Optimization Approach for Solving Linear Regression

1. Introduction

The optimization approach for linear regression involves finding the coefficients ($\boldsymbol{\theta}$) by minimizing the loss function. This is typically achieved using an iterative algorithm such as **Gradient Descent**. Instead of solving normal equations analytically, this approach uses numerical methods to approximate the optimal solution.

2. Model and Loss Function

Model Equation

The model for linear regression can be written in matrix form as:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\theta}$$

where:

- $\mathbf{y}$: $n \times 1$ vector of observed values,
- $\mathbf{X}$: $n \times (p+1)$ matrix of predictors (first column is ones for θ_0),
- $\boldsymbol{\theta}$: $(p+1) \times 1$ vector of coefficients.

Loss Function

The loss function for linear regression is the **Mean Squared Error (MSE)**:

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \sum_{i=1}^{n} \left(y_i - \mathbf{x}_i^T \boldsymbol{\theta}\right)^2$$

or, equivalently in matrix form:

$$J(\boldsymbol{\theta}) = \frac{1}{2n} \left(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\right)^T \left(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\right)$$

The objective is to minimize $J(\boldsymbol{\theta})$.

3. Gradient Descent Algorithm

Gradient of the Loss Function

The gradient of the loss function with respect to $\boldsymbol{\theta}$ is:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{1}{n} \mathbf{X}^T \left(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\right)$$

Gradient Descent Steps

The gradient descent algorithm updates the coefficients iteratively as follows:

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha \cdot \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

where:

- $\boldsymbol{\theta}^{(k)}$: Coefficients at iteration k,
- α: Learning rate, a hyperparameter controlling step size.

Algorithm Procedure

1. **Initialize** $\boldsymbol{\theta}^{(0)}$ (e.g., zeros or random values).
2. **Compute the gradient:**

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = -\frac{1}{n} \mathbf{X}^T \left(\mathbf{y} - \mathbf{X}\boldsymbol{\theta}\right)$$

3. **Update the coefficients:**

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \alpha \cdot \frac{\partial J(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}}$$

4. **Repeat** until convergence, i.e., when the change in $\boldsymbol{\theta}$ or the loss function $J(\boldsymbol{\theta})$ becomes negligibly small.

4. Convergence and Learning Rate

- A smaller learning rate α ensures convergence but slows down the process.
- A larger learning rate may cause the algorithm to diverge.
- Proper tuning of α and normalization of features improve convergence.

5. Advantages and Applications

- Suitable for large datasets where matrix inversion ($(\mathbf{X}^T\mathbf{X})^{-1}$) is computationally expensive.
- Works well with streaming data or mini-batch updates.
- Flexible and applicable to non-linear models.