МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТк

ФАКУЛЬТЕТ ЭЛЕКТРОННО-ИНФОРМАЦИОННЫХ СИСТЕМ

Кафедра интеллектуальных информационных технологий

Отчет по лабораторной работе №1

Специальность ИИ26(з)

Выполнил
Е.Д. Вирко,
студент группы ИИ-26

Проверил
К.В. Андренко,
ст. преп. кафедры ИИТ,
«___»_____2026 г.

Брест 2026

Цель работы: Изучить принципы бинарной классификации и реализо- вать однослойную нейронную сеть (персептрон) для решения задачи классификации с использованием пороговой функции активации, а также исследовать процесс обучения модели с применением среднеквадратич- ной ошибки (MSE).

Задание 1. Для заданного массива вещественных чисел найти среднее значение и стандартное отклонение.

Задачи лабораторной 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 ВСТАВИТЬ СВОЙ ДИАПАЗОН, например -0.5≤x1,x2≤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 1

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

Код программы:

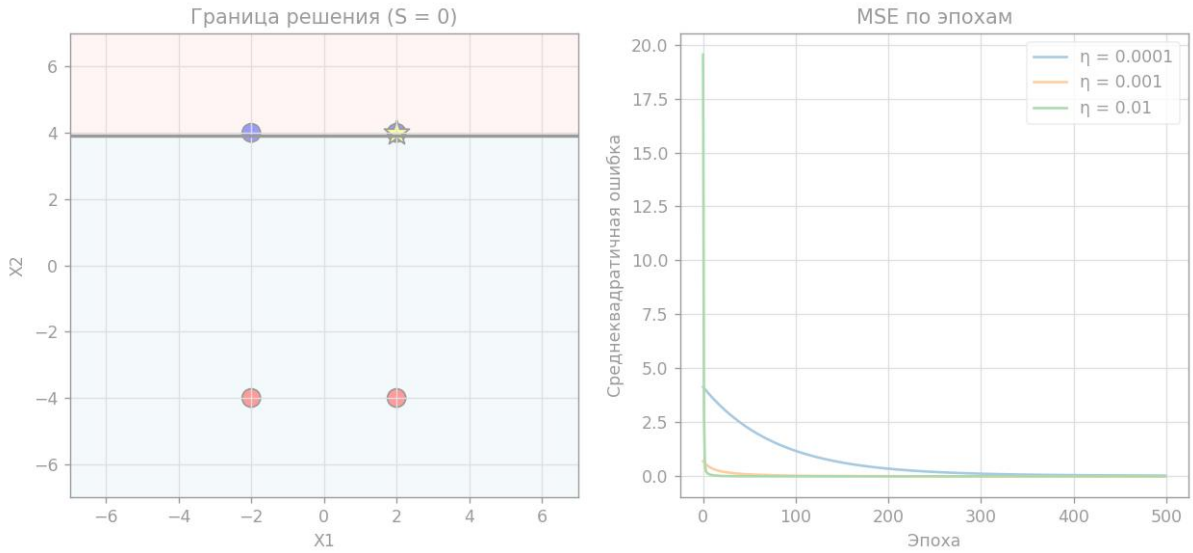
```
import numpy as np
import matplotlib.pyplot as plt


class LinearClassifier:

    def __init__(self, n_features, eta):
```

```
        self.lr = eta
        self.weights = np.random.randn(n_features + 1)
        self.inputs = None
        self.targets = None

    def load_data(self, X, y):
        X = np.asarray(X)

        if X.ndim == 1:
            X = X.reshape(1, -1)

        bias_column = -np.ones((X.shape[0], 1))
        self.inputs = np.hstack((bias_column, X))
        self.targets = np.asarray(y)

    def net(self, matrix):
        return matrix @ self.weights

    def step(self, values):
        return (values > 0).astype(int)

    def forward(self, X=None):
        data = self.inputs if X is None else X
        return self.step(self.net(data))

    def update_weights(self, err_vector):
        gradient = err_vector @ self.inputs
        self.weights -= self.lr * gradient

    def compute_mse(self, err_vector):
        return np.mean(0.5 * err_vector ** 2)

    def fit(self, n_epochs):
        history = []

        for _ in range(n_epochs):
            linear_output = self.net(self.inputs)
            error = linear_output - self.targets
            history.append(self.compute_mse(error))
            self.update_weights(error)

        return history


X_data = np.array([
    [2, 4],
    [-2, 4],
    [2, -4],
    [-2, -4]
])

e = np.array([0, 0, 1, 1])
```

```
plt.figure(figsize=(12, 5))


plt.subplot(1, 2, 2)

for rate in (1e-4, 1e-3, 1e-2):
    model = LinearClassifier(2, rate)
    model.load_data(X_data, e)
    mse_curve = model.fit(500)
    plt.plot(mse_curve, label=f"η = {rate}")

plt.title("MSE по эпохам")
plt.xlabel("Эпоха")
plt.ylabel("Среднеквадратичная ошибка")
plt.grid(True)
plt.legend()


main_model = LinearClassifier(2, 0.001)
main_model.load_data(X_data, e)
main_model.fit(1000)


def draw_scene(point=None, predicted=None):
    plt.subplot(1, 2, 1)

    x_axis = np.linspace(-7, 7, 120)
    y_axis = np.linspace(-7, 7, 120)
    grid_x, grid_y = np.meshgrid(x_axis, y_axis)

    grid_stack = np.column_stack((
        -np.ones(grid_x.size),
        grid_x.ravel(),
        grid_y.ravel()
    ))

    z_class = main_model.forward(grid_stack).reshape(grid_x.shape)
    z_line = main_model.net(grid_stack).reshape(grid_x.shape)

    plt.contourf(grid_x, grid_y, z_class,
                 levels=[-0.1, 0.5, 1.1],
                 colors=["#fde0dd", "#deebf7"],
                 alpha=0.8)

    plt.contour(grid_x, grid_y, z_line,
                levels=[0],
                colors="black",
                linewidths=2)

    plt.scatter(X_data[:, 0],
```

```
            X_data[:, 1],
            c=e,
            cmap="bwr",
            s=120,
            edgecolors="black")

    if point is not None:
        plt.scatter(point[0], point[1],
                    s=250,
                    marker="*",
                    color="yellow",
                    edgecolors="black")

    plt.xlim(-7, 7)
    plt.ylim(-7, 7)
    plt.xlabel("X1")
    plt.ylabel("X2")
    plt.title("Граница решения (S = 0)")
    plt.grid(True)


draw_scene()
plt.show(block=False)


print("\nВведите координаты X1 X2 (или exit)\n")

while True:
    user_input = input("X1 X2: ")

    if user_input.lower() in ("exit", "выход"):
        break

    parts = user_input.split()
    if len(parts) != 2:
        continue

    x1, x2 = map(float, parts)

    test_vector = np.array([[-1, x1, x2]])
    result = main_model.forward(test_vector)[0]

    print(f"Предсказанный класс: {result} "
          f"(0 — верхняя область, 1 — нижняя область)")

    plt.subplot(1, 2, 1)
    plt.cla()
    draw_scene((x1, x2), result)
    plt.draw()
    plt.pause(0.1)

plt.show()
```

Результат работы программы



Вывод: Изучил принципы бинарной классификации и реализовал однослойную нейронную сеть (персептрон) для решения задачи класификации с использованием пороговой функции активации, а также исследовал процесс обучения модели с применением среднеквадратичной ошибки (MSE).