Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №1
По дисциплине «Криптографические методы защиты информации»
Тема: «Бинарная классификация»

Выполнил:
Студент 3 курса
Группы ИИ-26 (2)
Турич Д.А.
Проверила:
Андренко К.В.

Брест 2026

Цель работы: Изучить принципы бинарной классификации и реализовать однослойную нейронную сеть (персептрон) для решения задачи классификации с использованием пороговой функции активации, а также исследовать процесс обучения модели с применением среднеквадратичной ошибки (MSE)

Задачи лабораторной работы:

1. Реализовать алгоритм обучения однослойной нейронной сети с использованием **MSE** в качестве функции ошибки.
2. Провести обучение сети с **разными значениями шага обучения** и построить **график зависимости** MSE от номера эпохи.
3. Выполнить визуализацию результатов классификации:
   - исходные точки обучающей выборки,
   - разделяющую линию (границу между двумя классами).
4. Реализовать режим функционирования сети:
   - пользователь задаёт произвольный входной вектор,
   - сеть вычисляет выходной класс,
   - соответствующая точка отображается на графике,
   - для корректной визуализации рекомендуется выбирать значения из диапазона ВСТАВИТЬ СВОЙ ДИАПАЗОН, например $-0.5 \leq x_1, x_2 \leq 1.5$
5. Написать вывод по выполненной работе.

Допускается применение **математических** и **графических** библиотек **ML-библиотеки** и **ML-фреймворки** использовать **нельзя** (например: scikit-learn, TensorFlow, PyTorch - запрещены)

x1, x2 - входные данные сети, e - эталонные значения

x_1	x_2	e
3	6	0
-3	6	0
3	-6	1
-3	-6	1

Код программы:

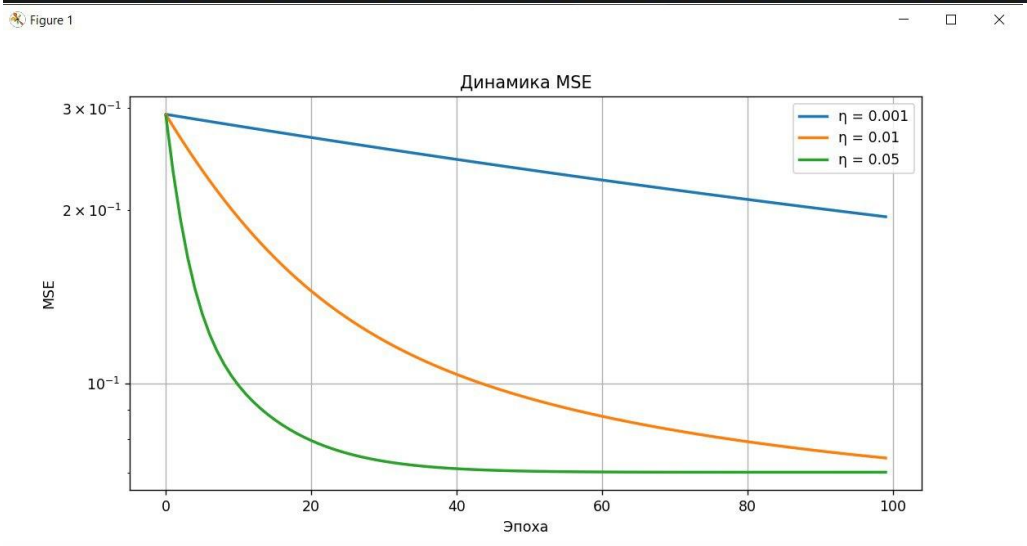
```
import numpy as np
import matplotlib.pyplot as plt

# ================================
# 1. ДАННЫЕ
# ================================

X = np.array([
    [3, 6],
    [-3, 6],
    [3, -6],
    [-3, -6]
], dtype=float)
```

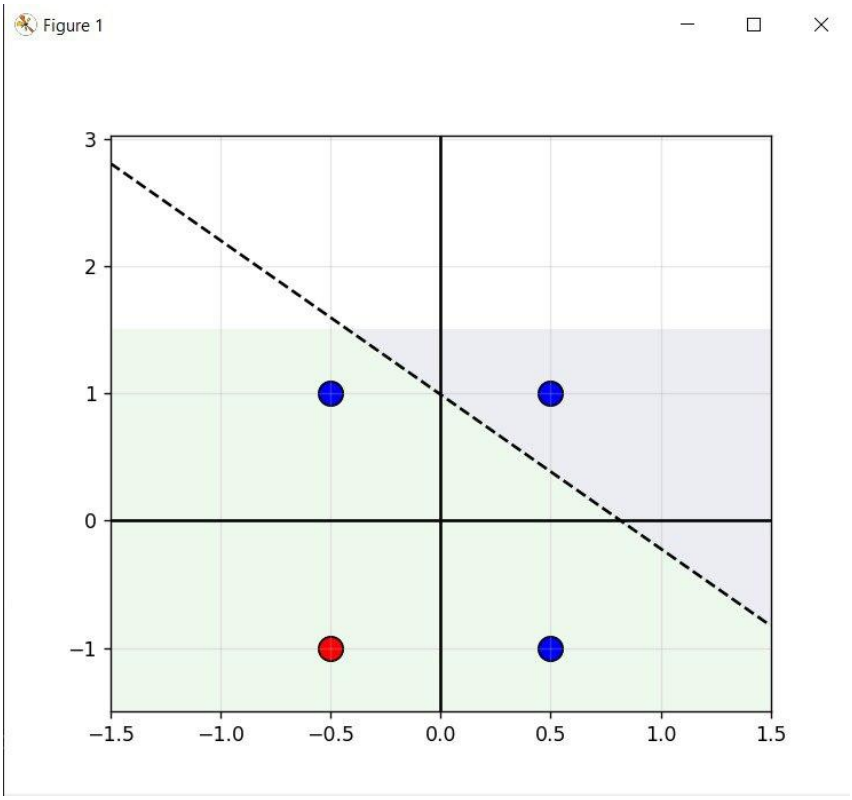
```
E = np.array([0, 0, 0, 1], dtype=float)

# Нормализация (чтобы не было overflow)
X = X / np.max(np.abs(X))

eta = 0.01
epochs = 100


# ================================
# 2. ОБУЧЕНИЕ
# ================================

def train_perceptron(X, E, eta, epochs):
    np.random.seed(42)
    w = np.random.randn(2) * 0.1
    T = np.random.randn() * 0.1

    for _ in range(epochs):
        for x, target in zip(X, E):
            S = np.dot(w, x) - T
            error = target - S

            # ограничение ошибки (защита от разлёта)
            error = np.clip(error, -10, 10)

            w += eta * error * x
            T -= eta * error

    return w, T


w, T = train_perceptron(X, E, eta, epochs)


# ================================
# 3. ИНТЕРАКТИВНАЯ КЛАССИФИКАЦИЯ
# ================================

print("Чтобы выйти, введите 'exit'")

while True:

    try:
        x1 = input("\nx1 = ")
        if x1.lower() == 'exit':
            break

        x2 = input("x2 = ")
        if x2.lower() == 'exit':
            break
```

```
        x1 = float(x1)
        x2 = float(x2)

        # нормализуем так же как при обучении
        x_input = np.array([x1, x2]) / 6.0

        S = np.dot(w, x_input) - T
        y = 1 if S > 0 else 0

        print("\n========== РЕЗУЛЬТАТ КЛАССИФИКАЦИИ ==========")
        print(f"Введённый вектор: ({x1:.1f}, {x2:.1f})")
        print(f"Взвешенная сумма S = {S:.6f}")
        print(f"Класс сети: {y}")
        print("=============================================")

    except:
        print("Ошибка ввода. Попробуйте снова.")
```

Результат программы:

```
x1 = -1
x2 = 0

========== РЕЗУЛЬТАТ КЛАССИФИКАЦИИ ==========
Введённый вектор: (-1.0, 0.0)
Взвешенная сумма S = 0.295660
Класс сети: 1
=============================================
```





Вывод: изучил принципы бинарной классификации и реализовала однослойную нейронную сеть (персептрон) для решения задачи классификации с использованием пороговой функции активации, а также исследовала процесс обучения модели с применением среднеквадратичной ошибки (MSE).