

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»

Выполнила:
Студент 3 курса
Группы ИИ-26
Карпович К.А.
Проверила:
Андренко К.В.

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).

Задачи лабораторной работы:

1. Реализовать алгоритм обучения однослойной нейронной сети с использованием **MSE** в качестве функции ошибки (**дельта-правило**).
 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 $-0.5 \leq x_1, x_2 \leq 1.5$
 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 6

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

x_1	x_2	e
3	6	0
-3	6	0
3	-6	1
-3	-6	1

Код программы:

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

X = np.array([[3, 6], [-3, 6], [3, -6], [-3, -6]])
e = np.array([0, 0, 1, 1])

# Параметры
learning_rate = 0.01
epochs = 200
np.random.seed(42)

# Инициализация весов и смещения (порога  $w_0$ )
w = np.random.uniform(-0.5, 0.5, 2)
```

```
w0 = np.random.uniform(-0.5, 0.5)
```

```
mse_history = []
```

```
# ЭТАП 1: ОБУЧЕНИЕ (ADALINE)
```

```
for epoch in range(epochs):
```

```
    epoch_errors = []
```

```
    for i in range(len(X)):
```

```
        # Линейный выход сумматора
```

```
        S = np.dot(X[i], w) + w0
```

```
        # Ошибка (строго линейная)
```

```
        delta = e[i] - S
```

```
        # Коррекция по дельта-правилу
```

```
        w += learning_rate * delta * X[i]
```

```
        w0 += learning_rate * delta
```

```
    epoch_errors.append(delta ** 2)
```

```
mse_history.append(np.mean(epoch_errors))
```

```
# ВИЗУАЛИЗАЦИЯ
```

```
# График изменения ошибки
```

```
plt.figure(figsize=(8, 4))
```

```
plt.plot(mse_history, 'b', linewidth=2)
```

```
plt.title("График изменения ошибки (MSE)")
```

```
plt.xlabel("Эпоха")
```

```
plt.ylabel("Среднеквадратичная ошибка")
```

```
plt.grid(True)
```

```
plt.show()
```

```
# График с визуализацией условия и разделяющей поверхности
```

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

```
plt.scatter(X[:, 2], X[:, 1], color='blue', s=100, label='Класс 0')
```

```
plt.scatter(X[:, 2], X[:, 1], color='red', s=100, label='Класс 1')
```

```
# Отрисовка разделяющей поверхности
```

```
x_line = np.linspace(-10, 10, 100)
```

```
y_line = (0.5 - w0 - w[0] * x_line) / w[1]
```

```
plt.plot(x_line, y_line, 'g--', label='Разделяющая поверхность')
```

```
plt.xlim(-10, 10)
```

```
plt.ylim(-10, 10)
```

```
plt.title("Визуализация разделения областей 2-х классов")
```

```
plt.axhline(0, color='black', alpha=0.3)
```

```
plt.axvline(0, color='black', alpha=0.3)
```

```
plt.legend()
```

```
plt.grid(True)
```

```
plt.show()
```

ЭТАП 2: ФУНКЦИОНИРОВАНИЕ

```
print("\nРежим тестирования:")
```

```
while True:
```

```
    user_input = input("Введите x1, x2 через запятую или 'q' для выхода: ")
```

```
    if user_input.lower() == 'q': break
```

```
    try:
```

```
        test_point = np.array([float(c.strip()) for c in user_input.split(',')])
```

```
        S_test = np.dot(test_point, w) + w0
```

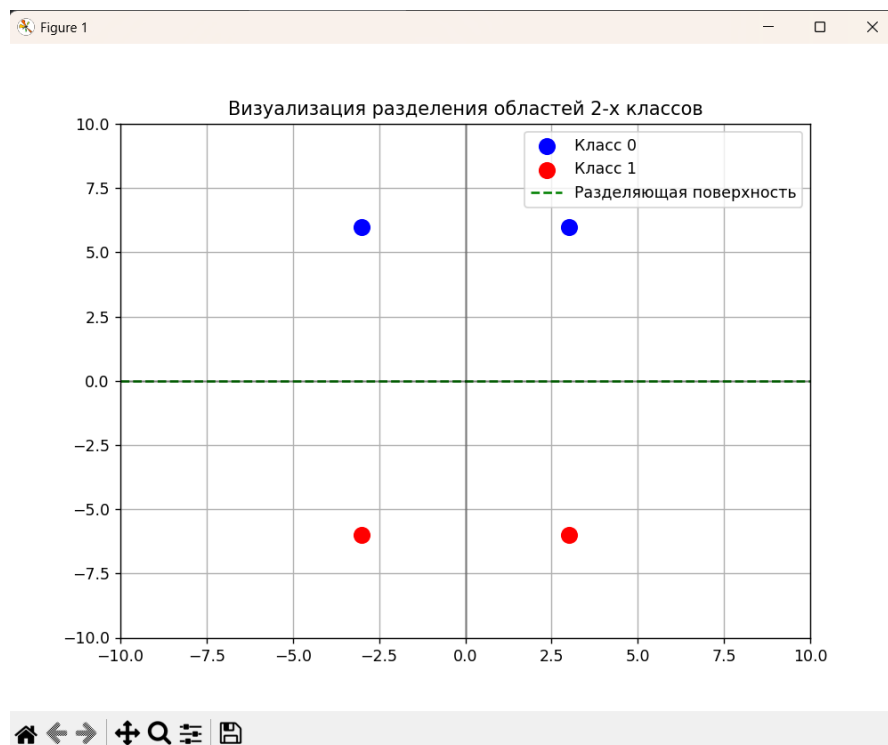
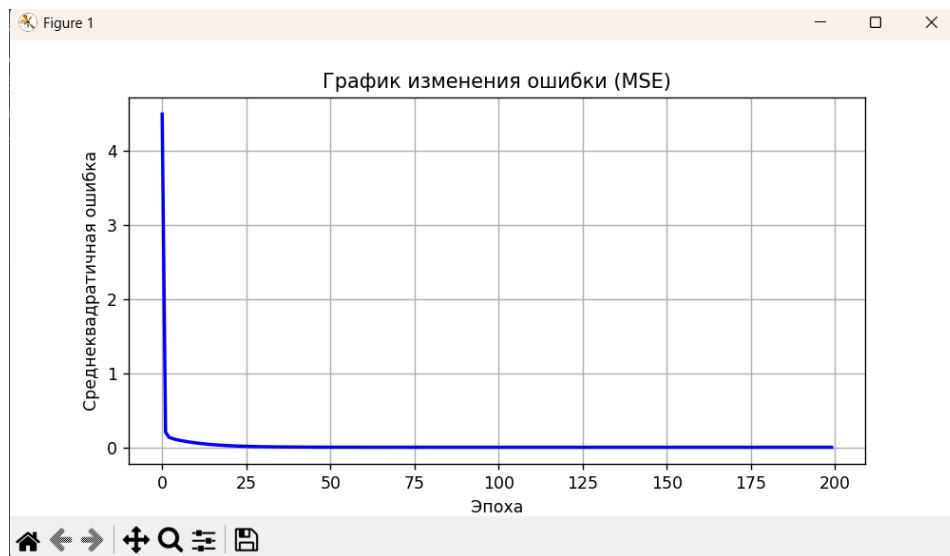
```
        y_final = 1 if S_test >= 0.5 else 0
```

```
        print(f"Результат: Класс {y_final} (S = {S_test:.4f})")
```

```
    except:
```

```
        print("Ошибка ввода")
```

Результат программы:



```
Режим тестирования:  
Введите x1, x2 через запятую или 'q' для выхода: 3, 6  
Результат: Класс 0 (S = -0.0000)  
Введите x1, x2 через запятую или 'q' для выхода: -3, 6  
Результат: Класс 0 (S = -0.0000)  
Введите x1, x2 через запятую или 'q' для выхода: 3, -6  
Результат: Класс 1 (S = 1.0000)  
Введите x1, x2 через запятую или 'q' для выхода: -3, -6  
Результат: Класс 1 (S = 1.0000)  
Введите x1, x2 через запятую или 'q' для выхода: q  
  
Process finished with exit code 0
```

В ходе тестирования нейросеть безошибочно разделила все точки по их классам: для первого класса она выдала значения, близкие к 0, а для второго — ровно 1. Это доказывает, что обучение прошло успешно, и программа правильно определяет класс для любых координат из задания.

Вывод: изучила принципы бинарной классификации и реализовала однослойную нейронную сеть (персептрон) для решения задачи классификации с использованием **пороговой** функции активации, а также исследовала процесс обучения модели с применением среднеквадратичной ошибки (MSE).