МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТк

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Специальность ИИ26(з)

Выполнил
Т.В.
Данилюк,
студент группы ИИ26

Проверила
К.В. Андренко,
ст. преп. кафедры ИИТ,
«___k_____2026 г.

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).

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≤x1,x2≤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)

Код программы:

Содержимое в файле main.py

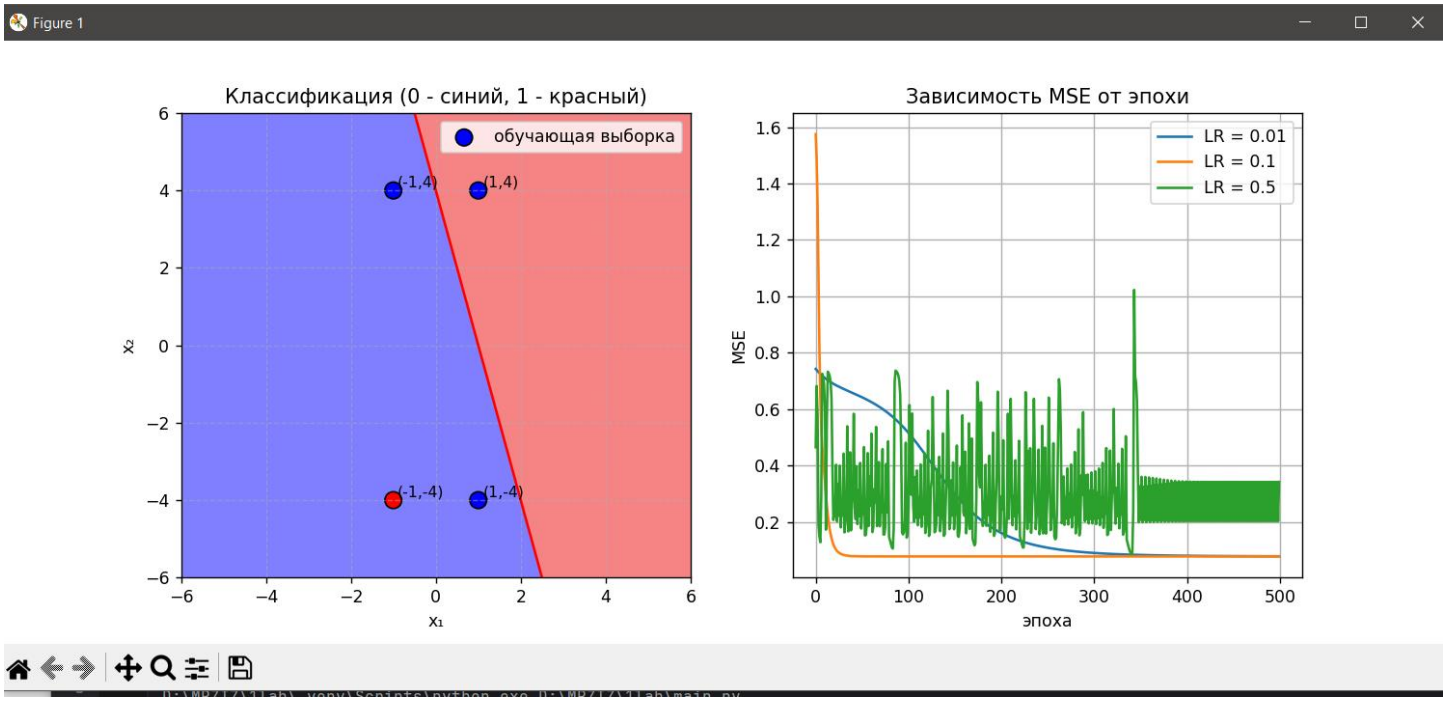
```
import numpy as np
import matplotlib.pyplot as plt


class Perceptron:
    def __init__(self, input_size=0, learning_rate=0.1):
        self.X = np.array([])
        self.w = np.random.uniform(-1.0, 1.0, input_size + 1)  # threshold w[0]
        self.learning_rate = learning_rate
        self.target = np.array([])

    def set_X(self, X: np.array) -> None:
        X = np.array(X)

        if X.ndim == 1:
            X = X.reshape(1, -1)
        elif X.ndim != 2:
            print("X must be 1D or 2D array")
            return

        self.X = X
        self.X = np.insert(self.X, 0, -1, axis=1)

    def set_w(self, w: np.array) -> None:
```

```
        self.w = w  # the first one is a threshold

    def set_target(self, target: np.array) -> None:
        if self.X.ndim != 2:
            print("X not setted!")
            return

        if len(target) != len(self.X):
            print(f"Invalid size of target vector. It must have length = {len(self.X)}. Now = {len(target)}")
            return

        self.target = target

    def activate(self, arr_wsum: np.array) -> np.array:
        return 2 / (1 + np.exp(-arr_wsum)) - 1

    def der_activate(self, y_pred: np.array) -> np.array:
        return 0.5 * (1 - y_pred ** 2)

    def forward(self, X_input=None) -> np.array:
        X = self.X if X_input is None else X_input

        if X.size == 0:
            print("Input X vector not set!")
            return None

        wsum = np.dot(X, self.w)
        y = self.activate(wsum)

        return y

    def train(self, epochs=1000) -> list:
        mse_history = []
        for epoch in range(epochs):
            y_pred = self.forward()

            error = y_pred - self.target  # gamma

            mse = np.mean(error ** 2)
            mse_history.append(mse)

            self.w = self.w - np.dot(self.learning_rate * error * self.der_activate(y_pred), self.X)

        return mse_history


# Данные из вашей таблицы
X_train = np.array([[1, 4], [-1, 4], [1, -4], [-1, -4]])
Y_targets = np.array([0, 0, 0, 1])

plt.figure(figsize=(12, 5))
plt.subplot(1, 2, 2)
for lr in [0.01, 0.1, 0.5]:
    test_p = Perceptron(input_size=2, learning_rate=lr)
    test_p.set_X(X_train)
    test_p.set_target(Y_targets)
    history = test_p.train(epochs=500)
    plt.plot(history, label=f'LR = {lr}')

plt.title("Зависимость MSE от эпохи")
plt.xlabel("эпоха")
plt.ylabel("MSE")
plt.legend()
```

```
plt.grid(True)

p = Perceptron(input_size=2, learning_rate=0.1)
p.set_X(X_train)
p.set_target(Y_targets)
p.train(epochs=1000)


def plot_current_state(user_point=None, user_class=None):
    plt.subplot(1, 2, 1)
    plt.cla()

    xx, yy = np.meshgrid(np.linspace(-6, 6, 200), np.linspace(-6, 6, 200))
    grid_points = np.c_[np.ones(xx.ravel().shape) * -1, xx.ravel(), yy.ravel()]
    Z = p.forward(grid_points).reshape(xx.shape)

    plt.contourf(xx, yy, Z, levels=0, colors=["#ef0a0a", "#0000ff"], alpha=0.5)
    plt.contour(xx, yy, Z, levels=[0], colors='r')

    # Окрашиваем точки в зависимости от класса (0 - синий, 1 - красный)
    colors = ['blue' if y == 0 else 'red' for y in Y_targets]
    plt.scatter(X_train[:, 0], X_train[:, 1], c=colors, s=100, edgecolors='k', label='обучающая выборка')

    # Подписываем точки
    for i, (x, y) in enumerate(X_train):
        plt.text(x + 0.1, y + 0.1, f'({x},{y})', fontsize=9)

    if user_point is not None:
        plt.scatter(user_point[0], user_point[1], color='yellow', marker='*', s=200,
                    edgecolors='black', label=f'Ввод: класс {user_class:.2f}')

    plt.xlim([-6, 6])
    plt.ylim([-6, 6])
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.legend(loc='upper right')
    plt.title("Классификация (0 - синий, 1 - красный)")
    plt.xlabel("x₁")
    plt.ylabel("x₂")
    plt.pause(0.5)


plot_current_state()
plt.show(block=False)

print("Введите координаты точки (x₁, x₂) для классификации")
print("Данные из таблицы:")
print("(1, 4) -> 0")
print("(-1, 4) -> 0")
print("(1, -4) -> 0")
print("(-1, -4) -> 1")

try:
    while True:
        line = input("Введите x₁ x₂ (через пробел или 'exit' для выхода): ")
        if line.lower() == 'exit':
            break

        coords = list(map(float, line.split()))

        if len(coords) != 2:
            print("Введите 2 координаты!")
            continue
```

```
        user_x = np.array([[-1, coords[0], coords[1]]])
        prediction = p.forward(user_x)[0]

        # Преобразуем непрерывное значение в бинарный класс
        binary_class = 1 if prediction > 0 else 0

        print(f"Результат: {prediction:.4f} (Класс {binary_class})")

        plt.subplot(1, 2, 1)
        plot_current_state(coords, binary_class)
        plt.draw()

except ValueError:
    print("Выход...")
except KeyboardInterrupt:
    print("\nВыход...")

plt.show()
```

Рисунки с результатами работы программы



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).