Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

Кафедра комп'ютерного моделювання процесів і систем

ЗВІТ

з лабораторної роботи №3

"МОДЕЛЮВАННЯ ДВОШАРОВОГО ПЕРСЕПТРОНУ.
НАВЧАННЯ ТА ТЕСТУВАННЯ"

з курсу

«Нейронні мережі та методи проектування»

Виконав: студент групи ІКМ-М222к Черкас Ю.В.

Перевірив: професор, д.т.н. Успенський В.Б.

Харків 2023р

Виконання

Постановка задачі

Призначення мережі, що розглядається, – реалізація логічної операції XOR, яка визначена на двовимірному векторі x=(x1, x2) (задача класифікації). Мережа на виході повинна формувати сигнал y, який дорівнюватиме "0" (клас C1) або "1" (клас C2) відповідно до таблиці істинності (рис. 1).

XOR

N	x_1	x_2	y	
1	0	0	0	C_1
2	0	1	1	C_2
3	1	0	1	C_2
4	1	1	0	C_1

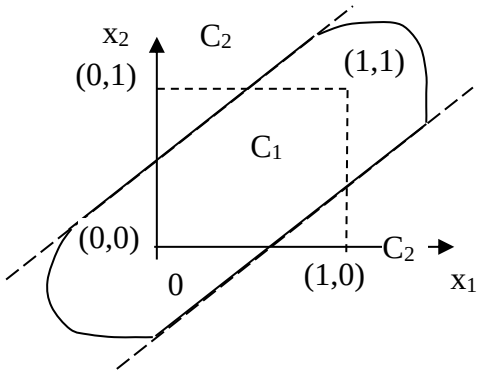

Рисунок 1 – Таблиця істинності функції XOR та розбиття простору вхідних векторів на два класи C1 та C2

Генерація даних для навчання

Для навчання мережі зазвичай потрібно багато даних. Будемо вважати, що клас C_1 утворюють усі точки, що лежать у смузі між прямими $x_2 = x_1 + 0.5$ та $x_2 = x_1 - 0.5$ (рис. 1). Тобто, якщо будь-яка точка (x_1, x_2) задовольняє умові $(x_2 < x_1 + 0.5)$ та $(x_2 > x_1 - 0.5)$, то вона належить до класу C_1. Інакше до класу C_2.

Таким чином, алгоритм генерації даних для навчання генерує файл, у першому стовпці якого вказане значення x_1, у другому x_2, у третьому – еталонне значения параметру.

Точки (x_1, x_2) генеруються випадковим чином в діапазоні $[0\,;1]$ для кожної компоненти. Загальна кількість – 1000. Точки та еталонні значення зберігаються у файлі.

```python
import random

afile = open("random.csv", "w")
n_max = 1000
afile.write("id,x1,x2,d\n")

for i in range(n_max):
    x1 = random.random()
    x2 = random.random()
    d = 1 if (x2 > x1 + 0.5) or (x2 < x1 - 0.5) else 0
    afile.write(f'{i},{x1},{x2},{d}\n')

afile.close()
```

Рисунок 2 – Скріншот програми генерації початкових даних на Python

На рисунку 3 зображений розподіл згенерованих початкових даних для навчання.

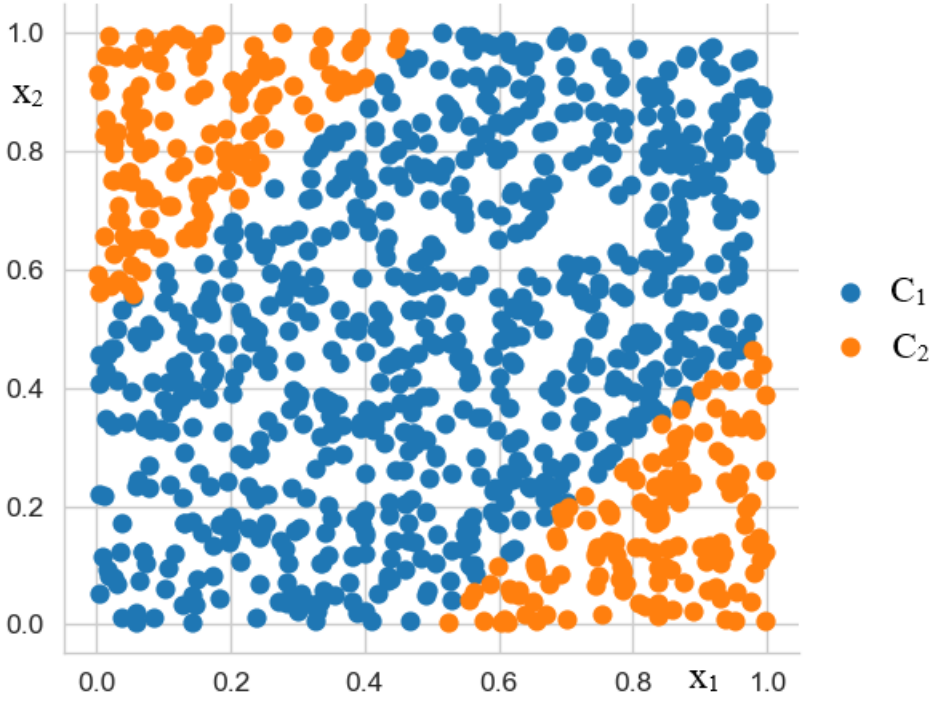

Рисунок 3 – Розподіл згенерованих даних для навчання

Практична частина

1) За наведеним в методичних вказівках алгоритмом промоделювати процес навчання та роботи двошарового персептрону.

Лістинг програми навчання двошарового персептрону на мові Python наведений в Додатку А та Додатку Б.

2) Дослідити вплив початкових умов вагових коефіцієнтів на ефективність навчання: провести моделювання для рекомендованих значень та для інших, вільно обраних значень. Порівнювати варіанти за кількістю епох навчання, необхідних для нормальної роботи мережі.

Проведемо моделювання ефективності навчання двошарового персептрону для наступних початкових значень вагових коефіцієнтів:

- всі вагові коефіцієнти рівні «0» (рис. 4);
- всі вагові коефіцієнти рівні «1» (рис. 5);
- вагові коефіцієнти мають значення: $w_{10}^{(1)} = 0$, $w_{11}^{(1)} = 1$, $w_{12}^{(1)} = -1$, $w_{20}^{(1)} = 0$, $w_{21}^{(1)} = 1$, $w_{22}^{(1)} = -1$, $w_{10}^{(2)} = 0$, $w_{11}^{(2)} = 1$ $w_{12}^{(2)} = -1$ (рис. 6);
- вагові коефіцієнти мають значення рекомендовані методичними вказівками: $w_{10}^{(1)} = 0.1$, $w_{11}^{(1)} = -0.3$, $w_{12}^{(1)} = 0.4$, $w_{20}^{(1)} = -0.7$, $w_{21}^{(1)} = -0.1$, $w_{22}^{(1)} = 0.01$, $w_{10}^{(2)} = 0.4$, $w_{11}^{(2)} - 0.2$, $w_{12}^{(2)} = 0.1$ (рис. 7).

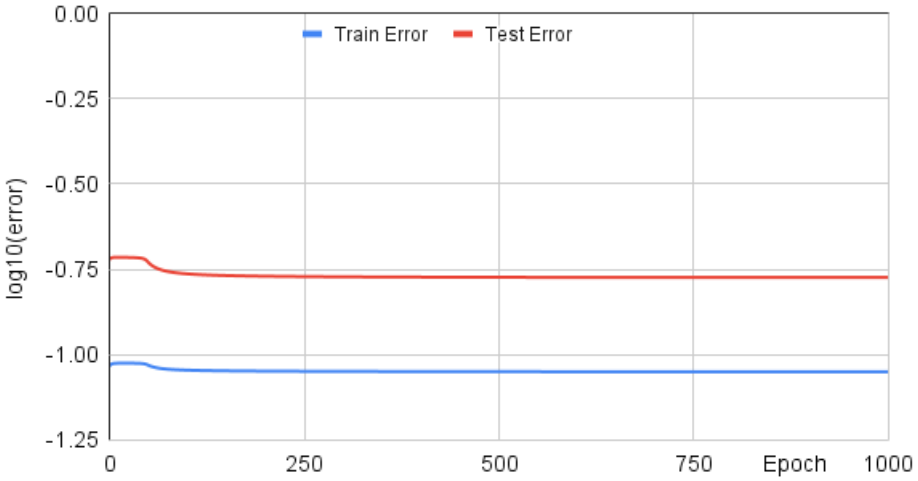

Рисунок 4 – Похибки при всіх вагових коефіцієнтах рівних «0»

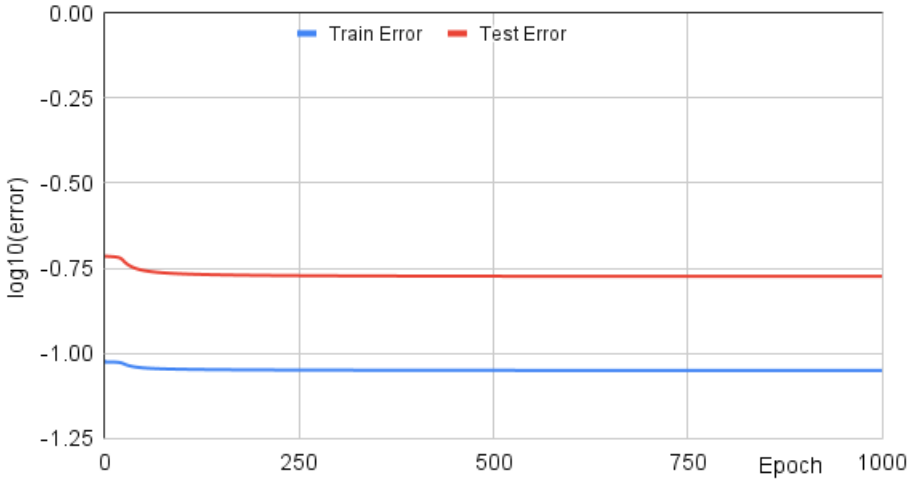

Рисунок 5 – Похибки при всіх вагових коефіцієнтах рівних «1»

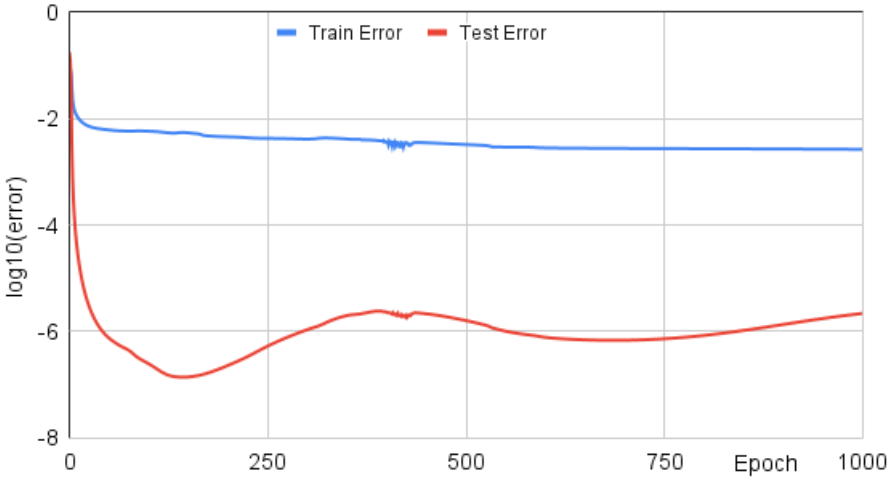

Рисунок 6 – Похибки при $w_{10}^{(1)} = 0$, $w_{11}^{(1)} = 1$, $w_{12}^{(1)} = -1$, $w_{20}^{(1)} = 0$, $w_{21}^{(1)} = 1$, $w_{22}^{(1)} = -1$, $w_{10}^{(2)} = 0$, $w_{11}^{(2)} = 1$ $w_{12}^{(2)} = -1$

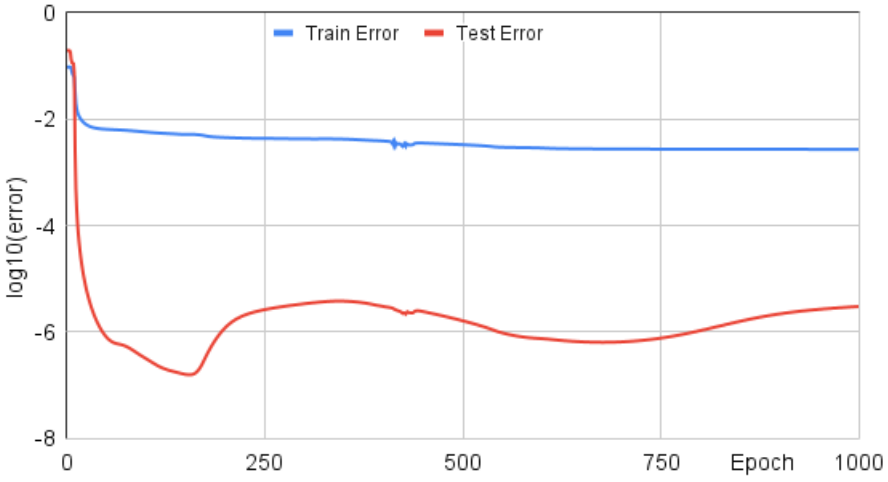

Рисунок 7 – Похибки при рекомендованих значеннях вагових коефіцієнтів

На рисунках 4-7 наведені графіки ефективності навчання двошарового персептрону. По осі абсцис відбувається зміна кількості епох навчання (Epoch). По осі ординат відображений десятинний логарифм середньої похибки (log10(error)) для навчальної вибірки (Train Error) та тестової вибірки (Test Error).

Як бачимо, якщо всі вагові коефіцієнти взяти рівними між собою, то навчання не відбувається, а мережа «зависає» на високому значенні похибки (рис. 4, рис. 5). Проте, якщо вагові коефіцієнти взяти нерівними між собою з випадковими значеннями, то похибка тестової вибірки досить швидко знижується і набуває мінімального значення для $\text{Epoch} \approx 150$ (рис. 6). Аналогічний графік з стрімким зниженням тестової вибірки і стабілізацією при $\text{Epoch} \approx 150$ був отриманий і для рекомендованих початкових значень вагових коефіцієнтів (рис. 7). Всі подальші розрахунки будемо проводити з рекомендованими початковими значеннями коефіцієнтів.

3) Дослідити вплив коефіцієнту швидкості навчання на його ефективність.

За визначенням коефіцієнт навчання η може змінюватися від 0 до 1 включно. Очевидно, що при значенні коефіцієнту навчання $\eta = 0$ мережа не навчається, тому дане значення виключаємо з розгляду. Проведемо дослідження впливу коефіцієнту навчання для наступних його значень:

- $\eta = 0.25$ (рис. 8);
- $\eta = 0.5$ (рис. 9);
- $\eta = 0.75$ (рис. 10);
- $\eta = 1$ (дане значення являється рекомендованим в методичних вказівках, його графік зображений на рис. 7).

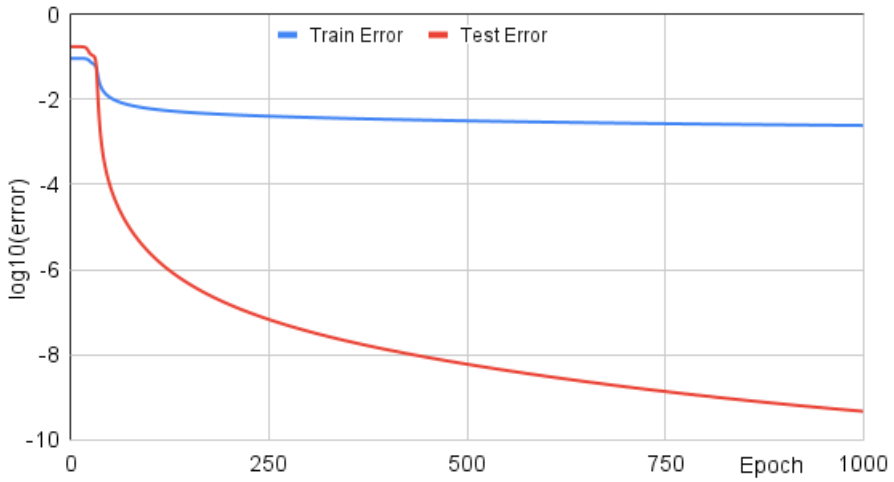

Рисунок 8 – Похибки при коефіцієнті навчання $\eta = 0.25$

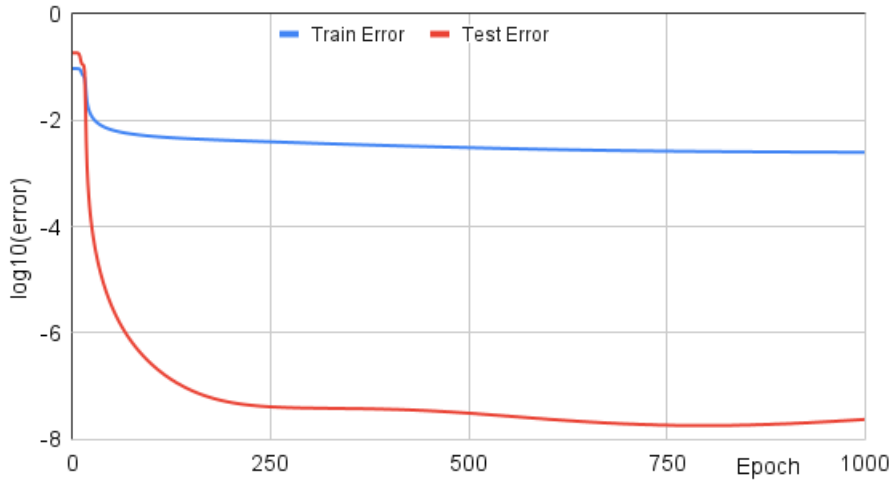

Рисунок 9 – Похибки при коефіцієнті навчання $\eta = 0.5$

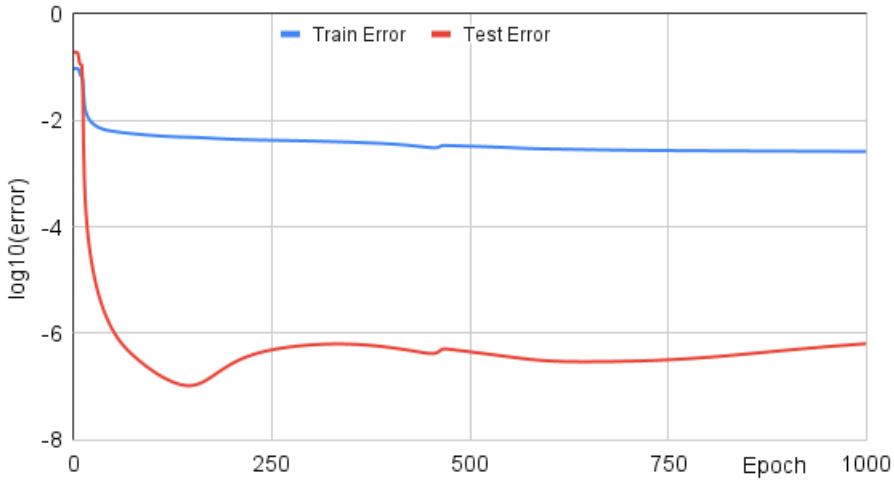

Рисунок 10 – Похибки при коефіцієнті навчання $\eta = 0.75$

Як бачимо з рисунку 8, при $\eta = 0.25$ похибка тестової вибірки плавно зменшується зі збільшені епохи, проте відсутня зміна характеру та зростання даної функції (ймовірно, він би проявився при продовженні навчання за 1000 епох). Для $\eta = 0.5$ мінімум похибки тестової вибірки наступає на $\text{Epoch} \approx 800$ (рис. 9), а її значення менше ніж у попередньому випадку. При $\eta = 0.75$ (рис. 10) як і при $\eta = 1$ (рис. 7) навчання мережі припиняється на $\text{Epoch} \approx 150$.

4) Дослідити залежність ефективності алгоритму навчання від параметра k, що входить до ступеню в функції активації.

Проведемо дослідження впливу параметру k функції активації для наступних його значень:

- $k = 0.5$ (рис. 11);
- $k = 0.75$ (рис. 12);
- $k = 1$ (дане значення являється рекомендованим в методичних вказівках, його графік зображений на рис. 7)
- $k = 1.25$ (рис. 13);
- $k = 1.5$ (рис. 14);

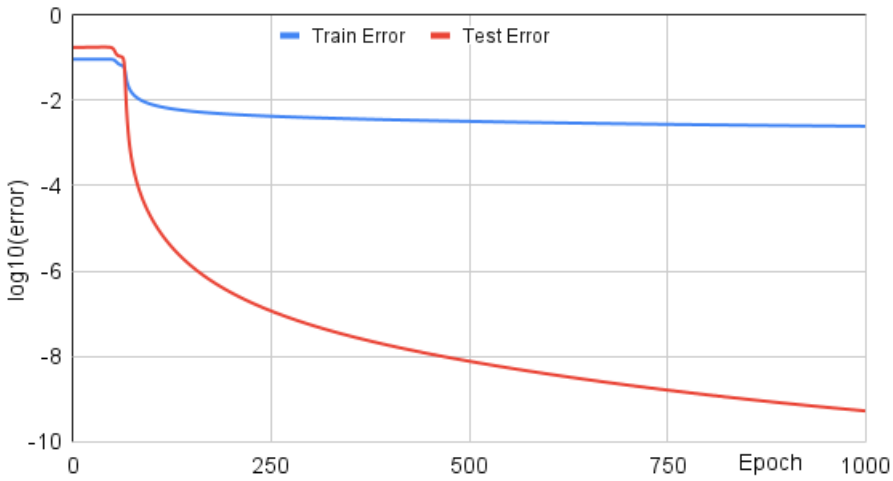

Рисунок 11 – Похибки при $k = 0.5$ в функції активації

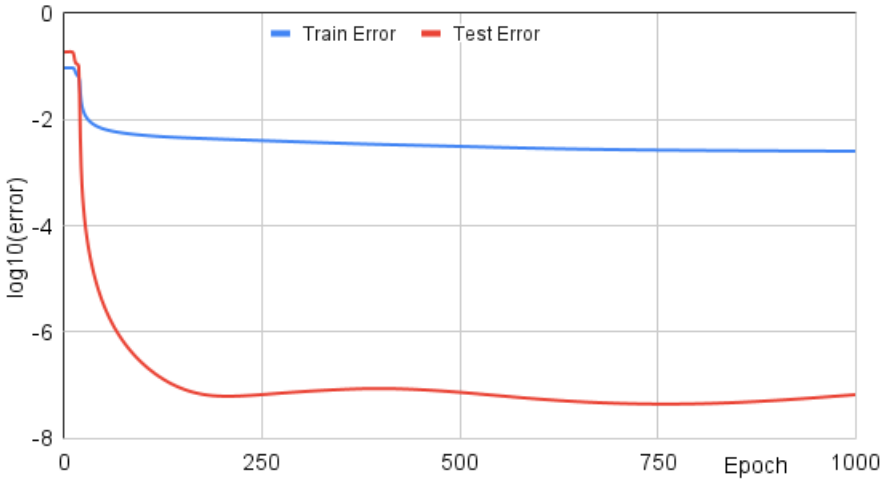

Рисунок 12 – Похибки при $k = 0.75$ в функції активації

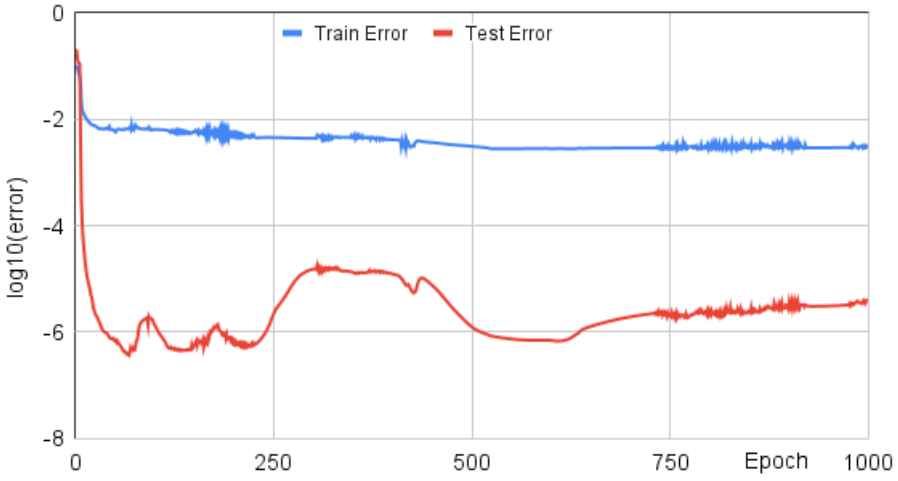

Рисунок 13 – Похибки при $k = 1.25$ в функції активації

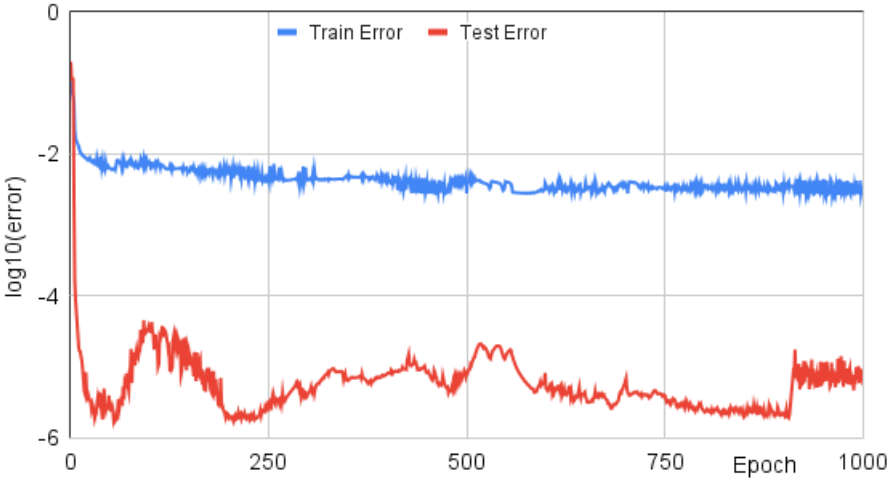

Рисунок 14 – Похибки при $k = 1.5$ в функції активації

Порівнюючи отримані графіки впливу параметру k в функції активації відносно рекомендованого значення $k = 1$ (рис. 7), можна зробити наступні висновки. При $k < 1$, потрібно обчислити більше епох для навчання мережі, проте отримані значення тестової похибки мають менші абсолютні значення. При $k > 1$ навчання відбувається швидше, проте отримані значення похибок гірші. Чим більше k від одиниці, тим крутіша функція активації і тим вона більш чутлива до випадкових викидів і похибок.

5) Для рекомендованого варіанту побудувати залежність E* від кількості епох навчання та визначити оптимальну кількість.

Графік залежності середньої похибки від кількості епох навчання зображений на рисунку 15. Середня похибка тестової вибірки набуває мінімуму при $\text{Epoch} = 155$. Відповідні їй вагові коефіцієнти мають наступні значення: $w_{10}^{(1)} = 10.82$, $w_{11}^{(1)} = -21.68$, $w_{12}^{(1)} = 22.05$, $w_{20}^{(1)} = -10.90$, $w_{21}^{(1)} = -20.17$, $w_{22}^{(1)} = 20.73$, $w_{10}^{(2)} = 7.33$, $w_{11}^{(2)} = -16.05$ $w_{12}^{(2)} = 15.90$.

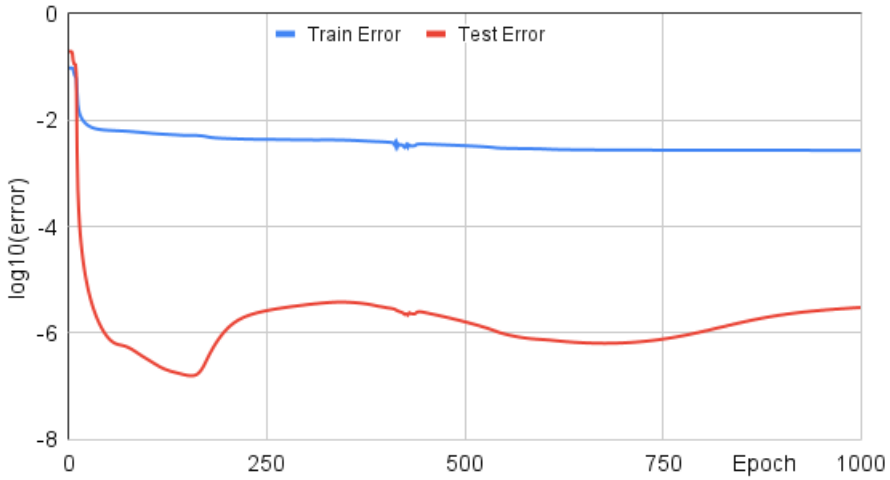

Рисунок 15 – Графік залежності десятинного логарифму середньої похибки від кількості епох навчання для рекомендованих параметрів алгоритму навчання двошарового перспектрону

Висновки

На даній лабораторній роботі ми промоделювали навчання двошарового персептрону для випадку вирішення задачі XOR. Дослідили вплив різних параметрів на ефективність навчання та визначили їх допустимі інтервали та оптимальні значення.

Побачили, що мережа здатна до навчання навіть при виборі початкових значень вагових коефіцієнтів випадковим чином. Проте існують «невдалі» їх комбінації, при яких навчання не відбувається (наприклад, коли вони всі рівні між собою: рис. 4, рис. 5).

Дослідили вплив коефіцієнту швидкості навчання η на ефективність навчання. Побачили, що чим менше його значення від 1, тим більша кількість епох необхідна для навчання мережі, проте тим менші значення похибки для тестової вибірки можна отримати.

Визначили вплив параметру k в функції активації на ефективність навчання. Даний параметр відповідає за крутизну сигмоїдальної функції в області «0». Чим він менший, тим більше епох необхідно для навчання (рис. 11, рис. 12). Великі ж значення параметру k роблять функцію активації більш чутливою і в негативних сценаріях вона може реагувати на випадкові викиди чи шуми. На рис. 14 при $k = 1.5$ можна спостерігати скачкоподібні зміни середньої похибки від епохи до епохи.

В заключній частині була визначена необхідна кількість епох навчання ($\text{Epoch} = 155$), при заданих параметрах, та відповідні їй вагові коефіцієнти двошарового персептрону ($w_{10}^{(1)} = 10.82,\ w_{11}^{(1)} = -21.68,\ w_{12}^{(1)} = 22.05,\ w_{20}^{(1)} = -10.90,\ w_{21}^{(1)} = -20.17,\ w_{22}^{(1)} = 20.73,\ w_{10}^{(2)} = 7.33,\ w_{11}^{(2)} = -16.05\ w_{12}^{(2)} = 15.90$).

Додаток А

Лістинг основної програми на мові Python

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from helpers import f, f_p, get_network_response, get_test_error

df = pd.read_csv('random.csv', index_col='id')
X = df[['x1', 'x2']]
y = df['d']
sns.set_style("whitegrid")
sns.FacetGrid(X.join(y), hue='d').map(plt.scatter, 'x1', 'x2').add_legend()
plt.show()
epoch_max = 1000
eta = 1
alfa = 1
b1_1 = 1
b1_2 = 1
b2_1 = 1
w1_10 = 0.1
w1_11 = -0.3
w1_12 = 0.4
w1_20 = -0.7
w1_21 = -0.1
w1_22 = 0.01
w2_10 = 0.4
w2_11 = -0.2
w2_12 = 0.1

afile = open("lab3_results.csv", "w")
afile.write("n_epoch,error_train,error_test\n")

for n_epoch in range(epoch_max):
    n = 0
    error = 0
    for index, row in df.iterrows():
        x1 = row['x1']
        x2 = row['x2']
        d = row['d']
        u1_1 = w1_10 * b1_1 + w1_11 * x1 + w1_12 * x2
        u1_2 = w1_20 * b2_1 + w1_21 * x1 + w1_22 * x2
        y1_1 = f(u1_1)
        y1_2 = f(u1_2)
        u2_1 = w2_10 * b2_1 + w2_11 * y1_1 + w2_12 * y1_2
        y = f(u2_1)
        e = d - y
        error = error + e * e / 2
```

```
        n = n + 1
        f_poi = f_p(u2_1)
        w2_10 = alfa*w2_10-eta*f_poi*(-e)
        w2_11 = alfa*w2_11-eta*f_poi*y1_1*(-e)
        w2_12 = alfa*w2_12-eta*f_poi*y1_2*(-e)
        f_poi1 = f_p(u1_1)
        w1_10 = alfa*w1_10-eta*f_poi*w2_11*f_poi1*(-e)
        w1_11 = alfa*w1_11-eta*f_poi*w2_11*f_poi1*x1*(-e)
        w1_12 = alfa*w1_12-eta*f_poi*w2_11*f_poi1*x2*(-e)
        f_poi1 = f_p(u1_2)
        w1_20 = alfa*w1_20-eta*f_poi*w2_12*f_poi1*(-e)
        w1_21 = alfa*w1_21-eta*f_poi*w2_12*f_poi1*x1*(-e)
        w1_22 = alfa*w1_22-eta*f_poi*w2_12*f_poi1*x2*(-e)

    train_error = error/n
    test_error = get_test_error(w1_10, w1_11, w1_12, w1_20, w1_21, w1_22,
w2_10, w2_11, w2_12, b1_1, b2_1)
    afile.write(f'{n_epoch},{train_error},{test_error}\n')

afile.close()
```

Додаток Б

Лістинг допоміжних методів на мові Python

```
import math

k = 1

def f(x):
    ee = math.exp(-k * x)
    return 1/(1+ee)

def f_p(x):
    ff = f(x)
    return ff * k * (1 - ff)

def get_network_response(w1_10, w1_11, w1_12, w1_20, w1_21, w1_22, w2_10,
w2_11, w2_12, b1_1, b2_1, x1, x2):
    u1_1 = w1_10*b1_1+w1_11*x1+w1_12*x2
    u1_2 = w1_20*b2_1+w1_21*x1+w1_22*x2
    y1_1 = f(u1_1)
    y1_2 = f(u1_2)
    u2_1 = w2_10*b2_1+w2_11*y1_1+w2_12*y1_2
    return f(u2_1)

def get_test_error(w1_10, w1_11, w1_12, w1_20, w1_21, w1_22, w2_10, w2_11,
w2_12, b1_1, b2_1):
    error = 0
    y1 = get_network_response(w1_10, w1_11, w1_12, w1_20, w1_21, w1_22, w2_10,
w2_11, w2_12, b1_1, b2_1, 0, 0)
    e = 0-y1
    error = error+e*e/2
    y2 = get_network_response(w1_10, w1_11, w1_12, w1_20, w1_21, w1_22, w2_10,
w2_11, w2_12, b1_1, b2_1, 1, 1)
    e = 0-y2
    error = error+e*e/2
    y3 = get_network_response(w1_10, w1_11, w1_12, w1_20, w1_21, w1_22, w2_10,
w2_11, w2_12, b1_1, b2_1, 0, 1)
    e = 1-y3
    error = error+e*e/2
    y4 = get_network_response(w1_10, w1_11, w1_12, w1_20, w1_21, w1_22, w2_10,
w2_11, w2_12, b1_1, b2_1, 1, 0)
    e = 1-y4
    error = error+e*e/2
    return error/4
```