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

Кушнеревич Е.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).

Ход 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 $-0.5 \leq x_1, x_2 \leq 1.5$

8.

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

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

Код программы:

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


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

etalon_value = np.array([0, 1, 1, 1]).reshape(-1, 1)
```

```python
def linear_output(weights, x, T):
    return x @ weights - T


def online_fit(X, y, alpha=0.001, epochs=2500, eps=1e-7, mix=True):
    N = X.shape[0]
    W = np.zeros((X.shape[1], 1))
    T = 0
    mse = []
    prev_loss = np.inf

    print(f"Старт обучения (alpha = {alpha})")

    for ep in range(epochs):
        order = np.random.permutation(N) if mix else np.arange(N)

        for idx in order:
            x_i = X[idx:idx+1]
            t_i = y[idx]

            s = linear_output(W, x_i, T)
            delta = s - t_i

            W -= alpha * x_i.T @ delta
            T += alpha * delta.item()

        preds = linear_output(W, X, T)
        loss = np.mean((preds - y) ** 2)
        mse.append(loss)

        if ep % 100 == 0 and ep > 0:
            print(f"  эпоха {ep:4d} | MSE = {loss:.8f}")

        if ep > 30 and abs(loss - prev_loss) < eps:
            print(f"  остановка на эпохе {ep}")
            break

        prev_loss = loss

    print(f" Обучение завершено | итоговая MSE = {loss:.8f}\n")
    return W, T, mse


learning_rates = [0.0003, 0.0007, 0.0012, 0.002]
models = {}

for alpha in learning_rates:
    W, T, err = online_fit(x, etalon_value, alpha=alpha)
    models[alpha] = (W, T, err)
```

```
plt.figure(figsize=(10, 5))
for lr, (_, _, err) in models.items():
    plt.plot(err, label=f"lr={lr}", lw=1.5)

plt.yscale("log")
plt.xlabel("Эпохи")
plt.ylabel("MSE")
plt.grid(alpha=0.35)
plt.legend()
plt.tight_layout()
plt.show()


best_alpha = min(models, key=lambda k: models[k][2][-1])
W, T = models[best_alpha][0], models[best_alpha][1]
print(f" Выбрана модель с alpha = {best_alpha}\n")


added_pts = []
added_cls = []

print("Введите координаты точки (x_1 x_2), либо exit\n")

while True:
    user_input = input(" -> ").strip()
    if user_input in ("exit", ""):
        break

    try:
        x1_val, x2_val = map(float, user_input.split())
        point = np.array([[x1_val, x2_val]])

        score = linear_output(W, point, T)[0, 0]
        cls = 1 if score >= 0.5 else 0
        color_name = "красный" if cls == 1 else "синий"

        print(f"  Значение S = {score:>9.6f}")
        print(f"  Класс     = {cls}")

        added_pts.append([x1_val, x2_val])
        added_cls.append(cls)

        fig, ax = plt.subplots(figsize=(8, 8))

        ax.scatter(
            x[etalon_value[:,0]==0][:,0],
            x[etalon_value[:,0]==0][:,1],
            s=150, color="royalblue", edgecolors="black", label="класс 0"
        )

        ax.scatter(
```

```
            x[etalon_value[:,0]==1][:,0],
            x[etalon_value[:,0]==1][:,1],
            s=150, color="crimson", edgecolors="black", label="класс 1"
        )

        w1, w2 = W.flatten()
        xs = np.linspace(-9, 9, 400)
        if abs(w2) > 1e-8:
            ys = (T + 0.5 - w1 * xs) / w2
            ax.plot(xs, ys, "green", lw=2.3, label="граница")

        for p, c in zip(added_pts, added_cls):
            ax.scatter(p[0], p[1],
                       marker="X",
                       s=220,
                       color="crimson" if c else "royalblue",
                       edgecolors="black")

        ax.set_xlim(-9, 9)
        ax.set_ylim(-9, 9)
        ax.set_xlabel("x")
        ax.set_ylabel("y")
        ax.grid(alpha=0.3)
        ax.legend()
        plt.tight_layout()
        plt.show()

    except Exception:
        print("Ошибка ввода. Используйте формат: x y\n")

print("Работа программы завершена.")
```

Результат тестирования:

Старт обучения (alpha = 0.0003)

эпоха 100 | MSE = 0.50306227

эпоха 200 | MSE = 0.40819011

эпоха 300 | MSE = 0.33382881

эпоха 400 | MSE = 0.27546168

эпоха 500 | MSE = 0.22964722

эпоха 600 | MSE = 0.19368815

эпоха 700 | MSE = 0.16546716

эпоха 800 | MSE = 0.14331707

эпоха 900 | MSE = 0.12593228

эпоха 1000 | MSE = 0.11228579

эпоха 1100 | MSE = 0.10157649

эпоха 1200 | MSE = 0.09316917

эпоха 1300 | MSE = 0.08657202

эпоха 1400 | MSE = 0.08139280

эпоха 1500 | MSE = 0.07732885

эпоха 1600 | MSE = 0.07413909

эпоха 1700 | MSE = 0.07163408

эпоха 1800 | MSE = 0.06966953

эпоха 1900 | MSE = 0.06812725

эпоха 2000 | MSE = 0.06691682

эпоха 2100 | MSE = 0.06596636

эпоха 2200 | MSE = 0.06522055

эпоха 2300 | MSE = 0.06463506

эпоха 2400 | MSE = 0.06417556

Обучение завершено | итоговая MSE = 0.06381828

Старт обучения (alpha = 0.0007)

эпоха 100 | MSE = 0.37811715

эпоха 200 | MSE = 0.24059735

эпоха 300 | MSE = 0.16300168

эпоха 400 | MSE = 0.11920752

эпоха 500 | MSE = 0.09450595

эпоха 600 | MSE = 0.08055743

эпоха 700 | MSE = 0.07269560

эпоха 800 | MSE = 0.06825636

эпоха 900 | MSE = 0.06574683

эпоха 1000 | MSE = 0.06433031

эпоха 1100 | MSE = 0.06353627

эпоха 1200 | MSE = 0.06308320

эпоха 1300 | MSE = 0.06282890

остановка на эпохе 1334

Обучение завершено | итоговая MSE = 0.06277507

Старт обучения (alpha = 0.0012)

эпоха 100 | MSE = 0.26826843

эпоха 200 | MSE = 0.13856790

эпоха 300 | MSE = 0.09057652

эпоха 400 | MSE = 0.07287843

эпоха 500 | MSE = 0.06634076

эпоха 600 | MSE = 0.06392189

эпоха 700 | MSE = 0.06302579

эпоха 800 | MSE = 0.06271926

эпоха 900 | MSE = 0.06259220

остановка на эпохе 951

Обучение завершено | итоговая MSE = 0.06256073

Старт обучения (alpha = 0.002)

эпоха 100 | MSE = 0.16362840

эпоха 200 | MSE = 0.08100092

эпоха 300 | MSE = 0.06588886

эпоха 400 | MSE = 0.06315687

эпоха 500 | MSE = 0.06261771

остановка на эпохе 512

Выбрана модель с alpha = 0.0012

Введите координаты точки (x_1 x_2), либо exit

-> 5 6

Значение S = 0.244878

Класс = 0

Обучение завершено | итоговая MSE = 0.06260255

График среднеквадратичной ошибки:

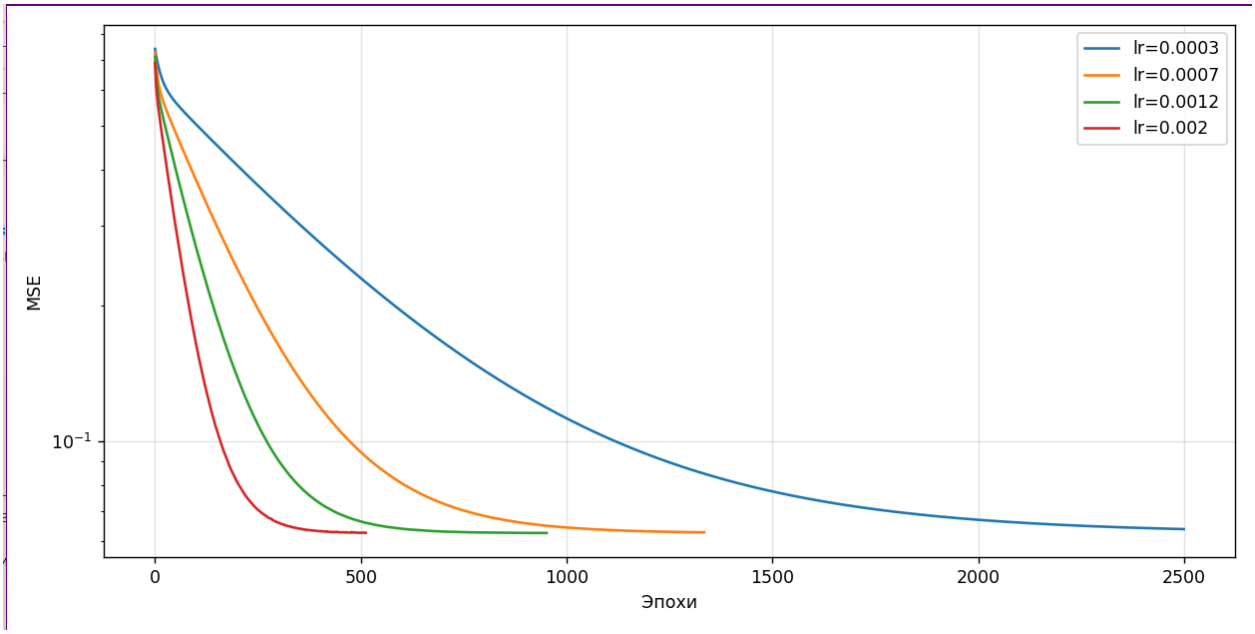
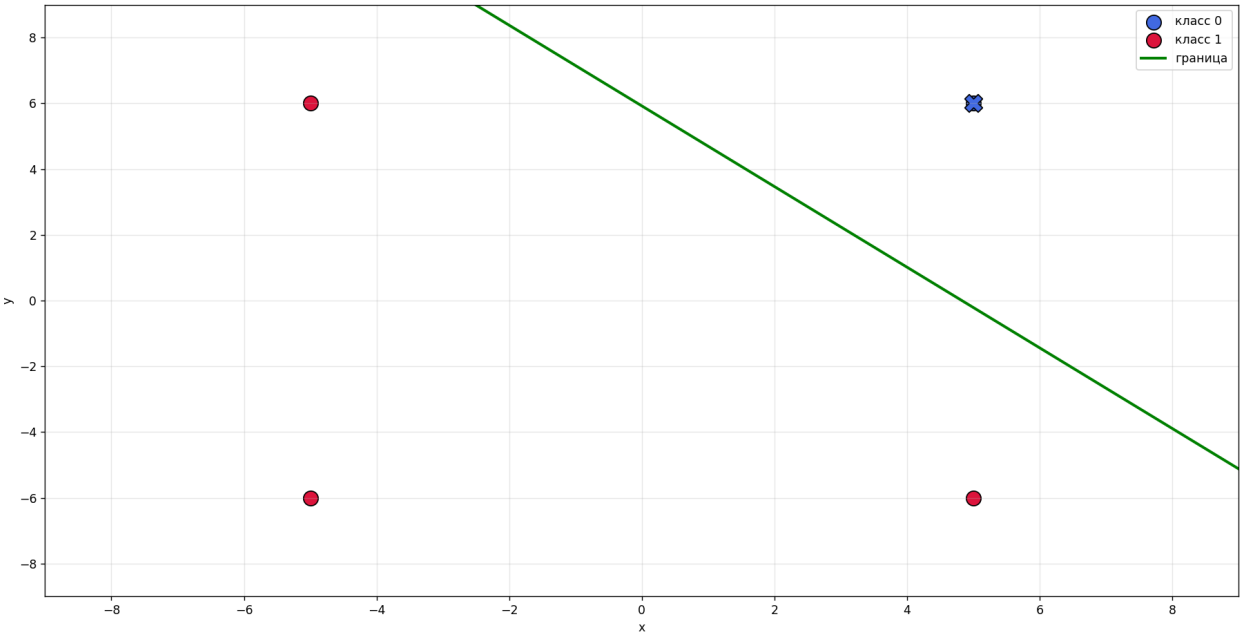


График с разделяющей поверхностью:



Вывод: Проделав данную лабораторную работу я 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).