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
Рудь В.В.
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 1:

x_1	x_2	e
2	4	0
-2	4	0
2	-4	1
-2	-4	1

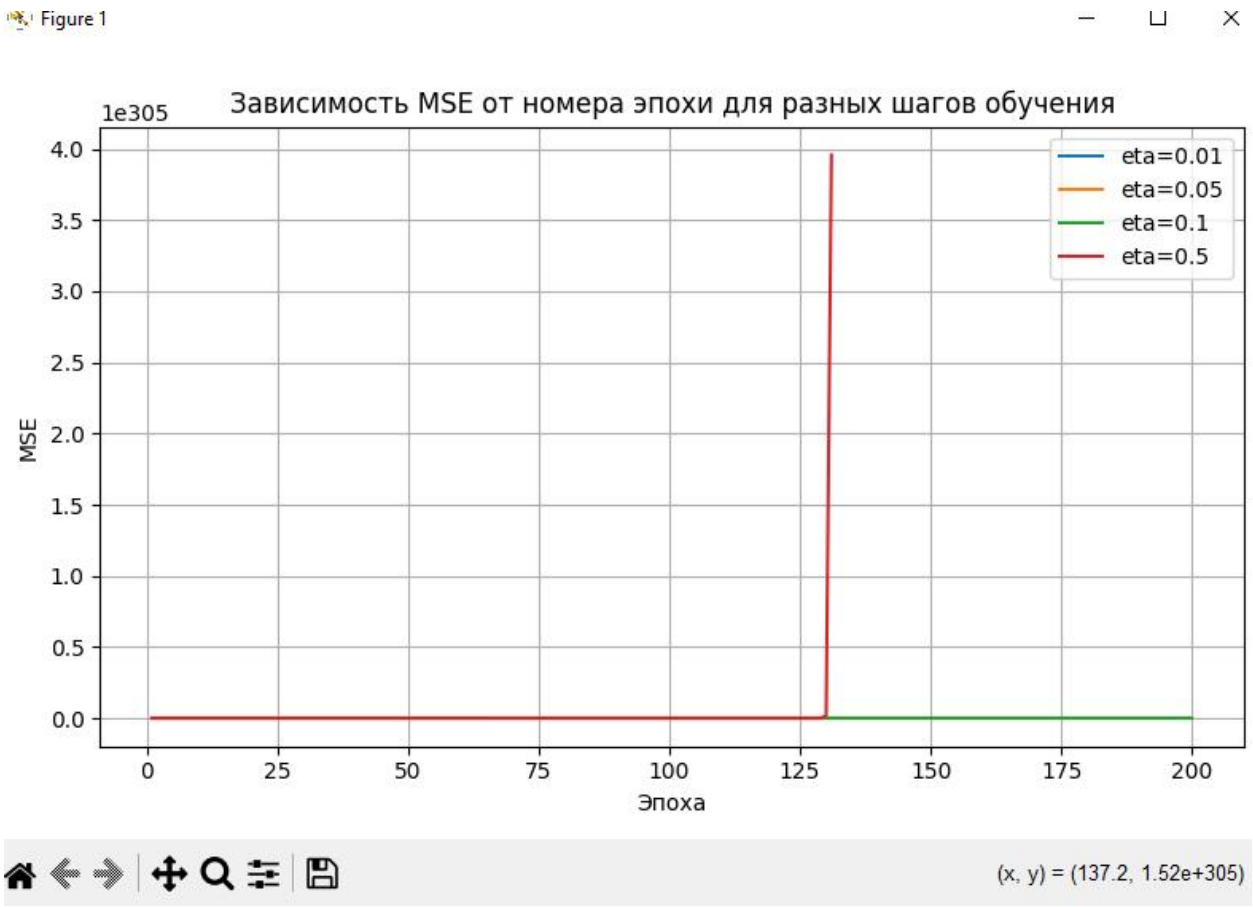
Ход работы :

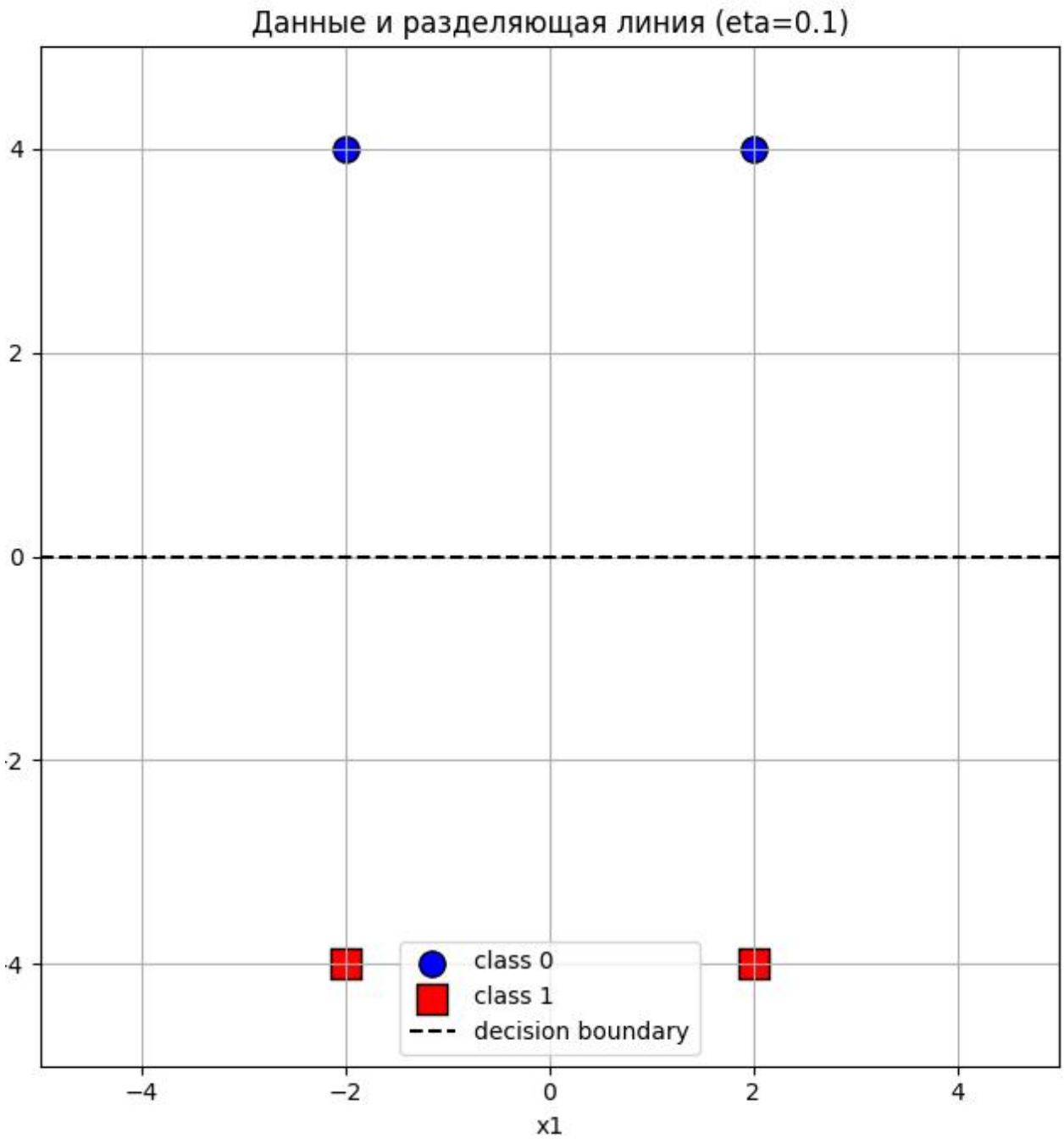
Код программы:

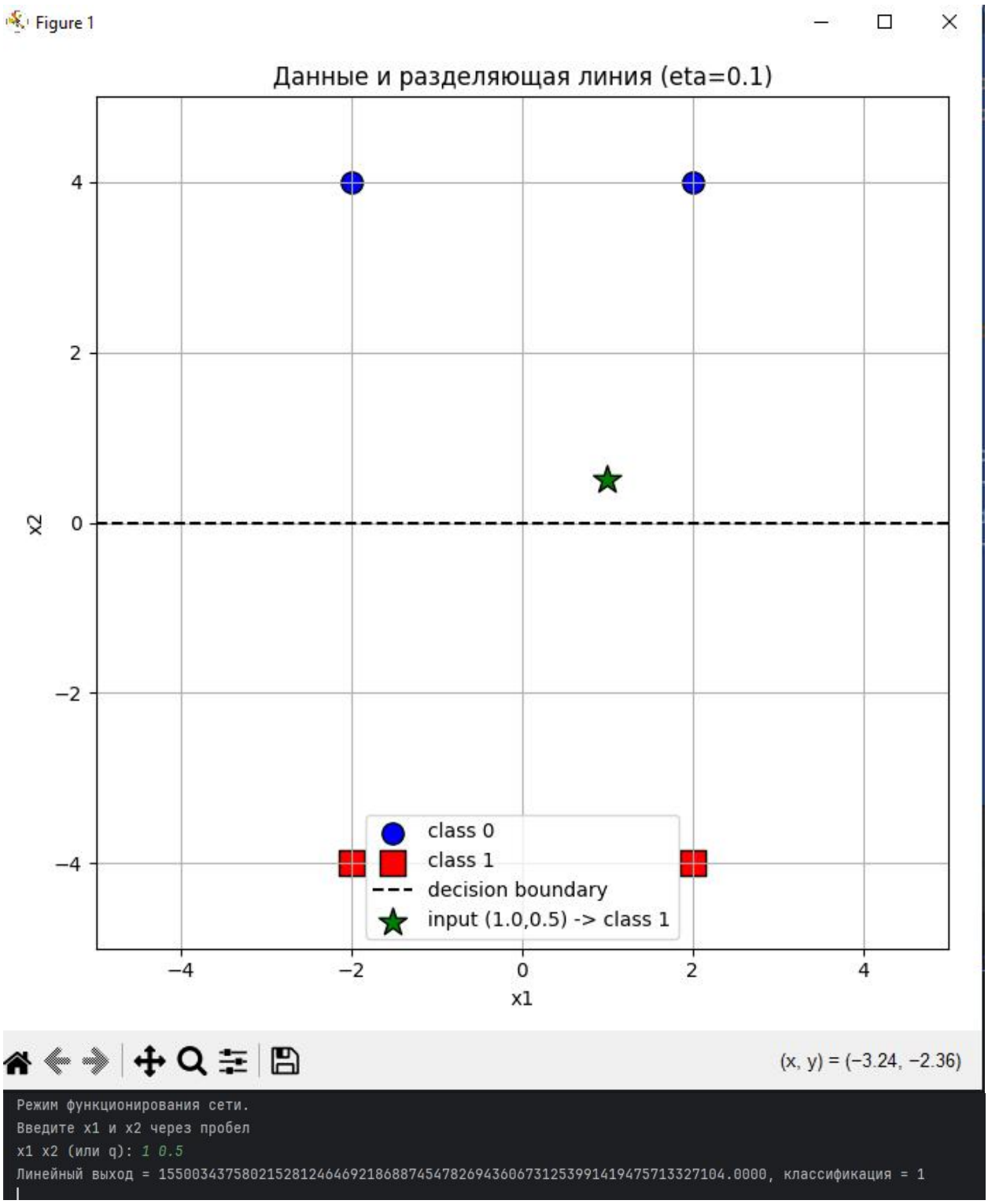
```
import numpy as np
import matplotlib.pyplot as plt

X = np.array([
    [2.0,  4.0],
    [-2.0, 4.0],
    [2.0, -4.0],
    [-2.0,-4.0]
], dtype=float)

T = np.array([0.0, 0.0, 1.0, 1.0], dtype=float)

epochs = 200
learning_rates = [0.01, 0.05, 0.1, 0.5]
seed = 42
np.random.seed(seed)


def init_weights():
    w = np.random.randn(2) * 0.1
    b = 0.0
    return w, b

def linear_output(X, w, b):
    return X.dot(w) + b

def step(x):
    return (x >= 0).astype(float)

def train_delta_rule_batch(X, T, w_init, b_init, eta, epochs):


    n = X.shape[0]
    w = w_init.copy()
    b = b_init
    mse_history = []
    for ep in range(epochs):
        y_lin = linear_output(X, w, b)
        errors = T - y_lin
        mse = np.mean(errors**2)
        mse_history.append(mse)
```

```
        grad_w = -2.0 / n * X.T.dot(errors)
        grad_b = -2.0 / n * np.sum(errors)
        w = w - eta * grad_w
        b = b - eta * grad_b
    return w, b, mse_history

results = {}
for eta in learning_rates:
    w0, b0 = init_weights()
    w_tr, b_tr, mse_hist = train_delta_rule_batch(X, T, w0, b0, eta, epochs)
    results[eta] = {'w': w_tr, 'b': b_tr, 'mse': mse_hist}


plt.figure(figsize=(8,5))
for eta, res in results.items():
    plt.plot(range(1, epochs+1), res['mse'], label=f'eta={eta}')
plt.xlabel('Эпоха')
plt.ylabel('MSE')
plt.title('Зависимость MSE от номера эпохи для разных шагов обучения')
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

margin = 1.0
x_min, x_max = X[:,0].min() - margin, X[:,0].max() + margin
y_min, y_max = X[:,1].min() - margin, X[:,1].max() + margin
low = min(x_min, y_min)
high = max(x_max, y_max)
plot_range = (low, high)

def plot_data_and_boundary(X, T, w, b, x_range=plot_range, show_point=None, title_suffix=''):
    plt.figure(figsize=(7,7))
    class0 = X[T==0]
    class1 = X[T==1]
    plt.scatter(class0[:,0], class0[:,1], c='blue', marker='o', s=120, edgecolor='k', label='class 0')
    plt.scatter(class1[:,0], class1[:,1], c='red', marker='s', s=160, edgecolor='k', label='class 1')
    x_vals = np.linspace(x_range[0], x_range[1], 400)
    if abs(w[1]) > 1e-8:
        y_vals = -(w[0]/w[1])*x_vals - b/w[1]
        plt.plot(x_vals, y_vals, 'k--', linewidth=1.5, label='decision boundary')
    else:
        if abs(w[0]) > 1e-8:
            x_vert = -b / w[0]
            plt.axvline(x=x_vert, color='k', linestyle='--', linewidth=1.5, label='decision boundary')
    if show_point is not None:
        x1u, x2u, cls = show_point
        plt.scatter([x1u], [x2u], c='green', marker='*', s=220, edgecolor='k', label=f'input ({x1u},{x2u}) -> class
{int(cls)}')
    plt.xlim(x_range)
    plt.ylim(x_range)
    plt.xlabel('x1')
    plt.ylabel('x2')
    plt.title(f'Данные и разделяющая линия {title_suffix}')
    plt.legend()
    plt.grid(True)
    plt.gca().set_aspect('equal', adjustable='box')
    plt.tight_layout()
    plt.show()

eta_show = 0.1 if 0.1 in results else list(results.keys())[0]
w_show = results[eta_show]['w']
b_show = results[eta_show]['b']
```

```
plot_data_and_boundary(X, T, w_show, b_show, x_range=plot_range, title_suffix=f'(eta={eta_show})')

print("\nРежим функционирования сети.")
print(f"Введите x1 и x2 через пробел ")
while True:
    s = input("x1 x2 (или q): ").strip()
    if s.lower() == 'q':
        break
    try:
        parts = s.split()
        if len(parts) != 2:
            print("Нужно ввести два числа через пробел.")
            continue
        x1u = float(parts[0])
        x2u = float(parts[1])
        y_lin = w_show[0]*x1u + w_show[1]*x2u + b_show
        y_bin = 1.0 if y_lin >= 0 else 0.0
        print(f"Линейный выход = {y_lin:.4f}, классификация = {int(y_bin)}")
        plot_data_and_boundary(X, T, w_show, b_show, x_range=plot_range, show_point=(x1u, x2u, y_bin),
title_suffix=f'(eta={eta_show})')
    except ValueError:
        print("Ошибка: введите корректные числа.")
```




Данные и разделяющая линия (eta=0.1)
4
2
0
2
4
−4
−2
0
2
4
x1
class 0
class 1
decision boundary



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).