Assignment 6: Gradient Descent for Linear Regression

Objective:
Learn how to fit a linear regression model ug gradient descent.

Instructions:
This assignment engages you in the practical application of gradient descent for linear regression modeling. Follow the steps carefully, and submit your work as a gle PDF file that includes all required code and graphs. Attach supplementary code files (.py and .ipynb) as used in this assignment.

Step-by-Step Instructions

Question 1: Linear Regression Model ug Maximum Likelihood Estimation

1.1 Plotting Training and Testing Data

Base this task on Assignment 4, Question 1.

```
# =============================================================

import numpy as np
import matplotlib.pyplot as plt

# Data Initialization
X = np.array([-4.5, -3.5, -3, -1.8, -0.2, 0.3, 1.3, 2.6, 3.8, 4.8]).reshape(-1,1)
y = np.array([
    [-0.91650116],
    [-0.47546053],
    [-0.10972425],
    [0.29504095],
    [-0.01596218],
    [0.10014949],
    [0.48104303],
    [0.10979023],
    [-0.99742128],
    [-0.91221826]
]).reshape(-1,1)

X_test = np.array([-3.99, -1.38, -1.37, -0.94,
  0.69,  1.4,   1.57,  1.78,  1.81,  4.89]).reshape(-1,1)
y_test = np.array([
    [-0.80737607],
    [0.19813376],
    [0.19537639],
    [0.07185977],
    [0.24954213],
    [0.50662504],
    [0.52943298],
    [0.52406997],
    [0.51999057],
    [-0.82318288]
]).reshape(-1,1)

#=================================================================
```

- **Task:** Plot the relationship between the training data (X, y) and the testing data (X_test, y_test) on the same graph ug distinct colors to differentiate between the two sets.
- **Requirements:**
  - Attach the Python code used to generate the graph in your PDF.
  - Display the graph, ensuring clarity by labeling axes and adding a legend.

1.2 Polynomial Feature Transformation

- **Task:** Implement the poly_features function to transform input data X into polynomial features of degree K.

(Polynomial Regression)

$$\phi(x) = \begin{bmatrix} \phi_0(x) \\ \phi_1(x) \\ \vdots \\ \phi_{K-1}(x) \end{bmatrix} = \begin{bmatrix} 1 \\ x \\ x^2 \\ x^3 \\ \vdots \\ x^{K-1} \end{bmatrix}$$

(Feature Matrix for Second-order Polynomials)

$$\mathbf{\Phi} = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ \vdots & \vdots & \vdots \\ 1 & x_N & x_N^2 \end{bmatrix}$$

- **Requirements:**
  - Attach the Python code used for the implementation.
  - Explain the purpose and effect of this transformation.

1.3 Fitting the Model Ug Maximum Likelihood

- **Task:** Describe the process of fitting a model ug the Maximum Likelihood Estimation (MLE) method. Transform both training and test datasets ug the poly_features function for a polynomial degree of 5.
- **Formula:**

$$\theta_{ML} = (\Phi^T \Phi)^{-1} \Phi^T y$$

- **Requirements:**
    - Attach Python code used, display calculated values of θ_{ML}, and include a graph that shows both data sets along with predictions for all x values from -8 to 8.

1.4 Model Evaluation

- **Task:** Use the Root Mean Square Error (RMSE) to evaluate the model accuracy.
    - Calculate and plot RMSE for polynomial degrees ranging from 0 to 15 for both training and test datasets.
    - Determine and discuss the optimal polynomial degree based on the test set RMSE.
- **Requirements:**
    - Attach Python code and generate the graph in your PDF.

1.5 Selecting the Best Model

- **Task:** Identify and justify the polynomial degree that results in the lowest RMSE for the test dataset.
- **Requirements:**
    - Include a graph that shows both training and test sets, along with predictions for all x values from -8 to 8.
    - Attach Python code used for the implementation and generate the graph in your PDF.

Question 2: Gradient Descent for Polynomial Regression

2.1 Mathematical Derivation

- **Task:** Derive the formula for the gradient of the Sum of square loss function used in polynomial regression. Show step-by-step differentiation of $J(\theta)$ with respect to θ.
- **Formula:**

$$J(\theta) = (\mathrm{y} - \Phi^T\theta)^2$$

2.2 Feature Transformation and Gradient Descent Implementation

- **Task:** Transform both training and test datasets into the e-modified polynomial feature space ug the optimal degree identified in Q1.
    - Code the Gradient Descent algorithm to minimize the MSE. Use a learning rate of 0.0002 and iterate up to 100,000 times. Implement feature normalization to aid in convergence.
    - Implement feature normalization example code:

```
#=====================

def normalize_features(Phi):
    """ Normalize features to have zero mean and unit variance """
    mu = np.mean(Phi, axis=0)
    sigma = np.std(Phi, axis=0)
    Phi_normalized = (Phi - mu) / sigma
    return Phi_normalized, mu, sigma

Phi = poly_features(x_train, optimal_k)
Phi_norm, mu, sigma = normalize_features(Phi)

#=====================
```

    - Discuss any dynamic adjustments to the learning rate during the descent to enhance convergence.

2.3 Model Evaluation and Comparison

- **Task:** Monitor and record the RMSE on the test set at each step of the Gradient Descent. Compare this RMSE with that obtained ug the MLE method from Q1.

- **Requirements:**
  - Include complete Python code that carries out feature transformation, gradient descent optimization, RMSE calculation, and plotting.
  - Attach plots demonstrating cost reduction over iterations and the final model fit against the test data.
  - Discuss the implementation, convergence behavior, and effectiveness compared to MLE.

Bonus: Earn an additional 10 points, not included in the standard 100 points. You will receive these points if the RMSE of the test set, using the Theta parameters from Question 2, is lower than the RMSE obtained using the MLE method from Question 1.

Submission Requirements:

- Submit your completed assignment as a PDF containing all necessary code snippets and graphs.
- Attach corresponding .py and .ipynb files containing executable code.
- Ensure all files are clearly labeled and organized.
- Late submissions will be subject to deductions according to the policy.

Evaluation Criteria: Your assignment will be graded based on the accuracy of your implementation, the clarity of your presentation, and the completeness of your submission. Specific attention will be given to how well you follow these guidelines:

- **Step-by-Step Explanation:** This assignment requires clear, step-by-step explanations. You must explicitly label and explain each step (e.g., Step 1, Step 2, Step 3, Step 4). Failure to provide detailed explanations for each required step will result in a proportional deduction in your score. For example, if the assignment requires four steps and you complete only three, this will result in a score of 75 out of 100.