Mathematical Implementation of Logistic Rgression

Namonye kenneth	2023/U/MMU/BCS/01668
Karamagi Henry Atuhaire	2023/U/MMU/BCS/00523
Mpagi Derrick Brair	2023/U/MMU/BCS/00101
Beinomugisha wyclif	2023/U/MMU/BCS/01681
Kato Paul	2023/U/MMU/BCS/00104

February 2025

1 Introduction

Logistic Regression: is a statistical and machine learning technique used for binary classification problems, where the goal is to predict one of two possible outcomes (e.g., yes/no, true/false, 0/1). Despite its name, logistic regression is a classification algorithm, not a regression algorithm. It models the probability of a binary outcome using a logistic function.

Here's a step-by-step explanation of the mathematical implementation of logistic regression:

1. **Problem Setup**

- Let X be the input features (independent variables), and y be the binary output (dependent variable) such that $y \in \{0, 1\}$. - The goal is to model the probability $P(y = 1|X)$, i.e., the probability that $y = 1$ given the input features X.

2. **Logistic Function (Sigmoid Function)**

The logistic regression model uses the sigmoid function to map predicted values to probabilities. The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Where: - z is a linear combination of the input features X and model parameters (weights) θ. - $\sigma(z)$ outputs a value between 0 and 1, which can be interpreted as a probability.

3. **Linear Combination of Inputs**

The linear combination z is computed as:

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \ldots + \theta_n x_n$$

Where: - θ_0 is the **bias term** (intercept). - $\theta_1, \theta_2, \ldots, \theta_n$ are the **weights** corresponding to the input features $x_1, x_2, \ldots, x_n$. - n is the number of features.

This can be written in vector form as:

$$z = \theta^T X$$

Where: - $\theta = [\theta_0, \theta_1, \ldots, \theta_n]$ is the weight vector. - $X = [1, x_1, x_2, \ldots, x_n]$ is the feature vector (with 1 added for the bias term).

4. **Probability Prediction**

The probability $P(y = 1|X)$ is modeled using the sigmoid function:

$$P(y = 1|X) = \sigma(z) = \frac{1}{1 + e^{-\theta^T X}}$$

Similarly, the probability $P(y = 0|X)$ is:

$$P(y = 0|X) = 1 - P(y = 1|X) = 1 - \sigma(z)$$

5. **Decision Boundary**

To make predictions, a threshold (usually 0.5) is applied to the predicted probability: - If $P(y = 1|X) \geq 0.5$, predict $y = 1$. - If $P(y = 1|X) < 0.5$, predict $y = 0$.

The decision boundary is the set of points where $P(y = 1|X) = 0.5$, which corresponds to $z = 0$ (i.e., $\theta^T X = 0$)

.

6. **Cost Function (Log Loss)**

To train the logistic regression model, we need a cost function to measure how well the model's predictions match the actual labels. The cost function used in logistic regression is the **log loss** (or binary cross-entropy loss):

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(h_\theta(X^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(X^{(i)})) \right]$$

Where: - m is the number of training examples. - $y^{(i)}$ is the actual label for the i-th example. - $h_\theta(X^{(i)}) = \sigma(\theta^T X^{(i)})$ is the predicted probability for the i-th example.

This cost function penalizes incorrect predictions more heavily, especially when the model is confident but wrong.

7. **Optimization (Gradient Descent)**

The goal is to minimize the cost function $J(\theta)$ by finding the optimal values of θ. This is typically done using **gradient descent**:

1. Initialize the weights θ randomly or to zero. 2. Compute the gradient of the cost function with respect to θ:

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

3. Update the weights iteratively:

$$\theta_j := \theta_j - \alpha \frac{\partial J(\theta)}{\partial \theta_j}$$

Where: - α is the learning rate (a hyperparameter controlling the step size). - The process is repeated until convergence.