Міністерство освіти і науки України  
НТУУ «КПІ ім. І. Сікорського»

Кафедра автоматизації проектування енергетичних процесів і систем

ЛАБОРАТОРНА РОБОТА № 1  
з дисципліни «МЕТОДОЛОГІЯ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНИХ КОМП’ЮТЕРНИХ ПРОГРАМ»

Тема: «Штучні нейронні мережі. Моделювання формальних логічних функцій.

Прогнозування часових рядів»

ТІ-92 Черноусов Денис

Перевірив д.т.н Мусієнко А. П.

КИЇВ 2021

**Мета роботи:**

Отримати початкові навички щодо створення штучних нейронних

мереж, що здатні виконувати прості логічні функції, та нейронних мереж, що

здатні прогнозувати часові ряди.

**Опис дії однієї (з конкретними значеннями) епохи при використанні алгоритму зворотного поширення (back propagation):**

Вхідні дані :

X1 = 0.58

X2 = 3.38

X3 = 0.91

Вихідні дані :

Y = 5.8

Початкове значення усіх ваг (зображені на малюнку нище) :

Wij = random(-0.1, 0.1)

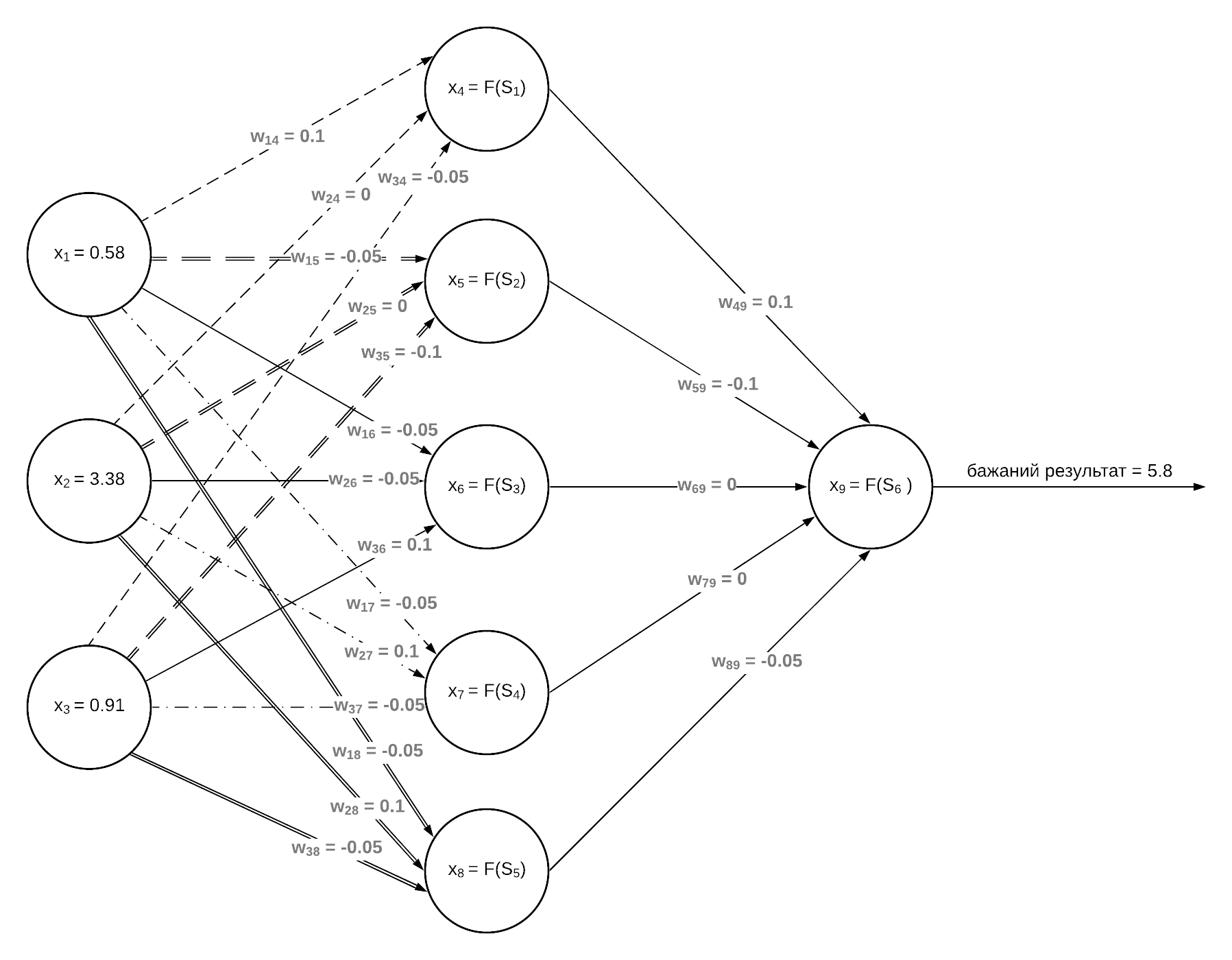
Коефіцієнт навчання:

V = 0.01

Кількість нейронів в прихованому шарі:

NUM\_NEURONS = 5

Функція активації: Сигмоїда з маштабом рівним 10, F(x) =

Графічне зображення нейроної мережі

1. **Етап переднього ходу**

Загалом, формула для обчилення суми на вхід до нейрона матиме формулу:

Знаходимо суму на вхід для нейрона №4 (самий верхній нейрон у схованому шарі) за формулою :

S1 = = x1 \* w14 + x2 \* w24 + x3 \* w34 = 0.58 \* 0.1 + 0 \* 3.38 + 0.91 \* (-0.05) = = 0.058 + -0.0455 = 0.0125

За допомогою суми знаходимо значення нейрону №4 :

X4 = F(S4) = 5.03125

Знайдемо значення виходу інших нейронів для інші значення :

S2 = 0.58 \* -0.05 + 0 \* 3.38 + 0.91 \* (-0.1) = -0.12

X5 = 4.70036

S3 = 0.58 \* -0.05 + -0.05 \* 3.38 + 0.91 \* 0.1 = -0.107

X6 = 4.73275

S4 = 0.58 \* -0.05 + 0.1 \* 3.38 + 0.91 \* (-0.05) = 0.2635

X7 = 5.65496

S5 = 0.58 \* -0.05 + 0.1 \* 3.38 + 0.91 \* (-0.05) = 0.2635

X8 = 5.65496

Зі здобутих значень нейронів в прихованому шарі знайдемо значення на вхід на прогнозоване значення :

S6 = = 0.1 \* 5.03125 + (-0.1) \* 4.70036 + 0 \* 4.73275 +0 \* 5.65496 +(-0.05) \* 5.65496 = -0.249659

X9 = 4.37907

1. **Етап заднього ходу (backpropogation)**

Знаходимо помилку прогнозу:

E = X9 – Y = 4.37907 – 5.8 = -1.420925

Знаходимо помилку нейронів за формулою :

D4 = = - 1.420925 \* 0.1 = - 0.1420925

D5 = = - 1.420925 \* (-0.1) = 0.1420925

D6 = = - 1.420925 \* 0 = 0

D7 = = - 1.420925 \* 0 = 0

D8 = = - 1.420925 \* -0.05 = 0.07104625

Знаходимо похідну сигмоїдальної функції:

Присвоюємо нові вагові коефіцієнти починаючи з вагів, що перед прихованим слоєм, за формулою:

wij = wij - V \* \* Dj \* xi

w14 = w14 - V \* \* D4 \* x1 = 0.1 – 0.01 \* \* (-0.1420925) \* 0.58 = 0.100206

w24 = w24 - V \* \* D4 \* x2 = 0 – 0.01 \* \* (-0.1420925) \* 3.38 = 0.00120063

w34 = w34 - V \* \* D4 \* x3 = -0.05 – 0.01 \* \* (-0.1420925) \* 0.91 = -0.0496768

w15 = w15 - V \* \* D5 \* x1 = -0.05 – 0.01 \* \* 0.1420925 \* 0.58 = -0.0502053

w24 = w24 - V \* \* D5 \* x2 = 0 – 0.01 \* \* 0.1420925 \* 3.38 = -0.0011964

w34 = w34 - V \* \* D5 \* x3 = -0.1 – 0.01 \* \* 0.1420925 \* 0.91 = -0.100322

w16 = w16 - V \* \* D6 \* x1 = **-0.05** – 0.01 \* \* 0 \* 0.58 = -0.05

w26 = w26 - V \* \* D6 \* x2 = **-0.05** – 0.01 \* \* 0 \* 3.38 = -0.05

w36 = w36 - V \* \* D6 \* x3 = **0.1** – 0.01 \* \* 0 \* 0.91 = 0.1

w17 = -0.05 + 0 = -0.05

w27 = 0.1 + 0 = 0.1

w37 = -0.05 + 0 = -0.05

w18 = w16 - V \* \* D6 \* x1 = -0.05 – 0.01 \* \* 0.07104625 \* 0.58 =

-0.0501012

w28 = w26 - V \* \* D6 \* x2 = 0.1 – 0.01 \* \* 0.07104625 \* 3.38 =

0.09941

w38 = w36 - V \* \* D6 \* x3 = -0.05 – 0.01 \* \* 0.07104625 \* 0.91 =

-0.0501589

Присвоємо значення вагам після прихованого шару :

W49 = w49 - V \* \* E \* x4 = 0.1 – 0.01 \* \* (-1.420925) \* 5.03125 = 0.117597

W59 = w59 - V \* \* E \* x5 = -0.1 – 0.01 \* \* (-1.420925) \* 4.70036 =

-0.0835604

W69 = w69 - V \* \* E \* x6 = 0 – 0.01 \* \* (-1.420925) \* 4.73275 =

0.0165529

W79 = w79 - V \* \* E \* x7 = 0 – 0.01 \* \* (-1.420925) \* 5.65496 =

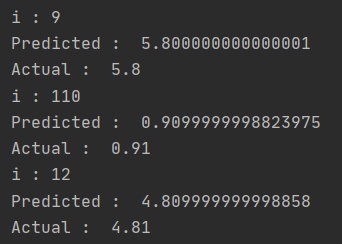
0.0197784

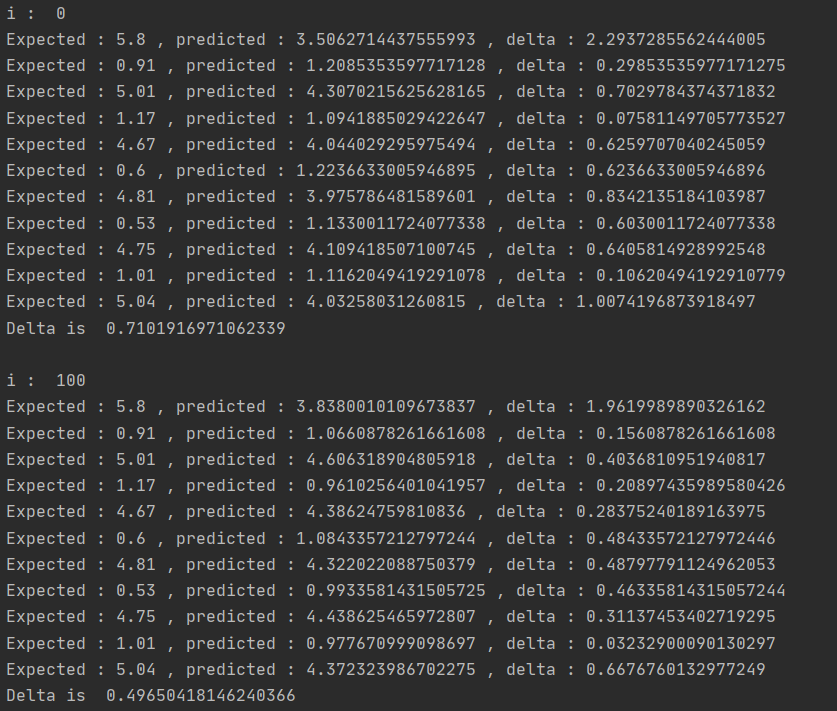
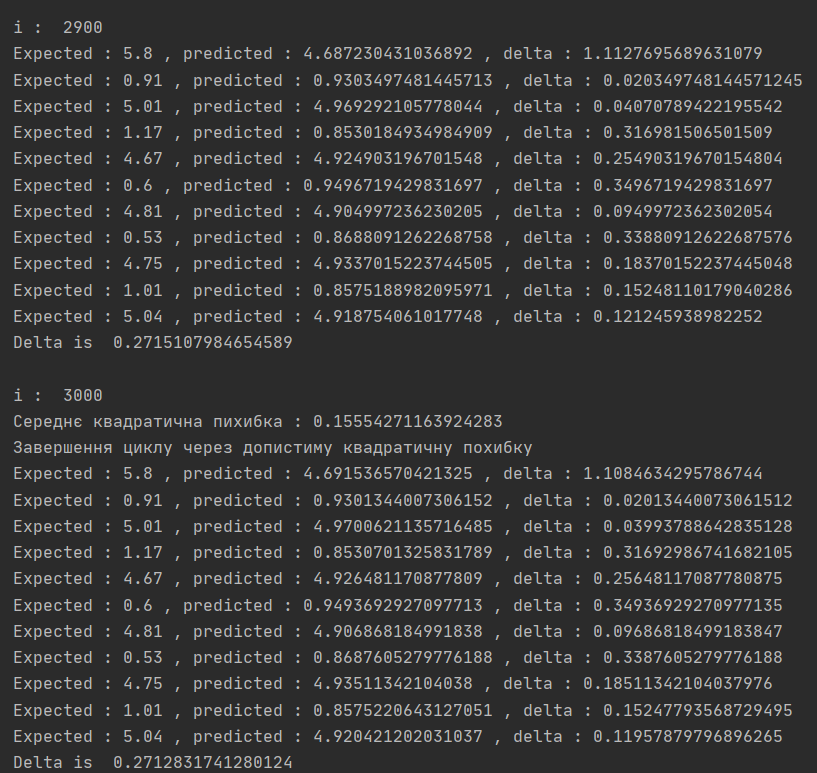
W89 = w79 - V \* \* E \* x7 = -0.05 – 0.01 \* \* (-1.420925) \* 5.65496 =

-0.0302216

Епоха завершена

**Скріншоти результатів навчання нейронної мережі**

Тестування нейронної мережі на малому ряді чисел. Мережа навчається тільки на 3-ох вхідних та 1-ому вихідному. В результаті мережа пробує вгадати за 3-ома тими самими вхідними даними те саме 1-не вихідне число зі заданою точністю.

Початок навчання нейромережі. Достатньо різке зменшення середньої похибки

Кінець навчання

**Код програми**

main.py

from logicals import test\_logical  
from row import \*  
  
test\_logical()  
  
row16 = [0.58, 3.38, 0.91, 5.8,  
 0.91, 5.01, 1.17, 4.67,  
 0.6, 4.81, 0.53, 4.75,  
 1.01, 5.04, 1.07]  
  
array16 = []  
array16\_in = []  
for i in range(len(row16) - 3):  
 array16.append(row16[i:i + 4])  
 array16\_in.append(row16[i:i + 3])  
  
print(np.matrix(array16))  
  
teach\_predict\_self(array16[0])  
teach\_predict\_self(array16[1])  
teach\_predict\_self(array16[6])  
  
print("\n")  
  
for i in range(100):  
 print("i : ", i \* 100)  
 stop = teach(array16)  
 predicted = forward\_set(array16\_in)  
 delta = []  
 for i in range(len(array16\_in) - 1):  
 delta.append(abs(predicted[i] - array16[i][-1]))  
 print("Expected :", array16[i][-1], ", predicted :", predicted[i], ", delta :", delta[i])  
 print("Delta is ", np.mean(delta), "\n")  
 if stop: break  
  
print("\nLast element in row :", forward(array16\_in[-1]))

row.py

import math  
import random  
  
import numpy as np  
  
NUM\_NEURONS = 5  
NUM\_INPUT = 3  
V = 0.01  
  
WAGES = [random.uniform(-0.1, 0.1)] \* (NUM\_NEURONS \* NUM\_INPUT + NUM\_NEURONS)  
  
  
def sigmoid\_function(x):  
 return (1 / (1 + math.exp(-x))) \* 10  
  
  
def calculate\_sum(input, start, end):  
 return np.inner(input, WAGES[start:end])  
  
  
def sigmoid\_function\_der(x):  
 return math.exp(x) / ((1 + math.exp(x)) \*\* 2)  
  
  
def forward(input: []):  
 Ss = []  
 Ys = []  
  
 for i in range(NUM\_NEURONS):  
 Ss.append(calculate\_sum(input, len(input) \* i, len(input) \* i + len(input)))  
 Ys.append(sigmoid\_function(Ss[i]))  
  
 Ss.append(calculate\_sum(Ys, NUM\_NEURONS \* len(input), NUM\_NEURONS \* len(input) + NUM\_NEURONS))  
 return sigmoid\_function(Ss[-1])  
  
  
def teach\_by\_row(input: [], output: float):  
 Ss = []  
 Ys = []  
  
 # calculate hidden neurons  
  
 for i in range(NUM\_NEURONS):  
 Ss.append(calculate\_sum(input, len(input) \* i, len(input) \* i + len(input)))  
 Ys.append(sigmoid\_function(Ss[i]))  
  
 # calculate output  
  
 Ss.append(calculate\_sum(Ys, NUM\_NEURONS \* len(input), NUM\_NEURONS \* len(input) + NUM\_NEURONS))  
 Ys.append(sigmoid\_function(Ss[-1]))  
  
 # calculate error  
  
 error = Ys[-1] - output  
 quad\_error = error \*\* 2  
  
 neuron\_error = []  
  
 for i in range(NUM\_NEURONS):  
 neuron\_error.append(error \* WAGES[NUM\_NEURONS \* len(input) + i])  
  
 for i in range(NUM\_INPUT \* NUM\_NEURONS):  
 WAGES[i] -= sigmoid\_function\_der(Ss[i // len(input)]) \* neuron\_error[i // len(input)] \* input[  
 i % len(input)] \* V  
  
 for i in range(NUM\_NEURONS):  
 WAGES[NUM\_NEURONS \* NUM\_INPUT + i] -= sigmoid\_function\_der(Ss[-1]) \* error \* Ys[i] \* V  
  
 return quad\_error  
  
  
def teach\_by\_dataset(dataset):  
 errors = []  
 for i in range(len(dataset)):  
 errors.append(teach\_by\_row(dataset[i][0:-1], dataset[i][-1]))  
  
 return np.mean(errors)  
  
  
def teach\_predict\_self(data):  
 old\_quad\_error = 0  
 for i in range(1000):  
 new\_quad\_error = teach\_by\_row(data[0:-1], data[-1])  
 predicted = forward(data[0:-1])  
 if abs(new\_quad\_error - old\_quad\_error) < 0.1 \*\* 20:  
 print("i :", i)  
 print("Predicted : ", predicted)  
 print("Actual : ", data[-1])  
 return  
 old\_quad\_error = new\_quad\_error  
  
  
def teach(data):  
 old\_error = 10  
 i = 0  
 for i in range(100):  
 new\_error = teach\_by\_dataset(data)  
 if old\_error < new\_error:  
 raise Exception("Увага! Нейронка розбігається, поставте меншу швидкість навчання. V =", V)  
 if abs(old\_error - new\_error) < 0.1 \*\* 5:  
 print("Середнє квадратична пихибка :", new\_error)  
 print("Завершення циклу через допистиму квадратичну похибку")  
 return True  
 old\_error = new\_error  
  
  
def forward\_set(set):  
 predicted = []  
 for i in range(len(set)):  
 predicted.append(forward(set[i]))  
 return predicted

logicals.py

import numpy as np  
  
def logical\_activator(S, T):  
 return 0 if S < T else 1  
  
  
def logical\_function(input: [int, int], wages: [float, float], type: float):  
 return logical\_activator(np.inner(input, wages), type)  
  
  
def logical\_and(input: [int, int], wages=[1, 1]):  
 return logical\_function(input, wages, 1.5)  
  
  
def logical\_or(input: [int, int], wages=[1, 1]):  
 return logical\_function(input, wages, 0.5)  
  
  
def logical\_not(input: int, wages=-1.5):  
 return logical\_function(input, wages, -1)  
  
  
def logical\_xor(input: [int, int]):  
 S1 = logical\_or(input, [1, -1])  
 S2 = logical\_or(input, [-1, 1])  
  
 return logical\_or([S1, S2], [1, 1])  
  
  
def test\_logical():  
 print("Logical and :")  
 print(logical\_and([1, 1]))  
 print(logical\_and([1, 0], [1, 1]))  
 print(logical\_and([1, 1], [1, -1]), "\n")  
  
 print("Logical or :")  
 print(logical\_or([1, 0]))  
 print(logical\_or([0, 0], [1, 1]))  
 print(logical\_or([1, 0], [0, 1]), "\n")  
  
 print("Logical not :")  
 print(logical\_not(0))  
 print(logical\_not(0, -1.5))  
 print(logical\_not(1, -1.5), "\n")  
  
 print("Logical xor :")  
 print(logical\_xor([1, 1]))  
 print(logical\_xor([1, 0]))  
 print(logical\_xor([0, 0]), "\n")

**Висновки**

У ході виконання лабораторної роботи було розроблено моделі логічних функцій АБО, І, НЕ, Виключне АБО і нейронна мережа для передбачення числового ряду. Був вивчений метод навчання нейронної мережі backpropagation. Освоєні безліч термінів машинного навчання для подальшого вивчення цієї сфери. Створена нейронна мережа показала достатньо точний результат в першому тесті, проте для числового ряду середня похибка складала 0.27, що може свідчити про нелінійність даного числового ряду. Тобто, для кращого передбачення числового ряду потребується ускладнити алгоритми та, мабуть, виділити більше комп’ютерних ресурсів.