МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Выполнил
Е.А. Ерошин
студент группы ИИ-26

Проверил
К.В. Андренко
ст. преп. кафедры ИИТ,
«___»_____2026 г.

Брест 2026

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

Задание 1. Для заданного массива вещественных чисел найти среднее значение и стандартное отклонение.

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 $-0.5 \leq x1, x2 \leq 1.5$

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 3

x_1	x_2	e
3	4	1
-3	4	1
3	-4	0
-3	-4	0

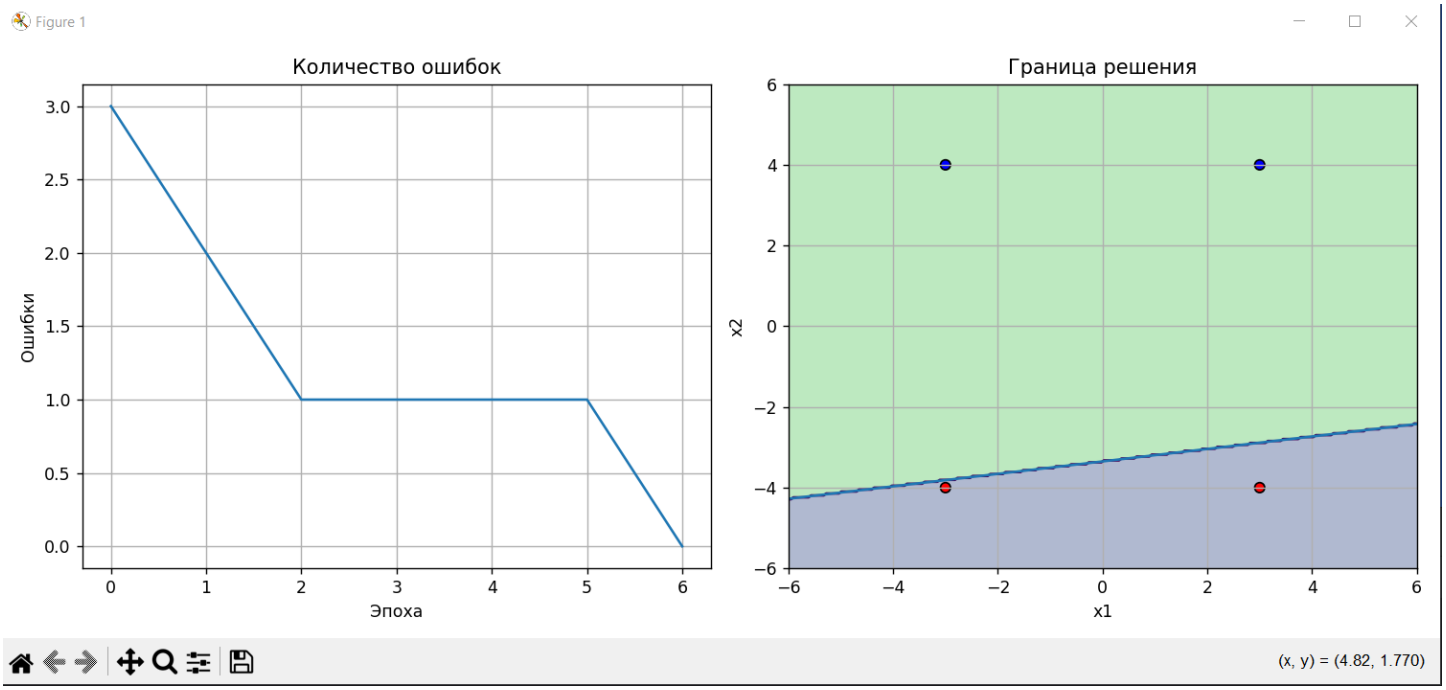
Код программы:

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


class LinearBinaryClassifier:
    def __init__(self, features_count, lr=0.01):
        self.lr = lr
        self.w = np.random.uniform(-0.5, 0.5, features_count + 1)
```

```
    def _add_bias_column(self, X):
        X = np.array(X, dtype=float)
        if X.ndim == 1:
            X = X.reshape(1, -1)
        bias = np.ones((X.shape[0], 1))
        return np.concatenate((bias, X), axis=1)

    def _linear_output(self, Xb):
        return Xb @ self.w

    def _activation(self, z):
        return np.where(z >= 0, 1, 0)

    def predict(self, X):
        Xb = self._add_bias_column(X)
        return self._activation(self._linear_output(Xb))

    def train(self, X, y, epochs=200):
        Xb = self._add_bias_column(X)
        y = np.array(y, dtype=float)
        history = []

        for epoch in range(epochs):
            errors = 0

            for xi, target in zip(Xb, y):
                output = np.dot(xi, self.w)          # линейный выход
                delta = target - output              # ошибка (дельта)
                self.w += self.lr * delta * xi       # дельта-правило

                predicted_class = 1 if output >= 0 else 0
                if predicted_class != target:
                    errors += 1

            history.append(errors)

            if errors == 0:
                break

        return history


# Данные
X_data = np.array([
    [3, 4],
    [-3, 4],
    [3, -4],
    [-3, -4]
])

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


# Обучение
classifier = LinearBinaryClassifier(features_count=2, lr=0.01)
loss_curve = classifier.train(X_data, y_data)


# Визуализация
fig, axes = plt.subplots(1, 2, figsize=(12, 5))
```

```
axes[0].plot(loss_curve)
axes[0].set_title("Количество ошибок")
axes[0].set_xlabel("Эпоха")
axes[0].set_ylabel("Ошибки")
axes[0].grid()

xmin, xmax = -6, 6
ymin, ymax = -6, 6

grid_x, grid_y = np.meshgrid(
    np.linspace(xmin, xmax, 250),
    np.linspace(ymin, ymax, 250)
)

grid_points = np.c_[grid_x.ravel(), grid_y.ravel()]
grid_pred = classifier.predict(grid_points).reshape(grid_x.shape)

axes[1].contourf(grid_x, grid_y, grid_pred, levels=[-0.1, 0.5, 1.1], alpha=0.4)
axes[1].contour(grid_x, grid_y, grid_pred, levels=[0.5])

colors = ['red' if label == 0 else 'blue' for label in y_data]
axes[1].scatter(X_data[:, 0], X_data[:, 1], c=colors, edgecolors='black')

if abs(classifier.w[2]) > 1e-6:
    x_line = np.linspace(xmin, xmax, 200)
    y_line = -(classifier.w[0] + classifier.w[1]*x_line) / classifier.w[2]
    axes[1].plot(x_line, y_line)

axes[1].set_xlim(xmin, xmax)
axes[1].set_ylim(ymin, ymax)
axes[1].set_xlabel("x1")
axes[1].set_ylabel("x2")
axes[1].set_title("Граница решения")
axes[1].grid()

plt.tight_layout()
plt.show()


print("\nВведите координаты точки (x1 x2) или 'q' для выхода\n")

while True:
    user_input = input(">> ").strip()

    if user_input.lower() in ('q', 'exit'):
        break

    try:
        x1, x2 = map(float, user_input.split())
        prediction = classifier.predict([x1, x2])[0]
        print(f"Класс точки: {prediction}")
    except:
        print("Ошибка ввода. Введите два числа через пробел.")
```

Результат работы программы



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