Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

За шестой семестр

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(1)
Вирко Е.Д.
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 t вычисляется на основе минимизации квадратичной формы ошибки для каждой итерации. Сравнить скорость сходимости с классическим алгоритмом из Лабораторной работы №1. Вариант сохраняется.

Задачи лабораторной 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 ij и порогов Tj согласно дельтаправилу.
3. Используя данные своего варианта, провести два эксперимента:

- Обучение с фиксированным шагом (например, α=0.1 или α=1p).
- Обучение с адаптивным шагом по Теореме 2.1 (формула 2.36).

Критерий остановки в обоих случаях – достижение заданной суммарной ошибки Es≤Ee.

4. Построить графики обучения Es(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,

7. 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.

Вариант 1

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

Код программы:

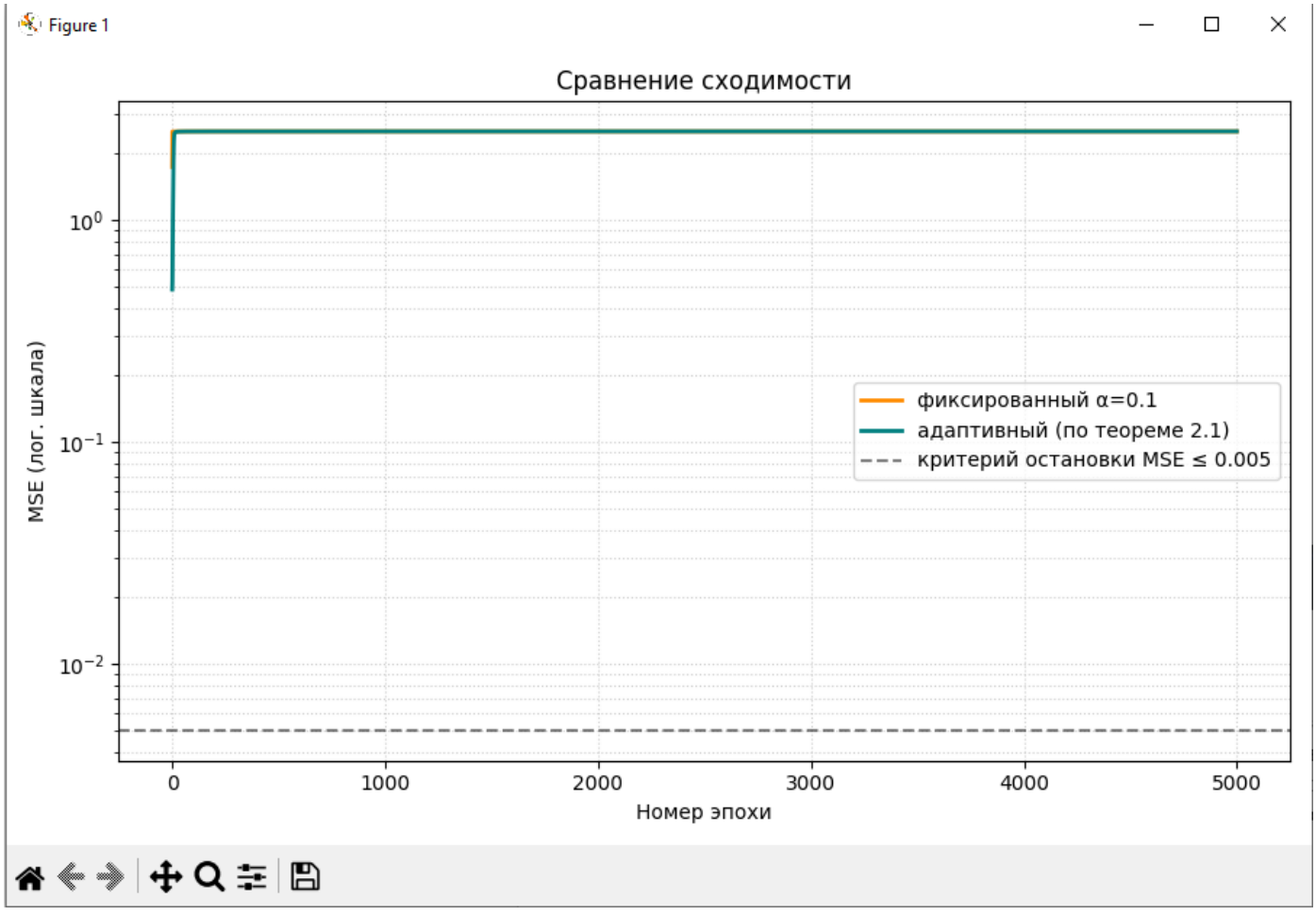
```
import numpy as np
import matplotlib.pyplot as plt

class AdaptivePerceptron:
  def __init__(self, input_size=2):
    self.w = np.random.uniform(-0.5, 0.5, input_size + 1)
    self.X = np.array([])
    self.target = np.array([])

  def set_data(self, X, y):
    X = np.asarray(X)
    if X.ndim == 1:
      X = X.reshape(1, -1)
    self.X = np.insert(X, 0, -1, axis=1)
    self.target = np.asarray(y)
    if len(self.target) != len(self.X):
      raise ValueError("Размерности X и y не совпадают")

  def activate(self, net):
    return np.tanh(net)

  def d_activate(self, out):
    return 1 - out**2

  def forward(self, X=None):
    if X is None:
      X = self.X
    else:
      X = np.insert(np.asarray(X), 0, -1, axis=1) if X.ndim == 2 else np.insert(np.asarray([X]), 0, -1)
    return self.activate(np.dot(X, self.w))

  def train_fixed(self, alpha=0.1, max_epochs=5000, target_mse=0.005):
    history = []
    for epoch in range(max_epochs):
      y_pred = self.forward()
      error = y_pred - self.target
      mse = np.mean(error**2)
      history.append(mse)

      if mse <= target_mse:
        print(f"Фиксированный α={alpha}: достигнута ошибка {mse:.6f} на эпохе {epoch+1}")
        break

      delta = alpha * error * self.d_activate(y_pred)
      self.w += np.dot(delta, self.X)

    else:
      print(f"Фиксированный α={alpha}: max эпох достигнут, MSE={mse:.6f}")
    return history

  def train_adaptive(self, max_epochs=5000, target_mse=0.005):
    history = []
    for epoch in range(max_epochs):
      y_pred = self.forward()
      error = y_pred - self.target
      mse = np.mean(error**2)
      history.append(mse)

      if mse <= target_mse:
        print(f"Адаптивный: достигнута ошибка {mse:.6f} на эпохе {epoch+1}")
        break

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

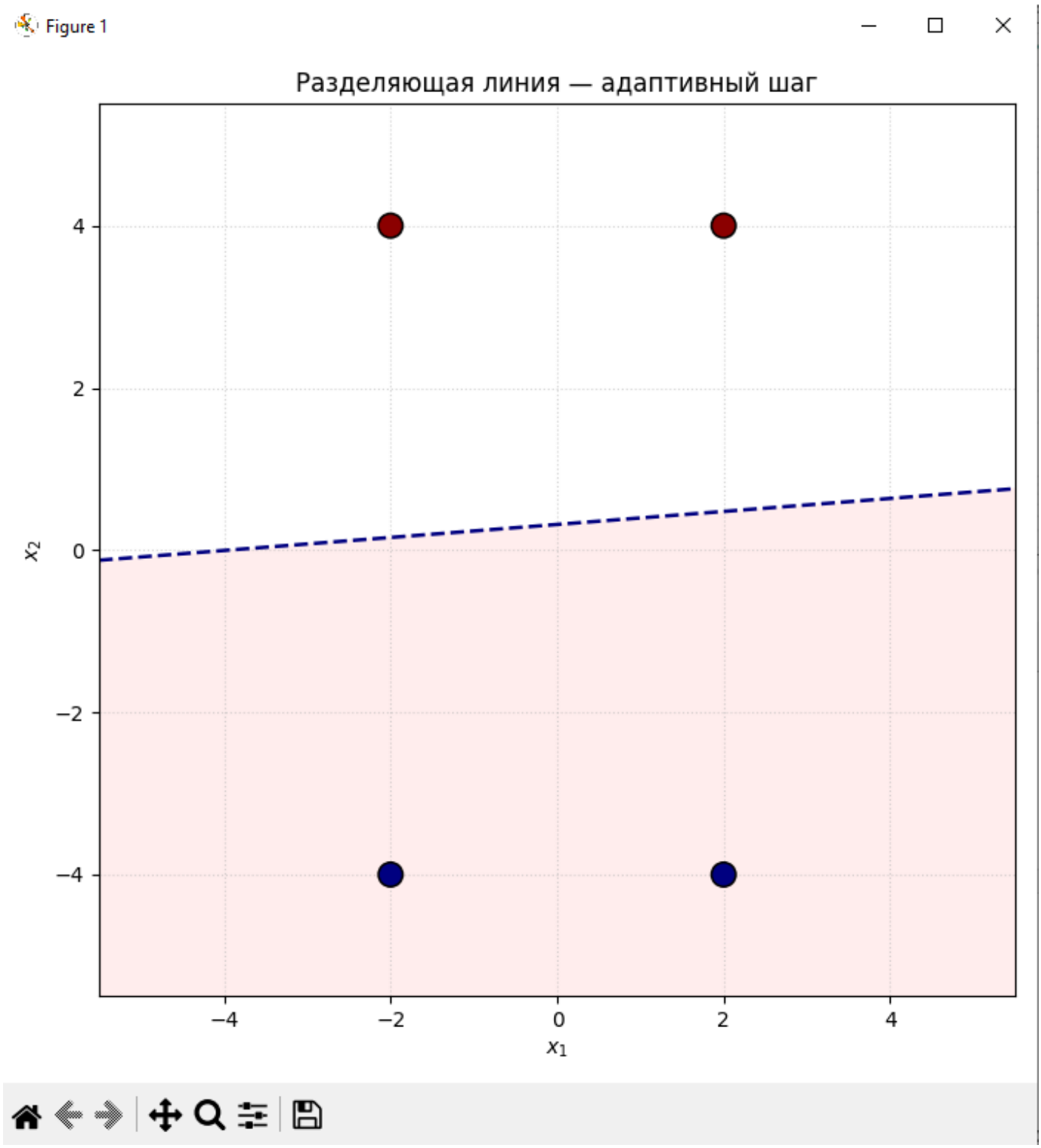
```
            x_i = self.X[i]
            y_i_pred = y_pred[i]
            e_i = error[i]
            norm_sq = np.dot(x_i, x_i)
            alpha_t = 1.0 / (1.0 + norm_sq) if norm_sq > 0 else 1.0

            delta_i = alpha_t * e_i * self.d_activate(y_i_pred)
            self.w += delta_i * x_i

        else:
            print(f"Адаптивный: max эпох достигнут, MSE={mse:.6f}")
        return history

    def plot_separator(self, title="Разделяющая линия (адаптивный)"):
        fig, ax = plt.subplots(figsize=(7, 7))
        xx, yy = np.meshgrid(np.linspace(-5.5, 5.5, 300), np.linspace(-5.5, 5.5, 300))
        grid_points = np.c_[xx.ravel(), yy.ravel()]
        Z = self.forward(grid_points).reshape(xx.shape)

        ax.contourf(xx, yy, Z, levels=[-1, 0, 1], colors=["#ffcccc", "white", "#cce5ff"], alpha=0.35)
        ax.contour(xx, yy, Z, levels=[0], colors='navy', linewidths=1.8, linestyles='--')

        colors = ['darkred' if t == 0 else 'navy' for t in self.target]
        ax.scatter(self.X[:,1], self.X[:,2], c=colors, s=140, edgecolors='black', linewidth=1.1, zorder=5)

        ax.set_xlim(-5.5, 5.5)
        ax.set_ylim(-5.5, 5.5)
        ax.set_xlabel('$x_1$')
        ax.set_ylabel('$x_2$')
        ax.set_title(title)
        ax.grid(True, ls=':', alpha=0.5)
        plt.tight_layout()
        plt.show()


def compare_convergence(hist_fixed, hist_adapt, target_mse=0.005):
    plt.figure(figsize=(9, 5.5))
    plt.semilogy(hist_fixed, label=f"фиксированный α=0.1", lw=2, color="darkorange")
    plt.semilogy(hist_adapt, label="адаптивный (по теореме 2.1)", lw=2, color="teal")

    plt.axhline(target_mse, color="black", ls="--", alpha=0.5, label=f"критерий остановки MSE ≤ {target_mse}")

    plt.title("Сравнение сходимости")
    plt.xlabel("Номер эпохи")
    plt.ylabel("MSE (лог. шкала)")
    plt.legend()
    plt.grid(True, which="both", ls=":", alpha=0.6)
    plt.tight_layout()
    plt.show()



X_train = np.array([[ 2,  4], [-2,  4], [ 2, -4], [-2, -4]])
Y_train = np.array([0, 0, 1, 1])


p_fixed = AdaptivePerceptron()
p_fixed.set_data(X_train, Y_train)
hist_fixed = p_fixed.train_fixed(alpha=0.1, target_mse=0.005)


p_adapt = AdaptivePerceptron()
p_adapt.set_data(X_train, Y_train)
hist_adapt = p_adapt.train_adaptive(target_mse=0.005)
```

```
compare_convergence(hist_fixed, hist_adapt)


p_adapt.plot_separator("Разделяющая линия — адаптивный шаг")


print("\nИнтерактивный режим (используется адаптивная сеть)")
print("Введите два числа: x1 x2  или 'exit' / 'q' для выхода\n")

while True:
  inp = input("→ ").strip()
  if inp.lower() in ['exit', 'q', 'выход']:
    break
  try:
    x1, x2 = map(float, inp.split())
    pred = p_adapt.forward([x1, x2])[0]
    cls = 1 if pred > 0 else 0
    print(f"  выход = {pred: .4f} → класс {cls}  (0 — тёмно-красный, 1 — тёмно-синий)")
  except:
    print("Ошибка ввода. Пример:  1.2  -3.4")
```

Результат:

```
C:\Users\User\PycharmProjects\PythonProject2\.venv\Scripts\python.exe C:\Users\User\PycharmProjects\PythonProject2\.venv\laba2.py
Фиксированный α=0.1: max эпох достигнут, MSE=2.499985
Адаптивный: max эпох достигнут, MSE=2.499969
```





Вывод: изучил алгоритм оптимизации градиентного спуска с использованием адаптивного шага обучения. Реализовал модифицированный персептрон, в котором параметр скорости обучения t вычисляется на основе минимизации квадратичной формы ошибки для каждой итерации. Сравнил скорость сходимости с классическим алгоритмом из Лабораторной работы №1.