Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине «Модели решения задач в интеллектуальных системах»

Тема: «Адаптивный шаг обучения в однослойном линейном персептроне (метод наискорейшего спуска)»

Выполнил:

Студент 3 курса

Группы ИИ-26

Кушнеревич Е.А

Проверила:

Андренко К.В.

Брест 2026

Цель работы: Изучить алгоритм оптимизации градиентного спуска с использованием адаптивного шага обучения. Реализовать модифицированный персептрон, в котором параметр скорости обучения $\alpha(t)$ вычисляется на основе минимизации квадратичной формы ошибки для каждой итерации. Сравнить скорость сходимости с классическим алгоритмом из Лабораторной работы №1. Вариант сохраняется.

Ход работы:

1. Модифицировать алгоритм последовательного обучения (из Лаб №1) таким образом, чтобы на каждой итерации t значение α вычислялось автоматически на основе текущего входного вектора x по формул (см раздел 2.9).
2. Применить вычисленный α(t) для обновления весов ω_{ij} и порогов T_j согласно дельта-правилу.
3. Используя данные своего варианта, провести два эксперимента:
   - Обучение с фиксированным шагом (например, α=0.1 или $\alpha = \frac{1}{p}$).
   - Обучение с адаптивным шагом по Теореме 2.1 (формула 2.36).

Критерий остановки в обоих случаях – достижение заданной суммарной ошибки $E_s \leq E_e$.

4. Построить графики обучения E_s(p), где p – номер эпохи, для обоих экспериментов на одних осях координат.
5. Выполнить графическую визуализацию разделяющей линии для адаптивного метода.
6. Реализовать режим функционирования сети:
   - пользователь задаёт произвольный входной вектор,
   - сеть вычисляет выходной класс,
   - соответствующая точка отображается на графике,

Написать вывод по выполненной работе. Оценить, насколько адаптивный шаг сокращает количество эпох обучения по сравнению с фиксированным. Обязательно сравнение результатов 1 и 2 лабораторных работ.

8.

x_1	x_2	e
5	6	0
-5	6	1
5	-6	1
-5	-6	1

Код программы:

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

x = np.array([
    [ 5,  6],
    [-5,  6],
    [ 5, -6],
    [-5, -6]
])
etalon_value = np.array([0, 1, 1, 1]).reshape(-1, 1)
```

```
def linear_output(W, X, T):
    return X @ W - T

def fixed_fit(X, y, alpha=0.01, Ee=0.5, max_epochs=20000):
    N, n = X.shape
    W = np.zeros((n, 1))
    T = 0.0
    Es_hist = []
    stopped_epoch = max_epochs

    print(f"[Фиксированный] Старт: α={alpha}, Ee={Ee}")

    for ep in range(max_epochs):
        order = np.random.permutation(N)
        for idx in order:
            xi = X[idx:idx+1]
            ti = float(y[idx, 0])

            s  = linear_output(W, xi, T)
            d  = float(s[0, 0]) - ti
            W -= alpha * xi.T * d
            T += alpha * d

        preds = linear_output(W, X, T)
        Es = float(np.sum((preds - y) ** 2))
        Es_hist.append(Es)

        if ep % 50 == 0 and ep > 0:
            print(f"  эпоха {ep:5d} | Es = {Es:.6f}")

        if Es <= Ee:
            stopped_epoch = ep + 1
            print(f"Остановка на эпохе {stopped_epoch}, Es={Es:.6f}\n")
            break
    else:
        print(f"Лимит {max_epochs} эпох, Es={Es_hist[-1]:.6f}\n")

    return W, T, Es_hist, stopped_epoch


def adaptive_fit(X, y, Ee=0.5, max_epochs=20000):
    N, n = X.shape
    W = np.zeros((n, 1))
    T = 0.0
    Es_hist = []
    stopped_epoch = max_epochs

    print(f"(Адаптивный) Старт: α(t), Ee={Ee}")
```

```
    for ep in range(max_epochs):
        order = np.random.permutation(N)
        for idx in order:
            xi = X[idx:idx+1]
            ti = float(y[idx, 0])

            norm_sq = float(np.sum(xi ** 2))
            alpha_t = 1.0 / norm_sq if norm_sq > 1e-12 else 0.0

            s  = linear_output(W, xi, T)
            d  = float(s[0, 0]) - ti

            W -= alpha_t * xi.T * d
            T += alpha_t * d

        preds = linear_output(W, X, T)
        Es = float(np.sum((preds - y) ** 2))
        Es_hist.append(Es)

        if ep % 10 == 0 and ep > 0:
            print(f"  эпоха {ep:5d} | Es = {Es:.6f}")

        if Es <= Ee:
            stopped_epoch = ep + 1
            print(f"Остановка на эпохе {stopped_epoch}, Es={Es:.6f}\n")
            break
    else:
        print(f"Лимит {max_epochs} эпох, Es={Es_hist[-1]:.6f}\n")

    return W, T, Es_hist, stopped_epoch

np.random.seed(42)
Ee = 0.5

W_fix,  T_fix,  es_fix,  ep_fix  = fixed_fit   (x, etalon_value, alpha=0.01,
Ee=Ee)
np.random.seed(42)
W_adap, T_adap, es_adap, ep_adap = adaptive_fit(x,
etalon_value,               Ee=Ee)

print(f"Ускорение: {ep_fix}/{ep_adap} = {ep_fix/ep_adap:.1f}")

fig, axes = plt.subplots(1, 2, figsize=(14, 6))
ax1 = axes[0]

ax1.plot(np.arange(1, len(es_fix)+1),  es_fix,  lw=2,
         color="#2563EB", label=f"Фиксированный α=0.01 (эпох: {ep_fix})")
ax1.plot(np.arange(1, len(es_adap)+1), es_adap, lw=2,
```

```
         color="#DC2626", label=f"Адаптивный α(t)(эпох: {ep_adap})")
ax1.axhline(Ee, color="gray", lw=1.4, ls="--", label=f"Eₑ = {Ee}")
ax1.set_xlabel("Эпоха (p)", fontsize=12)
ax1.set_ylabel("Es(p)", fontsize=12)
ax1.set_title("График обучения Es(p)", fontsize=13, fontweight='bold')
ax1.set_yscale("log")
ax1.legend(fontsize=10)
ax1.grid(alpha=0.35)

ax2 = axes[1]

cls0_mask = etalon_value[:, 0] == 0
cls1_mask = etalon_value[:, 0] == 1

ax2.scatter(x[cls0_mask][:,0], x[cls0_mask][:,1],
            s=200, color="#3B82F6", edgecolors="black", lw=1.5,
            label="Класс 0 (цель=0)", zorder=5)
ax2.scatter(x[cls1_mask][:,0], x[cls1_mask][:,1],
            s=200, color="#EF4444", edgecolors="black", lw=1.5,
            label="Класс 1 (цель=1)", zorder=5)

for pt, lbl in zip(x, ["(5,6)","(-5,6)","(5,-6)","(-5,-6)"]):
    ax2.annotate(lbl, xy=(pt[0], pt[1]),
                 xytext=(pt[0]+0.3, pt[1]+0.6), fontsize=9)

w1, w2 = W_adap.flatten()
xs_line = np.linspace(-9, 9, 500)
if abs(w2) > 1e-9:
    ys_line = (T_adap + 0.5 - w1 * xs_line) / w2
    ax2.plot(xs_line, ys_line, color="#16A34A", lw=2.5,
             label="Разделяющая линия")

ax2.set_xlim(-9, 9); ax2.set_ylim(-9, 9)
ax2.set_xlabel("x₁", fontsize=12); ax2.set_ylabel("x₂", fontsize=12)
ax2.set_title("Разделяющая граница (адаптивный)", fontsize=13,
fontweight='bold')
ax2.legend(fontsize=10); ax2.grid(alpha=0.3)
ax2.axhline(0, color='black', lw=0.5); ax2.axvline(0, color='black', lw=0.5)

plt.tight_layout()
plt.show()

added_pts, added_cls = [], []

print("−" * 45)
print(" Режим функционирования (модель: адаптивный)")
print(" Введите координаты x1 x2, или 'exit'\n")

while True:
```

```
    user_input = input(" -> ").strip()
    if user_input in ("exit", ""):
        break
    try:
        x1_val, x2_val = map(float, user_input.split())
        point = np.array([[x1_val, x2_val]])

        score = linear_output(W_adap, point, T_adap).item()
        cls   = 1 if score >= 0.5 else 0

        print(f"  S = {score:+.6f}")
        print(f"  Класс = {cls} ({'красный' if cls else 'синий'})\n")

        added_pts.append([x1_val, x2_val])
        added_cls.append(cls)

        fig, ax = plt.subplots(figsize=(8, 8))
        ax.scatter(x[cls0_mask][:,0], x[cls0_mask][:,1],
                   s=200, color="#3B82F6", edgecolors="black", label="Класс
0")
        ax.scatter(x[cls1_mask][:,0], x[cls1_mask][:,1],
                   s=200, color="#EF4444", edgecolors="black", label="Класс
1")

        if abs(w2) > 1e-9:
            ys_l = (T_adap + 0.5 - w1 * xs_line) / w2
            ax.plot(xs_line, ys_l, color="#16A34A", lw=2.5, label="Граница")

        for p, c in zip(added_pts, added_cls):
            ax.scatter(p[0], p[1], marker="X", s=280,
                       color="#EF4444" if c else "#3B82F6",
                       edgecolors="black", zorder=6)

        ax.set_xlim(-9, 9); ax.set_ylim(-9, 9)
        ax.set_xlabel("x1"); ax.set_ylabel("x2")
        ax.set_title(f"Последняя точка: ({x1_val}, {x2_val}) → класс {cls}")
        ax.legend(); ax.grid(alpha=0.3)
        plt.tight_layout(); plt.show()

    except Exception:
        print("  Ошибка ввода. Формат: x1 x2\n")

print("Работа завершена.")
```

Результат тестирования:

[Фиксированный] Старт: α=0.01, Ee=0.5

Остановка на эпохе 24, Es=0.499973

(Адаптивный) Старт: α(t), Ee=0.5

эпоха 10 | Es = 0.942490

эпоха 20 | Es = 0.457671

Остановка на эпохе 21, Es=0.457671

Ускорение: 24/21 = 1.1

Режим функционирования (модель: адаптивный)

Введите координаты x1 x2, или 'exit'

-> -5 6

S = +0.917733

Класс = 1 (красный)

График обучения $E_s(p)$:

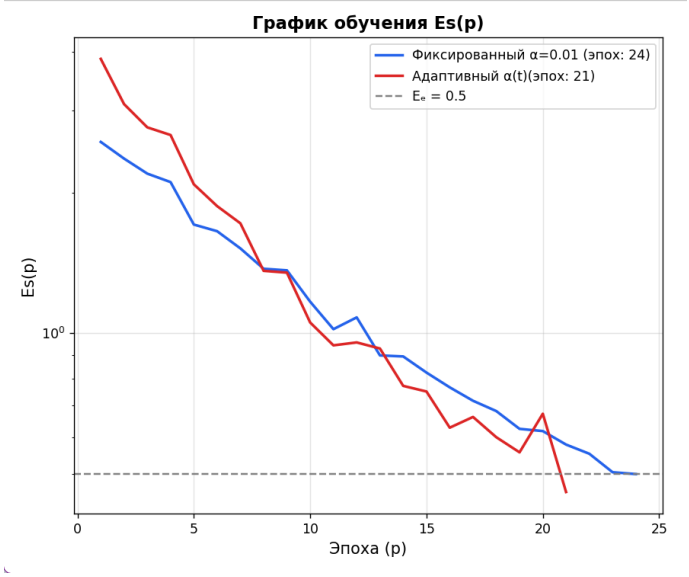
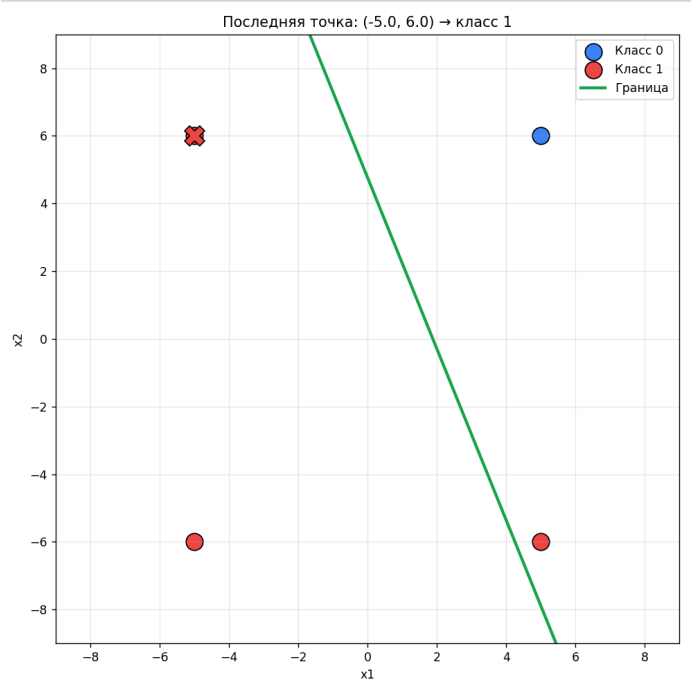


График разделяющей линии для адаптивного метода:



Вывод: проделав данную лабораторную работу я изучил алгоритм оптимизации градиентного спуска с использованием адаптивного шага обучения. Реализовал модифицированный персептрон, в котором параметр скорости обучения $\alpha(t)$ вычисляется на основе минимизации квадратичной формы ошибки для каждой итерации. Сравнил скорость сходимости с классическим алгоритмом из Лабораторной работы №1. Вариант сохраняется.