

Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

Лабораторная работа №1
За пятый семестр
По дисциплине: «Модели решения задач в ИС»
Тема: «Бинарная классификация»

Выполнил:
Студент 3 курса
Группы ИИ-26(1)
Рубцов Д.А.
Проверила:
Андренко К.В.

Брест 2026

Цель: 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 средне-квадратичной 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

Код программы:

```
import numpy as np
import matplotlib
matplotlib.use('TkAgg')
import matplotlib.pyplot as plt

X = np.array([
    [6., 4.],
    [-6., 4.],
    [6., -4.],
    [-6., -4.]
])

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

def predict(w, x, T):
    return x @ w - T

def train_online(X, e, alpha=0.002, epochs=2000, shuffle=True, tol=1e-7):
    N = X.shape[0]
    w = np.zeros((X.shape[1], 1))
    T = 0.0
    errors = []
    prev_mse = float('inf')

    print(f"Обучение с  $\alpha = \{alpha:.6f\}$  начато...")

    for epoch in range(epochs):
        if shuffle:
            idx = np.random.permutation(N)
            X_shuf, e_shuf = X[idx], e[idx]
        else:
            X_shuf, e_shuf = X, e

        for i in range(N):
            xi = X_shuf[i:i+1]
            ti = e_shuf[i]
            y = predict(w, xi, T)
            err = y - ti
            grad_w = xi.T @ err
            grad_T = -err[0,0]
            w -= alpha * grad_w
            T -= alpha * grad_T

        y_all = predict(w, X, T)
```

```

        mse = np.mean((y_all - e) ** 2)
        errors.append(mse)

        if epoch % 100 == 0 and epoch > 0:
            print(f"Эпоха {epoch:4d} MSE = {mse:.8f}")

        if epoch > 30 and abs(mse - prev_mse) < tol:
            print(f"Ранняя остановка на эпохе {epoch}")
            break
        prev_mse = mse

    print(f"Обучение завершено. Финальная MSE = {mse:.8f}\n")
    return w, T, errors

alphas = [0.0003, 0.0006, 0.001, 0.002]
results = {}

for alpha in alphas:
    w, T, errs = train_online(X, e, alpha=alpha)
    results[alpha] = (w, T, errs)

plt.figure(figsize=(10, 5.5))
for alpha, (_, _, errs) in results.items():
    plt.plot(errs, label=f' $\alpha = {alpha}$ ', linewidth=1.6)
plt.yscale('log')
plt.xlabel('Эпоха')
plt.ylabel('MSE (лог. шкала)')
plt.grid(True, alpha=0.3)
plt.legend()
plt.tight_layout()
plt.show()

best_alpha = min(results, key=lambda a: results[a][2][-1])
w, T = results[best_alpha][0], results[best_alpha][1]
print(f"Выбрана модель с  $\alpha = {best_alpha:.6f}$  (лучшая MSE)\n")

new_points = []
new_classes = []

print("Введите точки (x1 x2) или exit для завершения\n")

while True:
    s = input("x1 x2 → ").strip()
    if not s or s.lower() in ['exit']:
        break

    try:
        x1, x2 = map(float, s.split())
        pt = np.array([x1, x2])

        S = predict(w, pt, T)[0, 0]
        cls = 1 if S >= 0.5 else 0
        color_name = "красный" if cls == 1 else "синий"

        print(f"S = {S:>9.6f}")
        print(f"Класс = {cls} ({color_name})")

        new_points.append([x1, x2])
        new_classes.append(cls)

        fig, ax = plt.subplots(figsize=(9, 8))

        c0 = X[e.flatten() == 0]
        c1 = X[e.flatten() == 1]
        ax.scatter(c0[:,0], c0[:,1], color='blue', s=150, label='класс 0', edgecolors='black')
        ax.scatter(c1[:,0], c1[:,1], color='red', s=150, label='класс 1', edgecolors='black')

        w1, w2 = w.flatten()

```

```

x1_vals = np.linspace(-9, 9, 500)
if abs(w2) > 1e-6:
    x2_vals = (T + 0.5 - w1 * x1_vals) / w2
ax.plot(x1_vals, x2_vals, 'g-', lw=2.4, label='линия (порог 0.5)')

```

```

for pt_xy, cl in zip(new_points, new_classes):
    col = 'red' if cl == 1 else 'blue'
ax.scatter(pt_xy[0], pt_xy[1], color=col, marker='X', s=220,
edgecolors='black', linewidth=1.6, alpha=0.95)

```

```

ax.set_xlim(-9, 9)
ax.set_ylim(-9, 9)
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.grid(True, alpha=0.25)
ax.legend()
plt.tight_layout()
plt.show()

```

```

except ValueError:
    print("Ошибка: введите два числа через пробел\n")
except Exception as ex:
    print(f"Ошибка: {ex}\n")

```

```

print("Программа завершена")

```

Пример работы программы:

```

Обучение с  $\alpha = 0.000300$  начато...
Эпоха 100 MSE = 0.50432395
Эпоха 200 MSE = 0.40840899
Эпоха 300 MSE = 0.33405894
Эпоха 400 MSE = 0.27571327
Эпоха 500 MSE = 0.22989884
Эпоха 600 MSE = 0.19393113
Эпоха 700 MSE = 0.16569566
Эпоха 800 MSE = 0.14352492
Эпоха 900 MSE = 0.12611014
Эпоха 1000 MSE = 0.11244339
Эпоха 1100 MSE = 0.10170949
Эпоха 1200 MSE = 0.09328186
Эпоха 1300 MSE = 0.08666434
Эпоха 1400 MSE = 0.08147367
Эпоха 1500 MSE = 0.07739663
Эпоха 1600 MSE = 0.07419554
Эпоха 1700 MSE = 0.07168305
Эпоха 1800 MSE = 0.06971153
Эпоха 1900 MSE = 0.06816300
Обучение завершено. Финальная MSE = 0.06695671

```

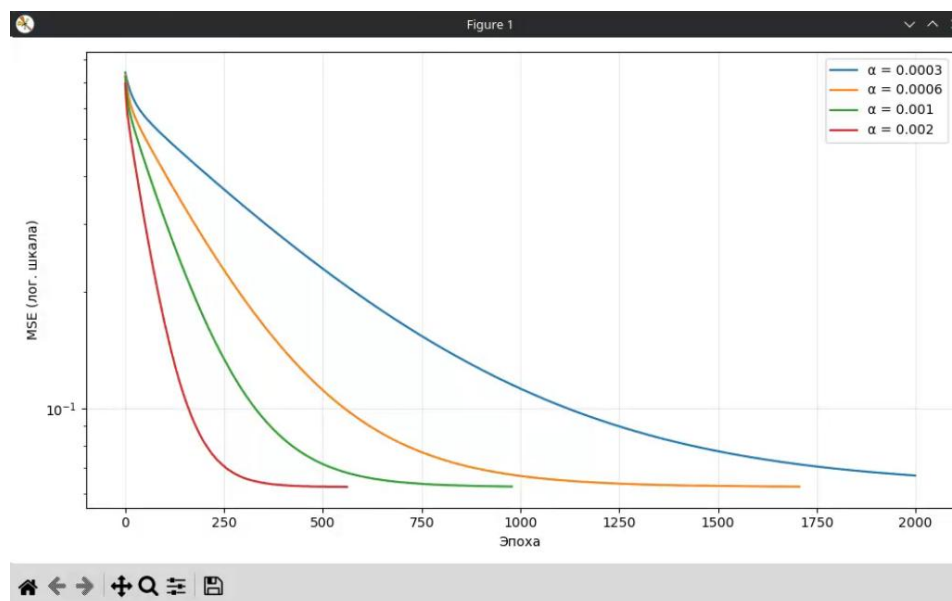
```

Обучение с  $\alpha = 0.000600$  начато...
Эпоха 100 MSE = 0.40629651
Эпоха 200 MSE = 0.27361711
Эпоха 300 MSE = 0.19216679
Эпоха 400 MSE = 0.14211640
Эпоха 500 MSE = 0.11139413
Эпоха 600 MSE = 0.09253110
Эпоха 700 MSE = 0.08094674
Эпоха 800 MSE = 0.07383325
Эпоха 900 MSE = 0.06946237
Эпоха 1000 MSE = 0.06677430
Эпоха 1100 MSE = 0.06512506
Эпоха 1200 MSE = 0.06411294
Эпоха 1300 MSE = 0.06348889
Эпоха 1400 MSE = 0.06311164
Эпоха 1500 MSE = 0.06287425
Эпоха 1600 MSE = 0.06273222
Эпоха 1700 MSE = 0.06264186
Ранняя остановка на эпохе 1706
Обучение завершено. Финальная MSE = 0.06263923

```

```
Обучение с  $\alpha = 0.001000$  начато...
Эпоха 100 MSE = 0.30782815
Эпоха 200 MSE = 0.17044208
Эпоха 300 MSE = 0.10996073
Эпоха 400 MSE = 0.08336301
Эпоха 500 MSE = 0.07168353
Эпоха 600 MSE = 0.06653241
Эпоха 700 MSE = 0.06427639
Эпоха 800 MSE = 0.06328583
Эпоха 900 MSE = 0.06284910
Ранняя остановка на эпохе 978
Обучение завершено. Финальная MSE = 0.06268546
```

```
Обучение с  $\alpha = 0.002000$  начато...
Эпоха 100 MSE = 0.16540012
Эпоха 200 MSE = 0.08149814
Эпоха 300 MSE = 0.06602596
Эпоха 400 MSE = 0.06316608
Эпоха 500 MSE = 0.06262137
Ранняя остановка на эпохе 561
Обучение завершено. Финальная MSE = 0.06254818
```



Выбрана модель с $\alpha = 0.002000$ (лучшая MSE)

Введите точки (x1 x2) или exit для завершения

x1 x2 →

```
UNI: python — Konsole
New Tab Split View Copy Paste Find...
Ранняя остановка на эпохе 1596
Обучение завершено. Финальная MSE = 0.06273722

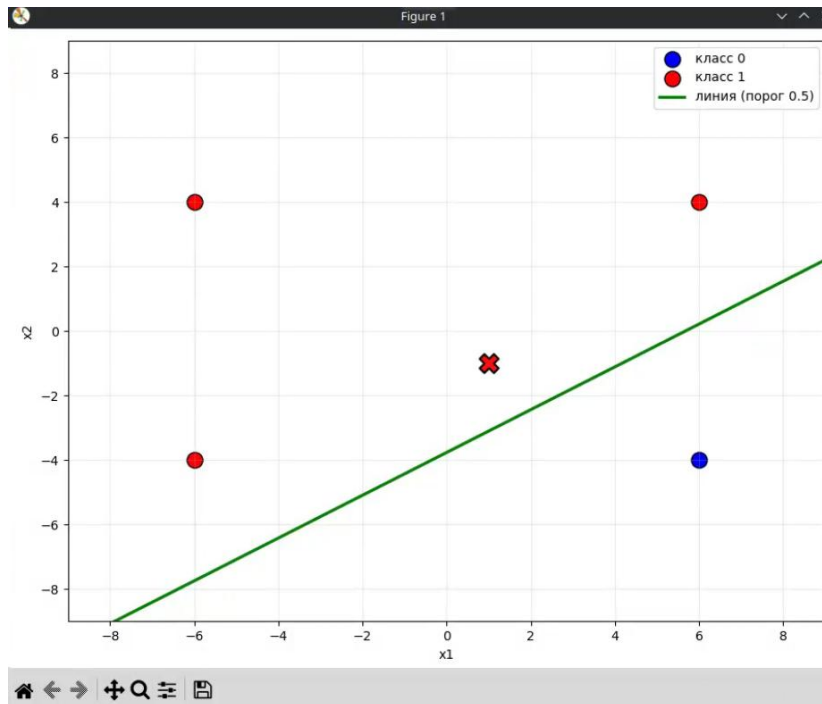
Обучение с  $\alpha = 0.001000$  начато...
Эпоха 100 MSE = 0.30802649
Эпоха 200 MSE = 0.17049222
Эпоха 300 MSE = 0.10999735
Эпоха 400 MSE = 0.08339791
Эпоха 500 MSE = 0.07171083
Эпоха 600 MSE = 0.06654243
Эпоха 700 MSE = 0.06428239
Эпоха 800 MSE = 0.06330585
Эпоха 900 MSE = 0.06284336
Ранняя остановка на эпохе 920
Обучение завершено. Финальная MSE = 0.06279719

Обучение с  $\alpha = 0.002000$  начато...
Эпоха 100 MSE = 0.16533095
Эпоха 200 MSE = 0.08156199
Эпоха 300 MSE = 0.06607019
Ранняя остановка на эпохе 322
Обучение завершено. Финальная MSE = 0.06499487

Выбрана модель с  $\alpha = 0.000600$  (лучшая MSE)

Введите точки (x1 x2) или exit для завершения
x1 x2 → 1 -1
```

x1 x2 → 1 -1
S = 0.630920
Класс = 1 (красный)



Вывод: Были изучены принципы бинарной классификации и реализована однослойная нейронную сеть (персептрон) для решения задачи классификации с использованием **пороговой** функции активации, а также исследован процесс обучения модели с применением среднеквадратичной ошибки (MSE). Было использовано последовательное(online) правило обучения Видроу-Хоффа что позволяет обучать сеть быстрее группового(batch) обучения. График зависимости MSE от количества эпох показывает что при скорости обучения 0.002 обучение достигается быстрее всего за 561 эпоху и показывает то что при понижении скорости обучения количество эпох для обучения кратно возрастает и при этом показатель MSE не улучшается.