

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

Лабораторная работа №1

По дисциплине: «Модели решения задач в интеллектуальных системах»
Тема: «Бинарная классификация»

Выполнил:
Студент 3 курса
Группы ИИ-26
Хомиченко И.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 (**дельта-правило**).
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)

Вариант 8

x_1, x_2 - входные данные сети, e - эталонные значения

x_1	x_2	e
5	6	0
-5	6	1
5	-6	1
-5	-6	1

Код программы:

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

data_points = np.array([
    [5, 6],
    [-5, 6],
    [5, -6],
    [-5, -6]
])

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

alpha = 0.01
```

```

iterations = 200
np.random.seed(7)

weights = np.random.uniform(-0.5, 0.5, 2)
bias = np.random.uniform(-0.5, 0.5)

loss_values = []

for step in range(iterations):
    step_errors = []
    for idx in range(len(data_points)):
        net = np.dot(data_points[idx], weights) + bias
        error = targets[idx] - net

        weights += alpha * error * data_points[idx]
        bias += alpha * error

    step_errors.append(error ** 2)

    loss_values.append(np.mean(step_errors))

plt.figure(figsize=(8, 4))
plt.plot(loss_values, color='purple', linewidth=2)
plt.title("MSE")
plt.xlabel("Epoch")
plt.ylabel("Error")
plt.grid(True)
plt.show()

plt.figure(figsize=(8, 6))

plt.scatter(data_points[targets == 0][:, 0],
            data_points[targets == 0][:, 1],
            color='orange', s=120, label='Class 0')

plt.scatter(data_points[targets == 1][:, 0],
            data_points[targets == 1][:, 1],
            color='cyan', s=120, label='Class 1')

axis_x = np.linspace(-10, 10, 100)
axis_y = (0.5 - bias - weights[0] * axis_x) / weights[1]

plt.plot(axis_x, axis_y, color='black', linestyle='-.', linewidth=2)

plt.xlim(-10, 10)
plt.ylim(-10, 10)
plt.title("Decision Boundary")
plt.legend()
plt.grid(True)
plt.show()

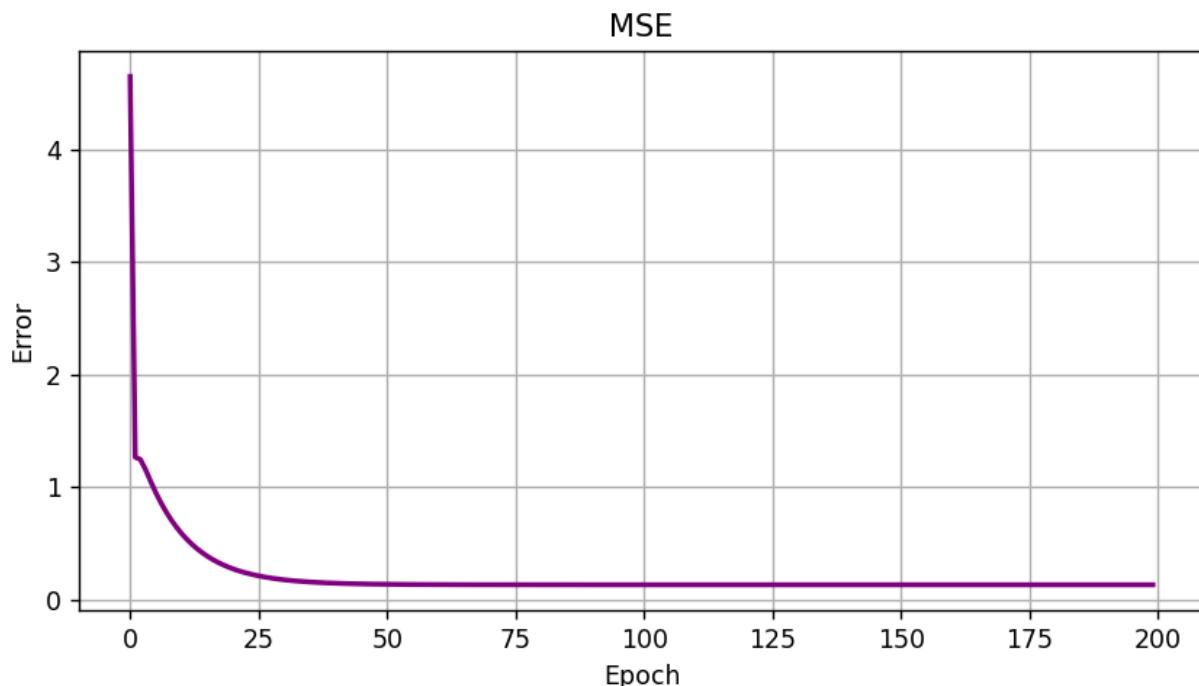
print("\nTest mode:")
while True:
    user_value = input("Enter x1, x2 or 'q': ")
    if user_value.lower() == 'q':
        break
    try:
        point = np.array([float(v.strip()) for v in user_value.split(',')])
        net_test = np.dot(point, weights) + bias
        result = 1 if net_test >= 0.5 else 0
        print(f"Class {result} (S = {net_test:.4f})")
    except:
        print("Input error")

```

Результат программы:

Figure 1

- □ ×

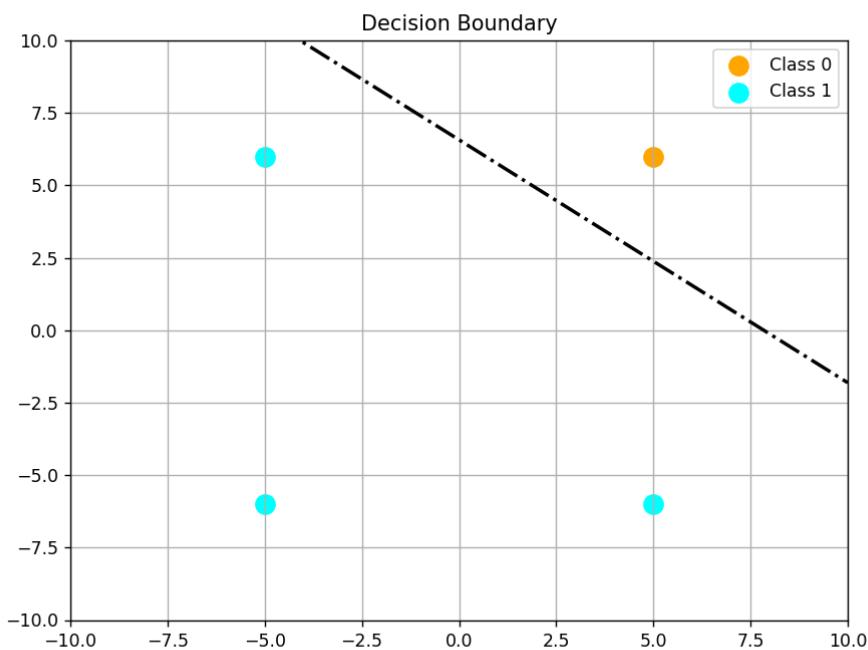


Home ← → | + Q E |

(x, y) = (0.2, 4.86)

Figure 1

- □ ×



Home ← → | + Q E |

```
Test mode:  
Enter x1, x2 or 'q': 5,6  
Class 0 (S = 0.3623)  
Enter x1, x2 or 'q': -5,6  
Class 1 (S = 0.6811)  
Enter x1, x2 or 'q': 5,-6  
Class 1 (S = 0.8188)  
Enter x1, x2 or 'q': -5,-6  
Class 1 (S = 1.1377)  
Enter x1, x2 or 'q':
```

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