйняття інформації мозком.

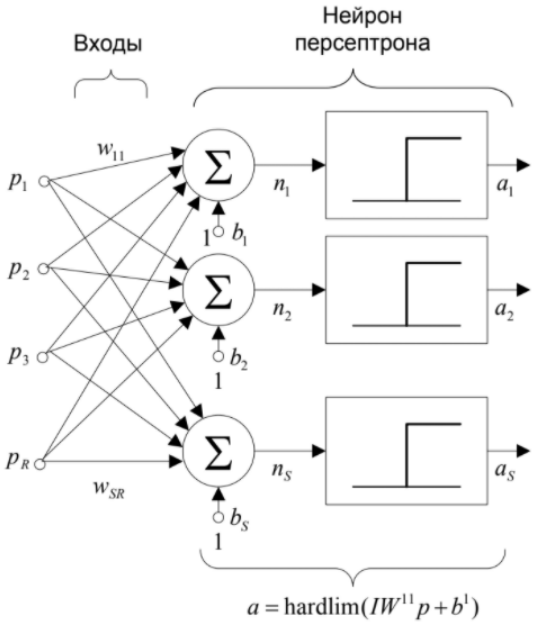

Рис 1. Архітектура персептрона

2. **Які алгоритми навчання персептрона ви знаєте?**

   Метод зворотного поширення помилки, корекції помилки.

3. **Охарактеризуйте правила налаштування параметрів персептрона.**

   Навчання персептрона відбувається з використанням навчальної множини. Позначимо через p вектор входів персептрона, а через t – вектор відповідних бажаних виходів. Мета навчання зменшити помилку e=t-a, яка рівна різниці між вектором виходу a та вектором цілі t.

   Налаштування параметрів персептрона відбувається за допомогою формули:

$$\begin{cases} W^{New} = W^{old} + e * p^T \\ b^{New} = b^{old} + e \end{cases}$$

4. **Охарактеризуйте процедуру адаптації параметрів персептрона**

Багаторазово використавши функцію sim і step для змін ваг і зрушень персепртрона, можна побудувати роздільну лінію, яка будет вирішувати завдання класифікації за умови, що дані є лінійно роздільні. Кожна реалізація процесу налаштування множини називається епохою або циклом.

5. **Які переваги та недоліки персептрона?**

   Вихід песептрона може дорінювати лише 0 або 1. Персептрони можуть вирішувати завдання класифікації тільки для лінійно роздільних наборів векторів.

Індивідуальне завдання:

1. Провести класифікацію векторів за варіантом в таблиці

Варіант 6

Координати точок: (-0.5;-0.5), (-1;0), (0;0.5), (0.5;1)

Номера точок першого класу: 1,2

Номера точок другого класу: 3,4

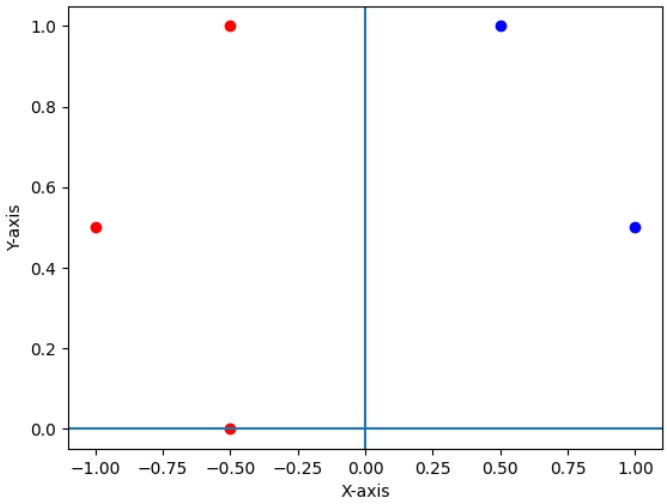

Рис. 1. Побудова графіку.

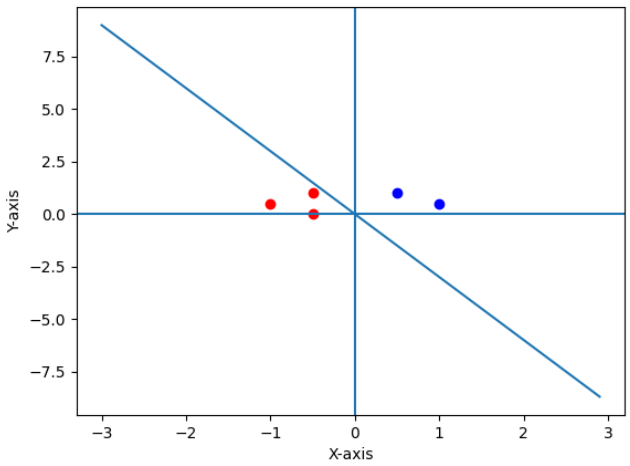

Рис. 2. Побудова графіку з роздільною лінією.

2. Провести класифікацію векторів за допомогою генерації

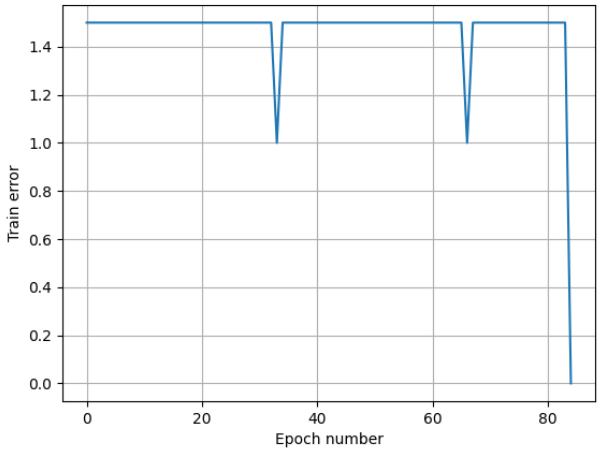

Рис. 3. Графік помилок при навчанні.

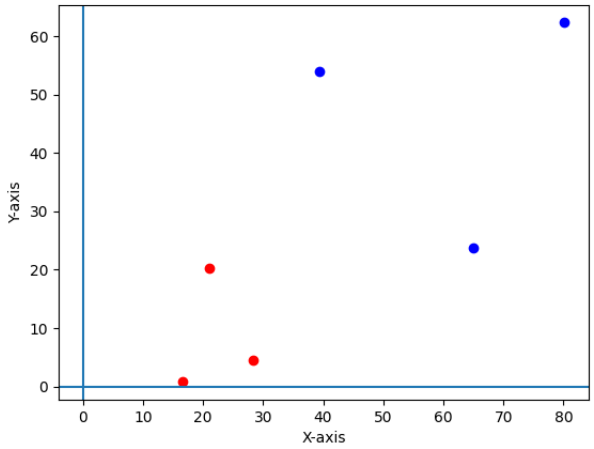

Рис. 4. Побудова графіку

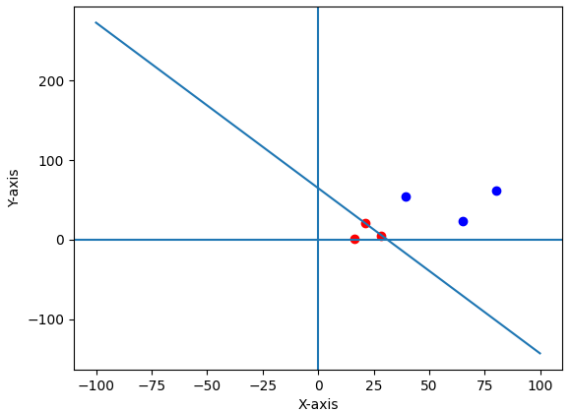

Рис. 5. Побудова графіка з роздільною лінією.

Табл. 1. Вхідні параметри.

X	Y
28.42112224 4.54296811	1
16.54585442 0.76991332	1
39.43321771 53.90530614	0
80.17723979 62.31385946	0

Табл. 2. Параметри вагів.

W1	-0.050411503210692626
W2	-0.024198686852762972
W3	1.5700000000000012

Табл. 3.Результат класифікації вектора.

X1	54.93675337
Y1	69.37873703
Результат	0

Висновок:

У даній роботі було створено програму для класифікації векторів, що подаються на вхід. Навчання відбувалося за допомогою бібліотеки Neuralab.

Результати програми можна практично застосовувати для розділення класів.

Отримані навички щодо навчання перцептрона можна використовувати в майбутньому для аналізу та класифікації даних.

Код програми:

```
import neurolab as nl
import pylab as pl
import numpy as np
import os


def plot_errors(errors):
    pl.plot(errors)
    pl.xlabel('Epoch number')
    pl.ylabel('Train error')
    pl.grid()
```

```
    pl.show()

def plot_graph(x, y, axis_bound=None, net=None):
    pl.figure(0)
    for xi,yi in zip(x, y):
        if yi[0] == 1:
            pl.scatter(xi[0], xi[1], c='r')
        else:
            pl.scatter(xi[0], xi[1], c='b')
    pl.xlabel('X-axis')
    pl.ylabel('Y-axis')
    pl.axvline(x=0)
    pl.axhline(y=0)

    if net is not None:
        x_axis = np.arange(-axis_bound, axis_bound, 0.1)
        w1 = net.layers[0].np['w'][0][0]
        w2 = net.layers[0].np['w'][0][1]
        print(w1)
        print(w2)
        print(net.layers[0].np['b'][0])
        print('------')
        m = -w1/w2
        c = - net.layers[0].np['b'][0] / w2
        pl.plot(x_axis, m*x_axis + c, label="decision boundary")
    pl.show()


x = [[-0.5, -0.5], [-1, 0], [0, 0.5], [0.5, 1]]
y = [[0], [0], [1], [1]]

plot_graph(x, y)
if os.path.exists('1nn.net'):
    load_net = nl.load('1nn.net')
    plot_graph(x, y, 3, load_net)
    x1  = float(input("Input first coord: "))
    x2  = float(input("Input second coord: "))
    print(f"PREDICTION: {load_net.sim([[x1,x2]]).T}")

else:
    net = nl.net.newp([[-2,2], [-2,2]], 1)
    errors = net.train(input=x, target=y, epochs=10, show=True)
    net.save('1nn.net')
    plot_errors(errors)
```

```
import neurolab as nl
import numpy as np
import os
import pylab as pl


def plot_errors(errors):
    pl.plot(errors)
    pl.xlabel('Epoch number')
    pl.ylabel('Train error')
    pl.grid()
    pl.show()

def plot_graph(x,y,axis_bound=None,net=None):
    pl.figure(0)
    for xi,yi in zip(x,y):
```

```
        if yi[0] == 1:
            pl.scatter(xi[0],xi[1],c='r')
        else:
            pl.scatter(xi[0],xi[1],c='b')
    pl.xlabel('X-axis')
    pl.ylabel('Y-axis')
    pl.axvline(x=0)
    pl.axhline(y=0)

    if net is not None:
        x_axis = np.arange(-axis_bound, axis_bound, 0.1)
        w1 = net.layers[0].np['w'][0][0]
        w2 = net.layers[0].np['w'][0][1]
        print(w1)
        print(w2)
        print(net.layers[0].np['b'][0])
        print('------')
        m = -w1/w2
        c = - net.layers[0].np['b'][0] / w2
        pl.plot(x_axis, m*x_axis + c, label="decision boundary")
    pl.show()


def display_variable_info(X,Y,P,T):
    print("*-------------------------*")
    print(f"X: {X}\tshape: {X.shape}")
    print(f"Y: {Y}\tshape: {Y.shape}")
    print(f"Input: {P}\tshape: {P.shape}")
    print(f"Target: {T}\tshape:{T.shape}")


if os.path.exists('2nn.net'):
    load_net = nl.load('2nn.net')
    x1 = float(input("Input first coord: "))
    x2 = float(input("Input second coord: "))
    print(f"PREDICTION: {load_net.sim([[x1,x2]]).T}")
else:
    n1 = 3
    n2 = 3
    x0 = 10
    alpha = 15
    y0 = 50
    beta = 20
    X = np.random.normal(x0,alpha,(n1,2))
    Y = np.random.normal(y0,beta,(n2,2))
    P = np.concatenate((X,Y))
    T = np.zeros((1, len(P)))

    T[0][:n1] = np.ones(n1)
    T = T.transpose()
    display_variable_info(X,Y,P,T)

    min_value = np.min(P)
    max_value = np.max(P)

    net = nl.net.newp([[min_value, max_value],[min_value, max_value]],1)

    errors = net.train(input=P, target=T, epochs=100, show=True)
    print(net.sim(P))
    plot_errors(errors)
    plot_graph(P,T)
    plot_graph(P,T,100,net)
    net.save('2nn.net')
```

```
import numpy as np
import neurolab as nl


alpha = 15
beta = 20
x1 = 45
y1 = 50

x1 = np.random.normal(x1,beta,1)
y1 = np.random.normal(y1,beta,1)

p = np.concatenate((x1,y1))
p = p.reshape(1,2)
print(p)
net = nl.load('2nn.net')
print(net.sim(p))
print(net.layers[0].np['w'])
```