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], dtype=float)
    y = np.array([p[2] for p in data], dtype=int)
    y = np.where(y == 0, -1, 1)
    return X, y

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

class ADALINE:
    def __init__(self, input_size, learning_rate=0.01):
        self.w = np.zeros(input_size, dtype=float)
        self.b = 0.0
        self.lr = learning_rate
        self.mse_history = []

    def predict_raw(self, x):
        return np.dot(self.w, x) + self.b

    def predict(self, x):
        return step_sign(self.predict_raw(x))

    def fit_adaline(self, X, y, epochs=50, shuffle=True):
        n = X.shape[0]
        for epoch in range(epochs):
            if shuffle:
                idx = np.random.permutation(n)
                X_epoch = X[idx]
                y_epoch = y[idx]
            else:
                X_epoch = X
                y_epoch = y

            mse = 0.0
            for xi, yi in zip(X_epoch, y_epoch):
                raw = self.predict_raw(xi)
                error = yi - raw
                self.w += self.lr * error * xi
                self.b += self.lr * error
                mse += error**2

            mse /= n
            self.mse_history.append(mse)
            print(f"Эпоха {epoch+1:2d}: MSE = {mse:.4f}")

    def decision_boundary(self, x_vals):
        if abs(self.w[1]) < 1e-12:
            return np.full_like(x_vals, np.nan)
        return (-(self.w[0] * x_vals + self.b) / self.w[1])

def learning_rate_study(X, y, rates, epochs=30):
    results = {}
```

```
    for lr in rates:
        model = ADALINE(input_size=2, learning_rate=lr)
        model.fit_adaline(X, y, epochs=epochs, shuffle=True)
        results[lr] = model.mse_history
    return results

def visualize(X, y, model, new_point=None):
    plt.figure(figsize=(10,5))

    plt.subplot(1,2,1)
    plt.plot(range(1, len(model.mse_history)+1), model.mse_history, 'b-o')
    plt.xlabel("Эпоха")
    plt.ylabel("MSE")
    plt.title("Ошибка по эпохам")
    plt.grid(True)

    plt.subplot(1,2,2)
    colors = ['red' if label == -1 else 'blue' for label in y]
    plt.scatter(X[:,0], X[:,1], c=colors, edgecolors='k', s=120)

    x_min, x_max = X[:,0].min()-1, X[:,0].max()+1
    x_vals = np.linspace(x_min, x_max, 200)
    y_vals = model.decision_boundary(x_vals)
    if not np.isnan(y_vals).all():
        plt.plot(x_vals, y_vals, 'g--', linewidth=2, label='Разделяющая линия')

    if new_point is not None:
        p = np.array(new_point)
        pred = model.predict(p)
        color = 'blue' if pred == 1 else 'red'
        plt.scatter(p[0], p[1], c=color, s=200, marker='*', edgecolors='k')
        plt.text(p[0], p[1], f"Класс: {pred}", fontsize=10)

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

if __name__ == "__main__":
    X, y = load_custom_dataset()

    rates = [0.001, 0.01, 0.05, 0.1]
    lr_results = learning_rate_study(X, y, rates, epochs=40)

    model = ADALINE(input_size=2, learning_rate=0.05)
    model.fit_adaline(X, y, epochs=50)

    new_point = np.array([0.0, 0.0])
    visualize(X, y, model, new_point=new_point)

    preds = np.array([model.predict(x) for x in X])
    acc = np.mean(preds == y) * 100
    print(f"Веса: {model.w}, bias: {model.b:.4f}")
    print(f"Точность на обучающей выборке: {acc:.1f}%")
```

Результат тестирования:

Эпоха 1: MSE = 1.7240
Эпоха 2: MSE = 1.1707
Эпоха 3: MSE = 0.8946
Эпоха 4: MSE = 1.0833
Эпоха 5: MSE = 0.5605
Эпоха 6: MSE = 0.4091
Эпоха 7: MSE = 0.5768
Эпоха 8: MSE = 0.7415
Эпоха 9: MSE = 0.5941
Эпоха 10: MSE = 0.7924
Эпоха 11: MSE = 0.3215
Эпоха 12: MSE = 0.5457
Эпоха 13: MSE = 0.8186
Эпоха 14: MSE = 0.6666
Эпоха 15: MSE = 0.6942
Эпоха 16: MSE = 0.2947
Эпоха 17: MSE = 0.5756
Эпоха 18: MSE = 0.6139
Эпоха 19: MSE = 0.5569
Эпоха 20: MSE = 0.6761
Эпоха 21: MSE = 0.7125

Эпоха 22: MSE = 0.4803
Эпоха 23: MSE = 0.5620
Эпоха 24: MSE = 0.6180
Эпоха 25: MSE = 0.7065
Эпоха 26: MSE = 0.8464
Эпоха 27: MSE = 0.6548
Эпоха 28: MSE = 0.3024
Эпоха 29: MSE = 0.5430
Эпоха 30: MSE = 0.7741
Эпоха 31: MSE = 0.6687
Эпоха 32: MSE = 0.6831
Эпоха 33: MSE = 0.6703
Эпоха 34: MSE = 0.6653
Эпоха 35: MSE = 0.6115
Эпоха 36: MSE = 0.8467
Эпоха 37: MSE = 0.6888
Эпоха 38: MSE = 0.2582
Эпоха 39: MSE = 0.2586
Эпоха 40: MSE = 0.7914
Эпоха 41: MSE = 0.6372
Эпоха 42: MSE = 0.5306
Эпоха 43: MSE = 0.5990
Эпоха 44: MSE = 0.3459
Эпоха 45: MSE = 0.8230
Эпоха 46: MSE = 0.6236
Эпоха 47: MSE = 0.5319
Эпоха 48: MSE = 0.8835
Эпоха 49: MSE = 0.2831
Эпоха 50: MSE = 0.5486
Веса: [0.18299149 0.61341057], bias: 0.4402
Точность на обучающей выборке: 100.0%

График с разделяющей поверхностью:

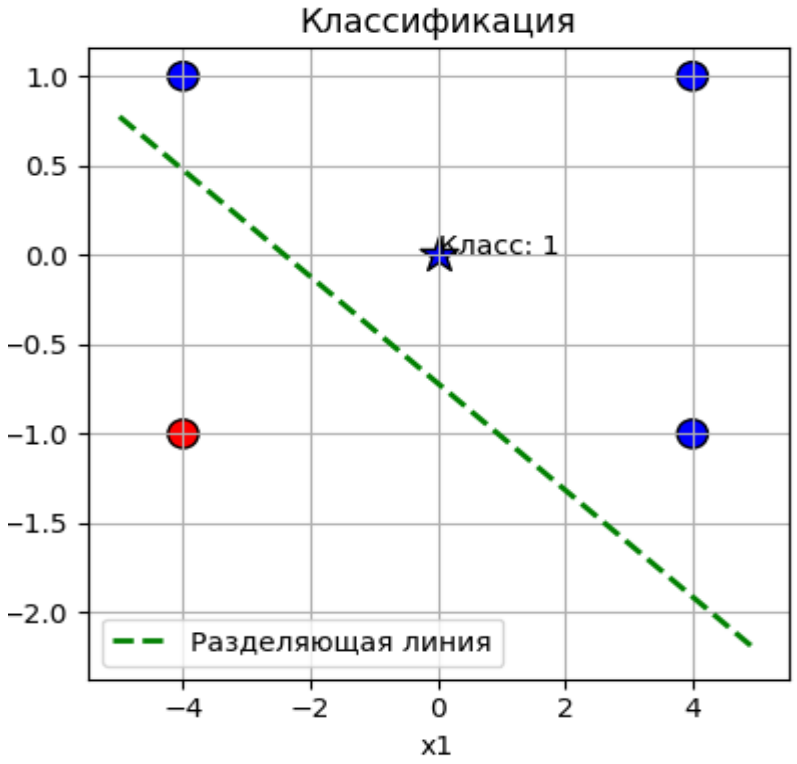
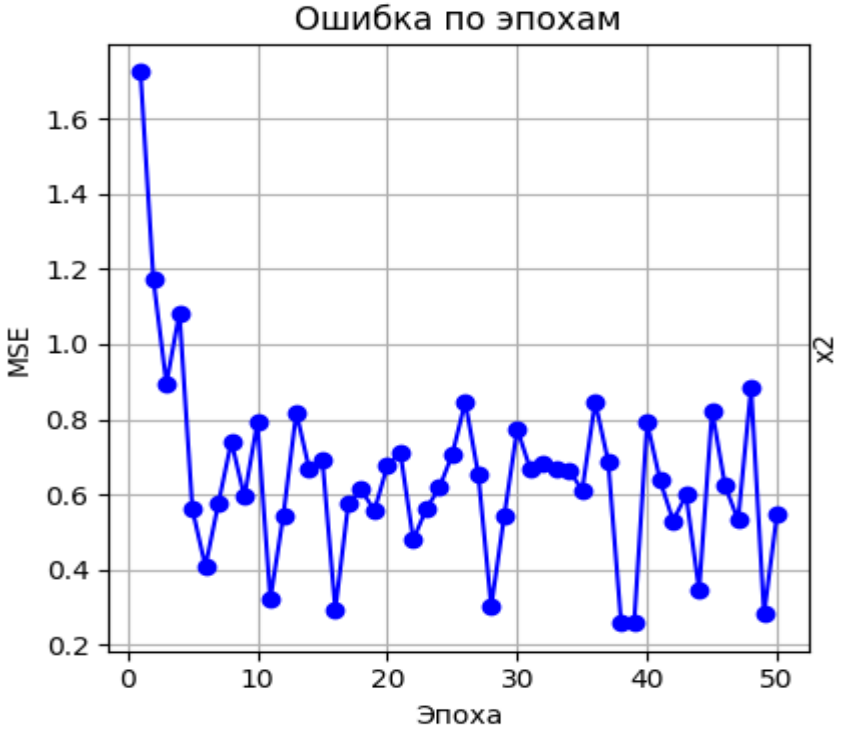


График изменения ошибки:



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).