Linear Regression Cost Function

1. Hypothesis/Output Function

In linear regression, the hypothesis is assumed to be a linear function of the input:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots + \theta_n x_n$$

In vectorized form:

$$h_\theta(x) = X \cdot \theta$$

where:

- X is the feature matrix of size $m \times (n+1)$,
- θ is the parameter vector of size $(n+1) \times 1$,
- $h_\theta(x)$ is the predicted value for the input x.

Hypothesis for Linear Regression

For a single training example $x^{(i)}$, the hypothesis function is:

$$h_\theta(x^{(i)}) = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_n x_n^{(i)}$$

This can also be written in vector form as:

$$h_\theta(x^{(i)}) = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}^T \cdot \begin{bmatrix} x_0^{(i)} \\ x_1^{(i)} \\ \vdots \\ x_n^{(i)} \end{bmatrix} = \theta^T \cdot x^{(i)}$$

Here:

- $x^{(i)} = \left[x_0^{(i)}, x_1^{(i)}, \ldots, x_n^{(i)}\right]^T$ is the feature vector for the i-th training example (with $x_0 = 1$ for the bias term).
- $\theta = \left[\theta_0, \theta_1, \ldots, \theta_n\right]^T$ is the parameter vector.

The result $h_\theta(x^{(i)})$ is a scalar value that represents the predicted output for the i-th training example.

Hypothesis for All Training Examples

For m training examples, we stack the feature vectors $x^{(1)}, x^{(2)}, \ldots, x^{(m)}$ into a single matrix X, called the **feature matrix**:

$$X = \begin{bmatrix} x_0^{(1)} & x_1^{(1)} & \dots & x_n^{(1)} \\ x_0^{(2)} & x_1^{(2)} & \dots & x_n^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ x_0^{(m)} & x_1^{(m)} & \dots & x_n^{(m)} \end{bmatrix}$$

Here:

- X is an $m \times (n+1)$ matrix:
  - m: Number of training examples (rows).
  - $n+1$: Number of features, including the bias term (columns).
- Each row of X corresponds to a single training example $x^{(i)}$.

For all training examples, the predictions can be computed as:

$$\hat{y} = X \cdot \theta$$

where:

- X is the feature matrix of size $m \times (n+1)$.
- θ is the parameter vector of size $(n+1) \times 1$.
- $\hat{y}$ is the prediction vector of size $m \times 1$, where each element is:

$$\hat{y}_i = h_\theta(x^{(i)}) = \sum_{j=0}^{n} \theta_j x_j^{(i)}$$

2. Error Between Prediction and Actual Value

For each training example i, the error is defined as the difference between the predicted value $h_\theta(x^{(i)})$ and the actual target value $y^{(i)}$:

$$\text{Error}_i = h_\theta(x^{(i)}) - y^{(i)}$$

3. Measuring the Error

To evaluate the performance of the model, we use the sum of squared errors (SSE):

$$\text{SSE} = \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Squaring the errors ensures that:

- Positive and negative errors do not cancel each other out.
- Larger errors are penalized more heavily than smaller errors.

4. Averaging the Error

To make the error independent of the number of training examples m, we calculate the mean squared error (MSE):

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

5. Cost Function Definition

The cost function $J(\theta)$ is defined as half the Mean Squared Error (MSE):

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

The factor of $\frac{1}{2}$ simplifies the gradient computation during optimization, as it cancels the factor of 2 that arises from differentiation.

6. Vectorized Form

In vectorized notation, the cost function can be written as:

$$J(\theta) = \frac{1}{2m} \left((X \cdot \theta - y)^T \cdot (X \cdot \theta - y) \right)$$

where:

- $X \cdot \theta$ computes the predicted values for all training examples,
- $X \cdot \theta - y$ is the vector of residuals (differences between predictions and actual values),
- $(X \cdot \theta - y)^T \cdot (X \cdot \theta - y)$ computes the sum of squared residuals.

7. Final Cost Function

The final cost function is:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Gradient Descent for Linear Regression

Gradient descent is an iterative optimization algorithm used to minimize the cost function $J(\theta)$ in linear regression. The updates to the parameter vector θ are performed by moving in the direction of the negative gradient of $J(\theta)$.

Cost Function

The cost function for linear regression is defined as:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Gradient of the Cost Function

To minimize $J(\theta)$, we compute the gradient of $J(\theta)$ with respect to θ:

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right) x_j^{(i)}$$

In vectorized form, this gradient can be written as:

$$\nabla_\theta J(\theta) = \frac{1}{m} X^T \left(X\theta - y \right)$$

where:

- X: Feature matrix of size $m \times (n+1)$.
- θ: Parameter vector of size $(n+1) \times 1$.
- y: Output vector of size $m \times 1$.
- $X\theta - y$: The vector of residuals (differences between predicted and actual values).

Gradient Descent Update Rule

Using the gradient, we update θ iteratively:

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$

where:

- α: Learning rate, which determines the step size.

Substituting the gradient:

$$\theta := \theta - \frac{\alpha}{m} X^T \left(X\theta - y \right)$$

Step-by-Step Computation (Vectorized Form)

1. Compute the predictions:

$$\hat{y} = X\theta$$

2. Compute the residuals:

$$\text{residuals} = X\theta - y$$

3. Compute the gradient:

$$\nabla_\theta J(\theta) = \frac{1}{m} X^T (X\theta - y)$$

4. Update the parameters:

$$\theta := \theta - \alpha \nabla_\theta J(\theta)$$