Выполнил:

студент группы УВП-412

Рогов К.Д.

**Задание 2-b**

**Задача**

1. Имеем экспериментальные данные  xi = 1,2...,20    yi = xi .    1). построить аппроксимирующую прямую  h(x) = theta0 + theta1 x , считая  theta0 = 0,  для разных значений  theta1               2) Построить зависимость функционала ошибки               J(theta1) =  1/(2m)\*sum(h(xi)-yi)^2  от b и из графика найти theta1=theta1min, соответствующее минимуму функционала
2. Выполнить предыдущее задание для зашумлённых данных (- добавить равномерно распределённые случайные данные в диапазоне (-2, 2))

**Решение**

Решение в виде программного кода реализует визуализацию аппроксимирующих линий и анализ функционала ошибки для линейной регрессии, которая используется для нахождения оптимального коэффициента (θ1​) для данных.

Основные функции кода

1. plot\_approximating\_lines:
   * Строит линии аппроксимации для заданных значений θ1​ для чистых и зашумленных данных.
   * На каждой итерации для каждого значения θ1​ вычисляется линейный результат h(x)=θ1⋅x.
   * Для первого значения θ1​ также вычисляются линии, представляющие tmin⋅x, где tmin – это минимум, найденный из функционала ошибки.
   * График отображает агрегированные данные: чистые данные (зеленые точки) и зашумленные данные (красные точки).
2. plot\_error\_function:
   * Вычисляет сумму квадратов ошибок J(θ1) для каждого значения θ1​.
   * Ошибка рассчитывается как J(θ1)=∑(θ1⋅x−y)^2, где *y* - это чистые или зашумленные данные.
   * График отображает зависимость функции ошибки от значений θ1​ и помогает визуализировать, какое значение наклона минимизирует ошибку.

После выполнения кода:

* Найдено минимальное значение θ1​ для чистых и зашумленных данных.
* Построены графики функционала ошибки.
* Построены линии аппроксимации.

import numpy as np  
import matplotlib.pyplot as plt  
  
  
def plot\_approximating\_lines(x, y\_clean, y\_noisy, theta1\_values, title, filename,  
 tmin\_c, tmin\_n):  
 plt.figure(figsize=(10, 6))  
  
 for theta1 in theta1\_values:  
 if theta1 == theta1\_values[0]:  
 h\_x\_clean = tmin\_c \* x  
 h\_x\_noisy = tmin\_n \* x  
 else:  
 h\_x\_clean = theta1 \* x  
 h\_x\_noisy = theta1 \* x  
  
 # Прямые для чистых данных (синяя сплошная линия)  
 plt.plot(x, h\_x\_clean, color='blue', alpha=0.7, linewidth=2,  
 label=f'Чистые данные' if theta1 == theta1\_values[0] else "")  
 # Прямые для зашумленных данных (оранжевая пунктирная линия)  
 plt.plot(x, h\_x\_noisy, color='orange', linestyle='--', alpha=0.7, linewidth=2, label=f'Зашумленные данные' if theta1 == theta1\_values[0] else "")  
  
 plt.scatter(x, y\_clean, color='green', label='Чистые данные')  
 plt.scatter(x, y\_noisy, color='red', label='Зашумленные данные')  
  
 plt.xlim(0, 21)  
 plt.xticks(np.arange(0, 22, 1))  
 plt.ylim(0, max(np.concatenate((y\_clean, y\_noisy))) \* max(theta1\_values) + 1)  
 plt.title(title)  
 plt.xlabel("x")  
 plt.ylabel("h(x) = theta1 \* x")  
 plt.grid(True)  
 plt.legend()  
 plt.savefig(filename)  
  
  
def plot\_error\_function(x, y\_clean, y\_noisy, theta1\_values, title, filename):  
 J\_theta1\_clean = [np.sum((theta1 \* x - y\_clean) \*\* 2) for theta1 in theta1\_values]  
 J\_theta1\_noisy = [np.sum((theta1 \* x - y\_noisy) \*\* 2) for theta1 in theta1\_values]  
  
 theta1\_min\_clean = theta1\_values[np.argmin(J\_theta1\_clean)]  
 theta1\_min\_noisy = theta1\_values[np.argmin(J\_theta1\_noisy)]  
  
 plt.figure()  
 plt.plot(theta1\_values, J\_theta1\_clean, label='J(theta1) Чистые данные', color='b')  
 plt.plot(theta1\_values, J\_theta1\_noisy, label='J(theta1) Зашумленные данные', color='orange')  
 plt.scatter(theta1\_min\_clean, min(J\_theta1\_clean), color='blue')  
 plt.scatter(theta1\_min\_noisy, min(J\_theta1\_noisy), color='red')  
 plt.ylim([0, 250])  
 plt.xlim([0.5, 1.5])  
 plt.title(title)  
 plt.xlabel('theta1')  
 plt.ylabel('J(theta1)')  
 plt.grid(True)  
 plt.legend()  
 plt.savefig(filename)  
  
 return theta1\_min\_clean, theta1\_min\_noisy  
  
  
x = np.arange(1, 21)  
y\_clean = x  
y\_noisy = x + np.random.uniform(-2, 2, x.shape)  
  
theta1\_values\_line = np.linspace(0, 2, 5)  
theta1\_values\_error = np.linspace(0, 2, 120)  
  
# Для ошибки  
theta1\_min\_clean, theta1\_min\_noisy = plot\_error\_function(x, y\_clean, y\_noisy, theta1\_values\_error,  
 "Функционал ошибки для чистых и зашумленных данных",  
 'error.png')  
  
# Для прямых  
plot\_approximating\_lines(x, y\_clean, y\_noisy, theta1\_values\_line,  
 "Аппроксимирующие прямые для чистых и зашумленных данных", 'lines.png', theta1\_min\_clean, theta1\_min\_noisy)  
  
  
  
print(f'Минимум theta1 (Чистые данные): {theta1\_min\_clean}')  
print(f'Минимум theta1 (Зашумленные данные): {theta1\_min\_noisy}')

**Результаты работы**



