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 $-0.5 \le x1, x2 \le 1.5$

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:

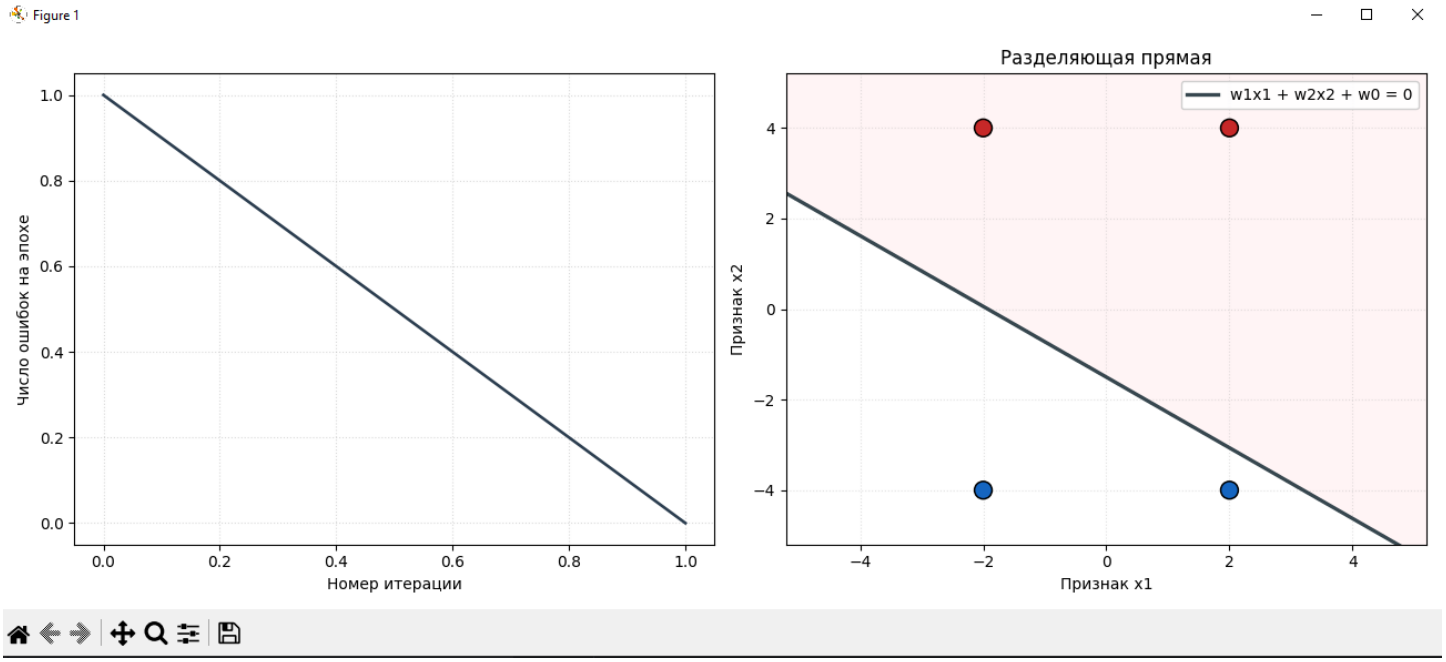
```
import numpy as np
import matplotlib.pyplot as plt

class SimplePerceptron:
    def __init__(self, n_inputs, eta=0.12):
        self.weights = np.random.randn(n_inputs + 1) * 0.4
        self.eta = eta

    def _with_bias(self, X):
```

```
        X = np.atleast_2d(X)
        ones = np.ones((X.shape[0], 1))
        return np.hstack([ones, X])

    def _compute_net(self, X):
        return np.dot(X, self.weights)

    def classify(self, X):
        Xb = self._with_bias(X)
        nets = self._compute_net(Xb)
        return np.where(nets >= 0.0, 1, 0)

    def fit(self, X, targets, max_iter=800):
        Xb = self._with_bias(X)
        targets = np.asarray(targets, dtype=int)
        mistakes_log = []

        for it in range(max_iter):
            wrong = 0

            for x_vec, true_y in zip(Xb, targets):
                y_hat = 1 if self._compute_net(x_vec) >= 0 else 0
                delta = true_y - y_hat

                if delta != 0:
                    self.weights += self.eta * delta * x_vec
                    wrong += 1

            mistakes_log.append(wrong)

            if wrong == 0:
                print(f"Сходимость достигнута на итерации {it+1}")
                break

        return mistakes_log


"""
ИКСЫ И ЕШКИ
"""

points = np.array([[2,4], [-2,4], [2,-4], [-2,-4]])
labels  = np.array([0, 0, 1, 1])

model = SimplePerceptron(n_inputs=2, eta=0.12)
errors_per_epoch = model.fit(points, labels, max_iter=600)


"""
Графики
"""

fig = plt.figure(figsize=(13, 5.2))

"""
график ошибок
"""

ax1 = fig.add_subplot(1, 2, 1)
ax1.plot(errors_per_epoch, lw=1.9, color='#2c3e50')
ax1.set_xlabel('Номер итерации')
ax1.set_ylabel('Число ошибок на эпохе')
```

```
ax1.grid(ls=':', alpha=0.45)

"""
Разделяющая поверхность
"""

ax2 = fig.add_subplot(1, 2, 2)

x_min, x_max = -5.2, 5.2
y_min, y_max = -5.2, 5.2

xx, yy = np.meshgrid(np.linspace(x_min, x_max, 280),
                     np.linspace(y_min, y_max, 280))

grid_points = np.c_[xx.ravel(), yy.ravel()]
pred_grid = model.classify(grid_points).reshape(xx.shape)

ax2.contourf(xx, yy, pred_grid,
             levels=[-0.1, 0.5, 1.1],
             colors=['#ffebee', 'white', '#e3f2fd'],
             alpha=0.55)
ax2.contour(xx, yy, pred_grid, levels=[0.5],
            colors='gray', linewidths=1.4, linestyles='--', alpha=0.8) # lw и ls

"""
Точки
"""

cls_colors = ['#c62828' if v==0 else '#1565c0' for v in labels]
ax2.scatter(points[:,0], points[:,1], c=cls_colors,
            s=130, edgecolor='k', lw=1.1, zorder=10)

"""
Разделение
"""

if abs(model.weights[2]) > 1e-9:
    x_vals = np.linspace(x_min, x_max, 180)
    y_vals = - (model.weights[0] + model.weights[1] * x_vals) / model.weights[2]
    ax2.plot(x_vals, y_vals, color='#37474f', lw=2.4,
             label='w1x1 + w2x2 + w0 = 0')

ax2.set_xlim(x_min, x_max)
ax2.set_ylim(y_min, y_max)
ax2.set_xlabel('Признак x1')
ax2.set_ylabel('Признак x2')
ax2.grid(ls=':', alpha=0.35)
ax2.legend(framealpha=0.92, loc='upper right')
ax2.set_title('Разделяющая прямая')

plt.tight_layout()
plt.show()



print("\nВведите два числа через пробел (x1 x2) или 'q' для выхода\n")

while True:
    s = input(": ").strip()
    if s.lower() in ('q', 'exit', 'выход'):
        break
    try:
```

```
    a, b = map(float, s.split())
    res = model.classify([a, b])[0]
    print(f"   : класс {res}  (0 = красный, 1 = синий)")
except:
    print("   некорректный ввод")
```

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).