Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №1

По дисциплине: «Модели решения задач в интеллектуальных системах»

Тема: «Бинарная классификация»

Выполнил:

Студент 3 курса

Группы ИИ-26(1)

Пасевич К.Ю.

Проверила:

Андренко 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).

Ход 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, например $-0.5 \leq x_1, x_2 \leq 1.5$.

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 9

9.

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

x_1, x_2 - входные данные сети, e - эталонные значения

Код программы:

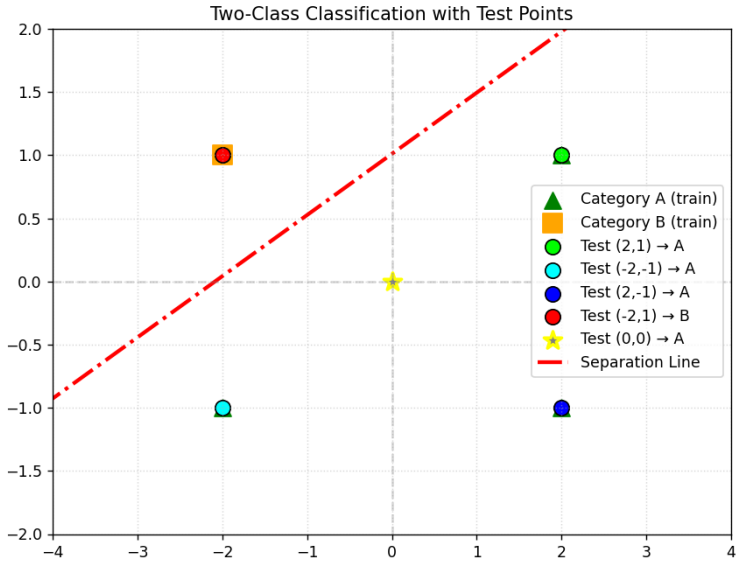
```
import numpy as np
import matplotlib.pyplot as plt

data_points = np.array([[2, 1], [-2, 1], [2, -1], [-2, -1]])
targets = np.array([0, 1, 0, 0])
```

```
alpha = 0.01
iterations = 200
np.random.seed(42)

weights = np.random.uniform(-0.5, 0.5, 2)
bias = np.random.uniform(-0.5, 0.5)

error_log = []

for step in range(iterations):
    step_errors = []
    for j in range(len(data_points)):
        linear_output = np.dot(data_points[j], weights) + bias
        error = targets[j] - linear_output
        weights += alpha * error * data_points[j]
        bias += alpha * error
        step_errors.append(error ** 2)
    error_log.append(np.mean(step_errors))

plt.figure(figsize=(8, 4))
plt.plot(error_log, 'purple', linewidth=2)
plt.title("Training Error Progress")
plt.xlabel("Iteration Number")
plt.ylabel("Loss Value")
plt.grid(True, linestyle=':', alpha=0.7)
plt.show()

plt.figure(figsize=(8, 6))

plt.scatter(data_points[0, 0], data_points[0, 1], color='green',
s=120, marker='^', label='Category A')
plt.scatter(data_points[1, 0], data_points[1, 1], color='orange',
s=150, marker='s', label='Category B')
plt.scatter(data_points[2, 0], data_points[2, 1], color='green',
s=120, marker='^')
plt.scatter(data_points[3, 0], data_points[3, 1], color='green',
s=120, marker='^')

x_vals = np.linspace(-10, 10, 100)
y_vals = (0.5 - bias - weights[0] * x_vals) / weights[1]
plt.plot(x_vals, y_vals, 'r-.', linewidth=2.5, label='Separation
Line')

plt.xlim(-4, 4)
plt.ylim(-2, 2)
plt.title("Two-Class Classification Map")
plt.axhline(0, color='gray', alpha=0.3, linestyle='--')
plt.axvline(0, color='gray', alpha=0.3, linestyle='--')
plt.legend()
```

```
plt.grid(True, linestyle=':', alpha=0.5)
plt.show()

print("\nPrediction Mode:")
print(f"Model parameters: w_vector = {weights}, bias_term = {bias}")
print("\nAutomated testing on predefined points:")
print("-" * 50)

test_points = np.array([[2, 1], [-2, -1], [2, -1], [-2, 1], [0, 0]])
point_names = ["(2, 1)", "(-2, -1)", "(2, -1)", "(-2, 1)", "(0, 0)"]

for idx, point in enumerate(test_points):
    activation = np.dot(point, weights) + bias
    prediction = 1 if activation >= 0.5 else 0
    expected = "B" if np.array_equal(point, [-2, 1]) else "A"
    print(f"Point {point_names[idx]}: Linear response =
{activation:.4f} → Group {prediction} (expected: {expected})")

print("\nVisualizing test points on graph...")
plt.figure(figsize=(8, 6))

plt.scatter(data_points[0, 0], data_points[0, 1], color='green',
s=120, marker='^', label='Category A (train)')
plt.scatter(data_points[1, 0], data_points[1, 1], color='orange',
s=150, marker='s', label='Category B (train)')
plt.scatter(data_points[2, 0], data_points[2, 1], color='green',
s=120, marker='^')
plt.scatter(data_points[3, 0], data_points[3, 1], color='green',
s=120, marker='^')

plt.scatter(test_points[0, 0], test_points[0, 1], color='lime', s=100,
marker='o', edgecolors='black', label='Test (2,1) → A')
plt.scatter(test_points[1, 0], test_points[1, 1], color='cyan', s=100,
marker='o', edgecolors='black', label='Test (-2,-1) → A')
plt.scatter(test_points[2, 0], test_points[2, 1], color='blue', s=100,
marker='o', edgecolors='black', label='Test (2,-1) → A')
plt.scatter(test_points[3, 0], test_points[3, 1], color='red', s=100,
marker='o', edgecolors='black', label='Test (-2,1) → B')
plt.scatter(test_points[4, 0], test_points[4, 1], color='gray', s=150,
marker='*', edgecolors='yellow', linewidth=2, label='Test (0,0) → A')

x_vals = np.linspace(-10, 10, 100)
y_vals = (0.5 - bias - weights[0] * x_vals) / weights[1]
plt.plot(x_vals, y_vals, 'r-.', linewidth=2.5, label='Separation
Line')

plt.xlim(-4, 4)
plt.ylim(-2, 2)
plt.title("Two-Class Classification with Test Points")
plt.axhline(0, color='gray', alpha=0.3, linestyle='--')
```

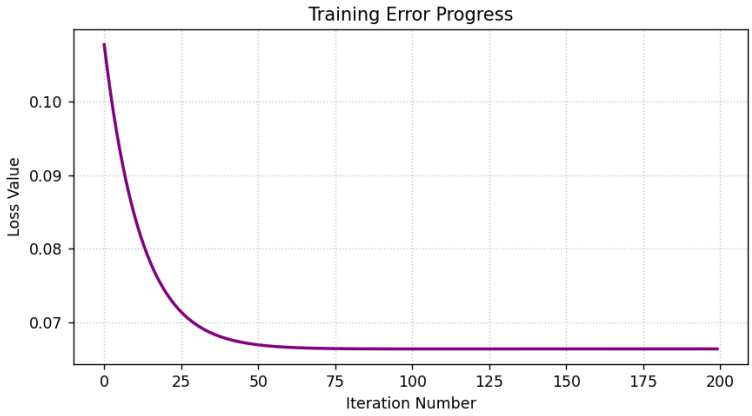
```
plt.axvline(0, color='gray', alpha=0.3, linestyle='--')
plt.legend()
plt.grid(True, linestyle=':', alpha=0.5)
plt.show()

print("\nTesting complete!")
```

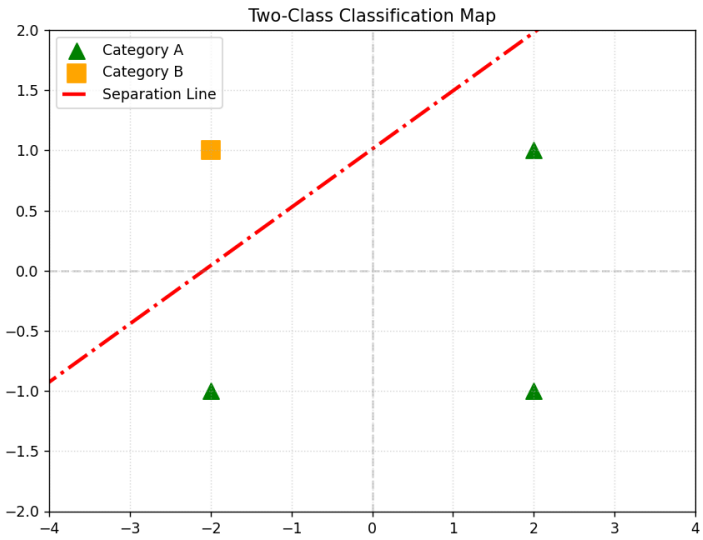
Результат тестирования:



Prediction Mode:
Model parameters: w_vector = [-0.11984541 0.24747591], bias_term = 0.24999530092575362
Automated testing on predefined points:
Point (2, 1): Linear response = 0.2578 → Group 0 (expected: A)
Point (-2, -1): Linear response = 0.2422 → Group 0 (expected: A)
Point (2, -1): Linear response = -0.2372 → Group 0 (expected: A)
Point (-2, 1): Linear response = 0.7372 → Group 1 (expected: B)
Point (0, 0): Linear response = 0.2500 → Group 0 (expected: A)

График среднеквадратичной ошибки и график с разделяющей поверхностью:





Вывод: в процессе выполнения работы был реализован персептрон для задачи бинарной классификации, освоен принцип его обучения на основе пороговой функции активации и выполнена оценка сходимости с помощью среднеквадратичной ошибки.