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

Лабораторная работа №1
По дисциплине : «Модели решения задач в ИС»
По теме : «Бинарная классификация»

Выполнил:
студент 3 курса
группы ИИ-26
Заруцкий В.Я.
Проверил:
Адренко 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).

Вариант 4:

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

Ход работы :

Код программы:

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

def load_custom_dataset():
    data = [
        [4, 1, 1],
        [-4, 1, 1],
        [4, -1, 1],
        [-4, -1, 0]
    ]
    X = np.array([[p[0], p[1]] for p in data])
    y = np.array([p[2] for p in data])
    return X, y

def step_function(x):
    return 1 if x >= 0 else 0

class Perceptron:
    def __init__(self, input_size, learning_rate=0.1):
        self.weights = np.random.randn(input_size) * 0.1
        self.bias = np.random.randn() * 0.1
        self.learning_rate = learning_rate
        self.mse_history = []

    def predict(self, x):
        z = np.dot(x, self.weights) + self.bias
        return step_function(z)

    def fit(self, X, y, epochs=20):
        n_samples = X.shape[0]
        print("Старт обучения")
        print(f"Начальные веса: {np.round(self.weights, 3)}, bias={self.bias:.3f}")

        for epoch in range(epochs):
            total_error = 0
            for i in range(n_samples):
                z = np.dot(X[i], self.weights) + self.bias
                prediction = step_function(z)
                error = y[i] - prediction
                total_error += error ** 2
                self.weights += self.learning_rate * error * X[i]
                self.bias += self.learning_rate * error
            mse = total_error / n_samples
            self.mse_history.append(mse)

        print("Обучение завершено")

    def decision_boundary(self, x):
        if abs(self.weights[1]) < 1e-10:
            return np.full_like(x, np.nan)
        return (-self.weights[0] * x - self.bias) / self.weights[1]

def visualize_results(X, y, perceptron, new_point=None):
    plt.figure(figsize=(12, 5))

    plt.subplot(1, 2, 1)
    epochs = range(1, len(perceptron.mse_history) + 1)
    plt.plot(epochs, perceptron.mse_history, 'b-o')
    plt.xlabel("Эпоха")
    plt.ylabel("MSE")
    plt.title("Снижение ошибки по эпохам")
```

```
    plt.grid(True)

    plt.subplot(1, 2, 2)
    colors = ['red' if label == 0 else 'blue' for label in y]
    plt.scatter(X[:, 0], X[:, 1], c=colors, edgecolors='k', s=150)

    x_min, x_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x_vals = np.linspace(x_min, x_max, 100)
    y_vals = perceptron.decision_boundary(x_vals)
    plt.plot(x_vals, y_vals, 'g--', linewidth=2, label="Разделяющая линия")

    if new_point is not None:
        px, py = new_point
        pred = perceptron.predict(new_point)
        color = "blue" if pred == 1 else "red"
        plt.scatter(px, py, c=color, s=200, marker="*", edgecolors='k')
        plt.text(px, py, f"Класс: {pred}", fontsize=10)

    plt.xlabel("x1")
    plt.ylabel("x2")
    plt.title("Классификация и разделяющая линия")
    plt.grid(True)
    plt.legend()
    plt.tight_layout()
    plt.show()

def user_input_point():
    print("\nВведите новую точку для классификации:")
    x1 = float(input("x1 = "))
    x2 = float(input("x2 = "))
    return np.array([x1, x2])

if __name__ == "__main__":
    X, y = load_custom_dataset()
    perceptron = Perceptron(input_size=2, learning_rate=0.1)
    perceptron.fit(X, y, epochs=20)

    new_point = user_input_point()
    visualize_results(X, y, perceptron, new_point=new_point)

    print("\nИтоговые параметры модели")
    print(f"Веса: {np.round(perceptron.weights, 3)}")
    print(f"Смещение: {perceptron.bias:.3f}")
    print("\nMSE (первые 5 эпох):", [round(m, 3) for m in perceptron.mse_history[:5]], "...")

    correct = sum(perceptron.predict(X[i]) == y[i] for i in range(len(X)))
    accuracy = correct / len(X) * 100
    print(f"\nТочность на обучающей выборке: {accuracy:.0f}% ({correct}/{len(X)})")

    print("\nКлассификация новой точки")
    print(f"Точка: x1 = {new_point[0]}, x2 = {new_point[1]}")
    print(f"Предсказанный класс: {perceptron.predict(new_point)}")
```

Результат тестирования:

Старт обучения
Начальные веса: [-0.066 -0.039], bias=0.137
Обучение завершено

Введите новую точку для классификации:
x1 = 2
x2 = 0

Итоговые параметры модели
Веса: [0.334 0.661]
Смещение: 0.837

MSE (первые 5 эпох): [0.75, 0.5, 0.5, 0.5, 0.5] ...

Точность на обучающей выборке: 100% (4/4)

Классификация новой точки
Точка: x1 = 2.0, x2 = 0.0

График с разделяющей поверхностью:

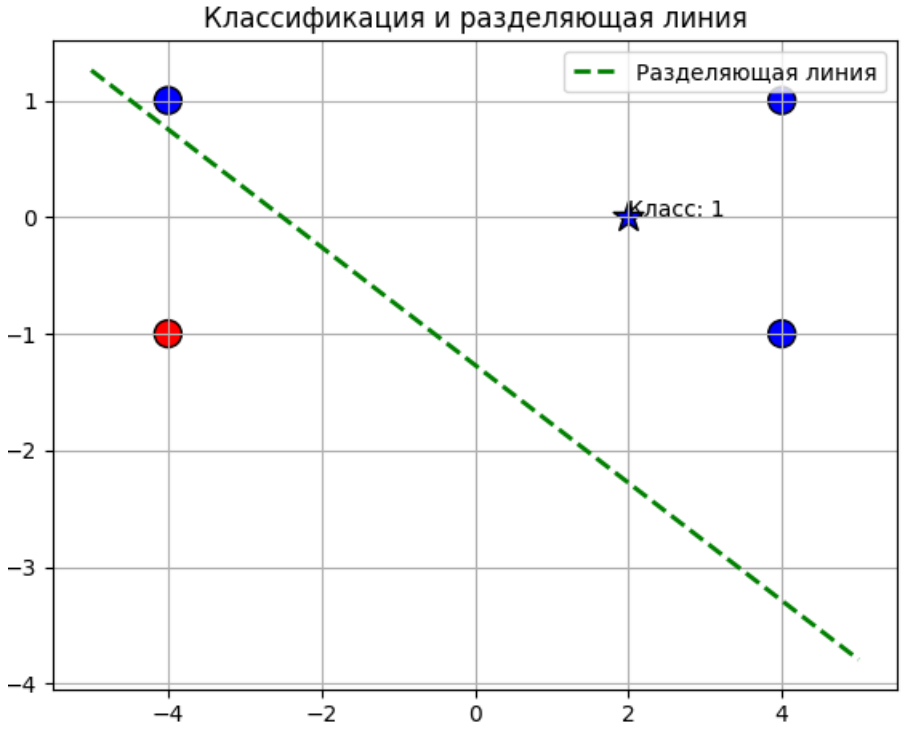
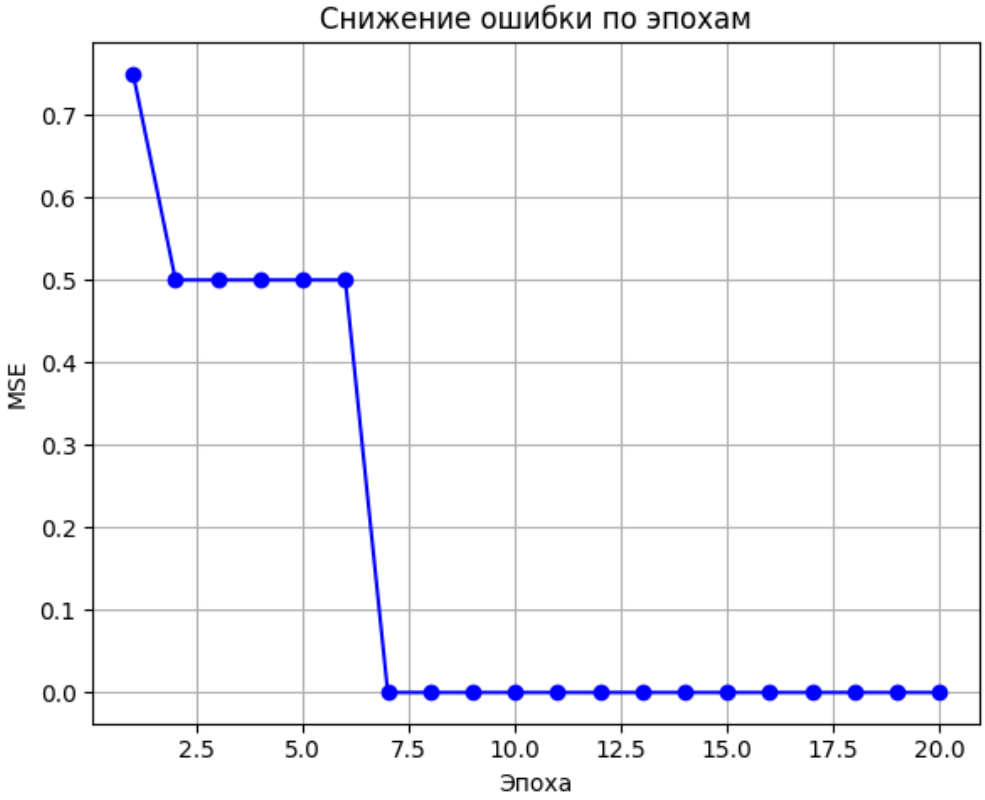


График изменения ошибки:



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).