# КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА



## ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

## **Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №4**

# з курсу **«Інтелектуальні системи»**

*Студента 4 курсу*

*Групи ПП-41*

*спеціальності 122*

*«Комп'ютерні науки»*

*ОП«Прикладне програмування»*

*Селецького Віктора Романовича*

## Київ 2023

# Тема

Алгоритми навчання нейромереж

# **Мета**

Використовуючи вбудовані бібліотеки мови програмування Python, створити нейромережі з реалізацією навчання за алгоритмами прямого розповсюдження помилки та зворотнього розповсюдження помилки.

# Завдання

1. Прорахувати одну ітерацію циклу навчання по Δ -правилу одношарової бінарної однорідної нейронної мережі, що складається з 2 нейронів і має лінійну функцію активації (k =0,6). В якості навчальної вибірки використовувати таблицю істинності для операцій кон’юнкції і диз’юнкції (не використовувати перший рядок таблиці). Синаптичні ваги задати випадковим чином.
2. Прорахувати одну ітерацію циклу навчання методом зворотного поширення помилки багатошарової бінарної однорідної нейронної

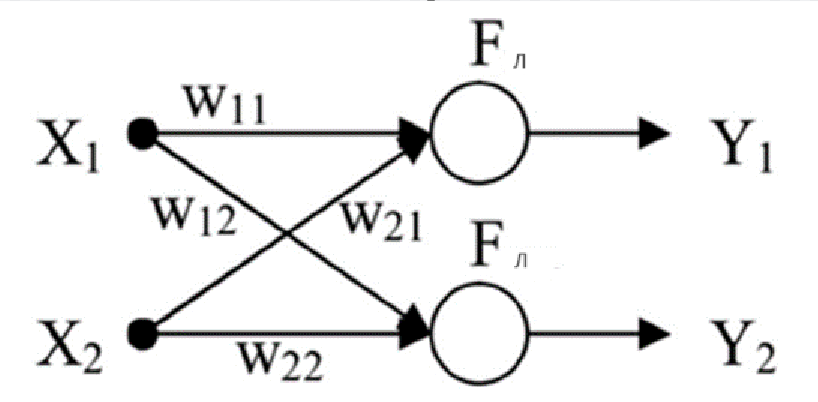
мережі, що складається з 2 шарів, причому в першому шарі знаходиться 2 нейрона, а в другому - 1. Функція активації нейронів мережі -

сигмоїдальна (k = 1) функція. Для навчальної вибірки використовувати таблицю істинності для операції імплікації (не використовувати

перший рядок таблиці). Синаптичні ваги задати випадковим чином.

**Виконання вправи 1**

Структура мережі:



Матриця ваг:

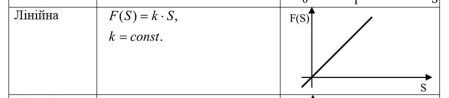
|  |  |  |
| --- | --- | --- |
|  | 1 | 2 |
| 1 | 0,5 | 0,6 |
| 2 | 0,7 | 0,2 |

Навчальна вибірка:

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | D1 | D2 |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

Вектор навчання - 1001 (вектор X = {1,0 }, вектор D = {0,1})

Обчислення вихідних значень відповідно до формули лінійної формули активації (F(S) =k \* S):



Обчислення розбіжності очікуваного і реального результатів:

Додамо коефіцієнт навчання і змінимо ваги:



Нехай Ƞ = 0.5, тоді:

|  |  |  |
| --- | --- | --- |
|  | 1 | 2 |
| 1 |  |  |
| 2 | 0,7 | 0,2 |

Середьноквадратична помилка:

Програмна реалізація:

import numpy as np

weights = np.array([[0.5, 0.7], [0.6, 0.2]])

X = np.array([1, 0])

D = np.array([0, 1])

learning\_rate = 0.5

def linear\_activation(x, k=0.6):

return k \* x

neuron1\_output = linear\_activation(np.dot(X, weights[0]))

neuron2\_output = linear\_activation(np.dot(X, weights[1]))

errors = D - np.array([neuron1\_output, neuron2\_output])

new\_weights = weights - learning\_rate \* np.outer(errors, X)

weights = new\_weights

print("Y1:", round(neuron1\_output,3))

print("Y2:", round(neuron2\_output,3))

print("E1:", round(errors[0],3))

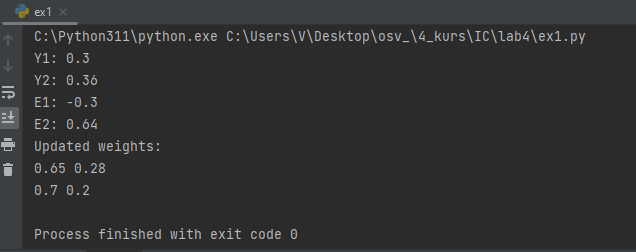
print("E2:", round(errors[1],3))

print("Updated weights:")

print(round(weights[0,0],3), round(weights[1,0],3) )

print(weights[0,1], weights[1,1] )

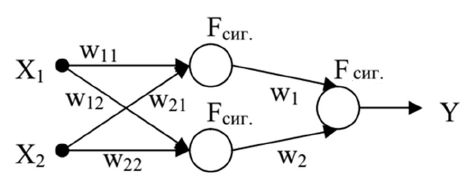
Результат виконання коду:



Результат збігається.

**Виконання вправи 2**

Структура мережі:



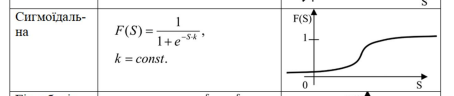
Матриці ваг:

|  |  |  |
| --- | --- | --- |
|  | 1 | 2 |
| 1 | 0,5 | 0,3 |
| 2 | 0,4 | 0,8 |

|  |  |  |
| --- | --- | --- |
|  | 1 | 2 |
| 1 | 0,8 | 0,3 |

|  |  |  |
| --- | --- | --- |
| X1 | X2 | D1 |
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Навчальна вибірка (операція - імплікація, вектор 011)



Прямий прохід:

k = 1

Зворотній прохід:

Нові ваги:

|  |  |  |
| --- | --- | --- |
|  | 1 | 2 |
| 1 | 0,5 | 0,3 |
| 2 | 0,707 | 0,202 |

|  |  |  |
| --- | --- | --- |
|  | 1 | 2 |
| 1 |  |  |

Помилка:

Програмна реалізація:

import numpy as np

weights\_input = np.array([[0.5, 0.3], [0.4, 0.8]])

weights\_output = np.array([0.8, 0.3])

print("\nSynaptic weights layer 1:")

print(weights\_input)

print("\nSynaptic weight layer 2:")

print(weights\_output)

X = np.array([0, 1])

D = np.array([1])

learning\_rate = 0.5

def sigmoid(s):

return 1 / (1 + np.exp(-s))

neuron1\_output = sigmoid(np.dot(X, weights\_input))

neuron2\_output = sigmoid(np.dot(neuron1\_output, weights\_output))

er2\_ = (D - neuron2\_output) \* neuron2\_output \* (1 - neuron2\_output)

new\_weights\_output = weights\_output - learning\_rate \* neuron1\_output\*er2\_

er1\_ = neuron1\_output \* (1 - neuron1\_output) \* er2\_\*weights\_output

new\_weights\_input = weights\_input + learning\_rate \*np.outer(X, er1\_)

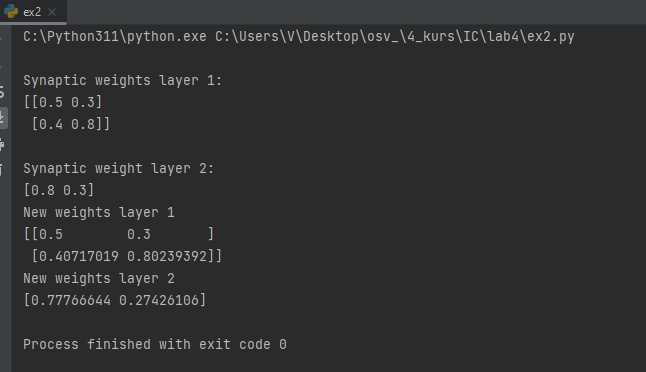
print("New weights layer 1")

print(new\_weights\_input)

print("New weights layer 2")

print(new\_weights\_output)

Результат:



З урахуванням округлень можна вважати, що результати збігаються.

**Висновок**

В результаті виконання даної лабораторної роботи, я, використовуючи вбудовані бібліотеки мови програмування Python, створив нейромережі з реалізацією навчання за алгоритмами прямого розповсюдження помилки та зворотнього розповсюдження помилки.