Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №4**

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «MLP. Задача исключающего или»

**Выполнил:**

Студент 3 курса

Группы ИИ-21

Карагодин Д. Л.

**Проверил:**

Туз И. С.

Брест 2023

**Цель:** Изучить обучение и функционирование ИНС типа MLP при решении задач исключающего или.

**Ход работы**

Написать нейронную сеть (multilayer perceptron с одним скрытым слоем) решающую задачу исключающее или. Датасет сгенерировать самостоятельно. Использовать Сигмоидные функции активации

**Код программы:**

import numpy as np

# Инициализация весов сети

def initialize\_weights(input\_dim, hidden\_dim, output\_dim):

    np.random.seed(0)

    W1 = np.random.randn(input\_dim, hidden\_dim)    # Веса первого слоя

    b1 = np.zeros((1, hidden\_dim))                  # Смещение первого слоя

    W2 = np.random.randn(hidden\_dim, output\_dim)    # Веса второго слоя

    b2 = np.zeros((1, output\_dim))                  # Смещение второго слоя

    return W1, b1, W2, b2

# Сигмоидная функция активации

def sigmoid(x):

    return 1 / (1 + np.exp(-x))

# Производная сигмоидной функции

def sigmoid\_derivative(x):

    return x \* (1 - x)

# Функция обучения нейронной сети с одним скрытым слоем

def train(X, y, hidden\_dim, epochs, learning\_rate):

    input\_dim = X.shape[1]

    output\_dim = y.shape[1]

    # Инициализация весов

    W1, b1, W2, b2 = initialize\_weights(input\_dim, hidden\_dim, output\_dim)

    for epoch in range(epochs):

        # Прямое распространение

        hidden\_layer = sigmoid(np.dot(X, W1) + b1)    # Выход скрытого слоя

        output\_layer = sigmoid(np.dot(hidden\_layer, W2) + b2)    # Выход сети

        # Ошибка

        error = y - output\_layer

        # Обратное распространение

        delta\_output = error \* sigmoid\_derivative(output\_layer)

        delta\_hidden = np.dot(delta\_output, W2.T) \* sigmoid\_derivative(hidden\_layer)

        # Обновление весов

        W2 += learning\_rate \* np.dot(hidden\_layer.T, delta\_output)

        b2 += learning\_rate \* np.sum(delta\_output, axis=0, keepdims=True)

        W1 += learning\_rate \* np.dot(X.T, delta\_hidden)

        b1 += learning\_rate \* np.sum(delta\_hidden, axis=0, keepdims=True)

    return W1, b1, W2, b2

# Тестирование обученной нейронной сети

def test(X, W1, b1, W2, b2):

    hidden\_layer = sigmoid(np.dot(X, W1) + b1)

    output\_layer = sigmoid(np.dot(hidden\_layer, W2) + b2)

    return np.round(output\_layer)

# Создание искусственного датасета XOR

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

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

# Параметры обучения

hidden\_dim = 4

epochs = 2500

learning\_rate = 0.1

# Обучение нейронной сети

W1, b1, W2, b2 = train(X, y, hidden\_dim, epochs, learning\_rate)

# Тестирование нейронной сети на новых примерах

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

predictions = test(X\_test, W1, b1, W2, b2)

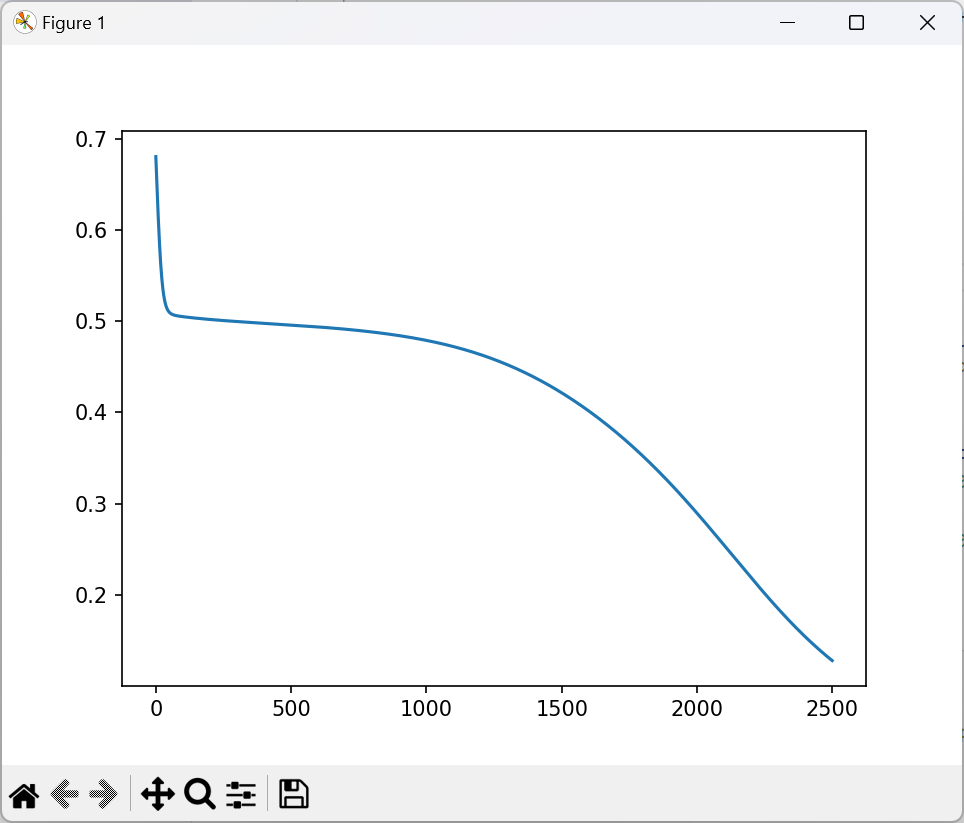
# Вывод результатов

print("Predictions:")

for i in range(len(X\_test)):

    print(f"{X\_test[i]} -> {predictions[i]}")

**Результат программы №1 (График ошибки MSE и Тестирование):**



Predictions:

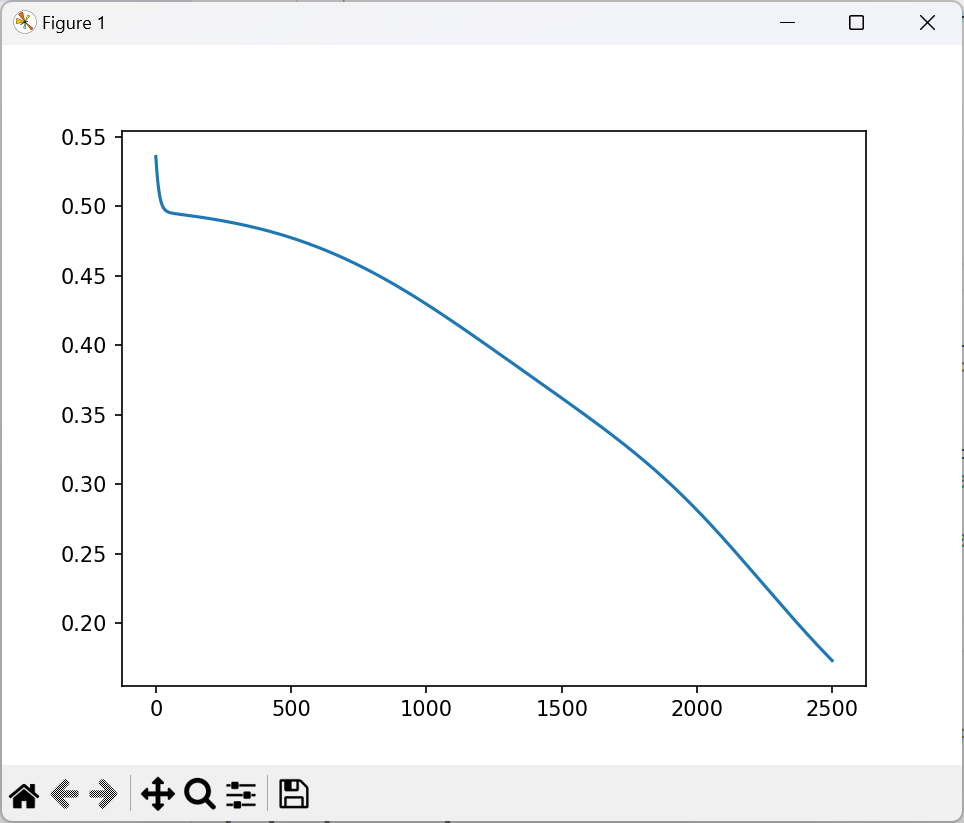
[0 0] -> [0.26743535]

[0 1] -> [0.71069239]

[1 0] -> [0.7782721]

[1 1] -> [0.22538065]

**Результат программы №2 (График ошибки MSE и Тестирование):**



Predictions:

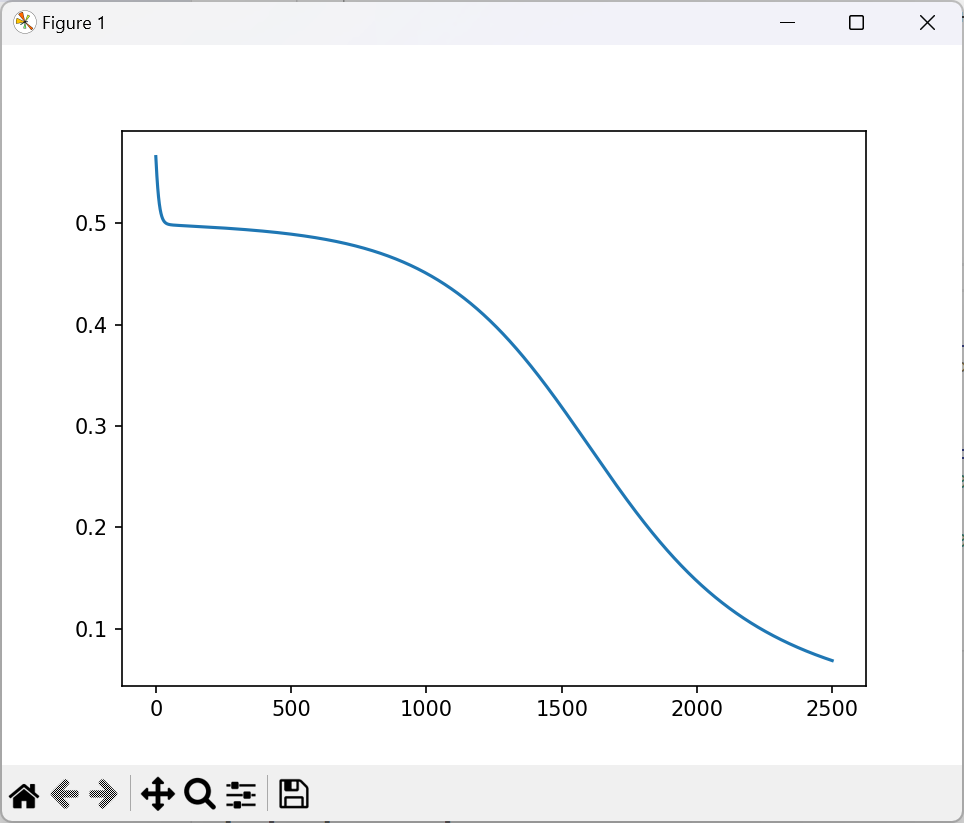
[0 0] -> [0.18315569]

[0 1] -> [0.6801148]

[1 0] -> [0.73291672]

[1 1] -> [0.37233238]

**Результат программы №3 (График ошибки MSE и Тестирование):**



Predictions:

[0 0] -> [0.12668748]

[0 1] -> [0.82716427]

[1 0] -> [0.79423046]

[1 1] -> [0.21975745]

**Вывод:** изучил обучение и функционирование ИНС типа MLP при решении задач исключающего или.