Derivation of MLE for Logistic Regression

Derivation of MLE for Logistic Regression

Logistic Regression Model

In logistic regression, we model the probability that a binary response variable Y (taking values 0 or 1) equals 1 given a set of predictors X (which can be a vector). The logistic function is defined as:

$$P(Y = 1 \mid X) = \sigma(\beta^T X) = \frac{1}{1 + e^{-\beta^T X}}$$

where: - $\sigma(z)$ is the logistic function. - β is the vector of coefficients (parameters) we want to estimate. - $\beta^T X$ is the dot product of β and X.

Likelihood Function

For a dataset with n observations (X_i, Y_i) for $i = 1, 2, \ldots, n$, the likelihood function $L(\beta)$ is the product of the probabilities of the observed outcomes:

$$L(\beta) = \prod_{i=1}^{n} P(Y_i \mid X_i) = \prod_{i=1}^{n} \sigma(\beta^T X_i)^{Y_i} (1 - \sigma(\beta^T X_i))^{1-Y_i}$$

Log-Likelihood Function

Since it's easier to work with sums than products, we take the natural logarithm of the likelihood function to get the log-likelihood function:

$$\ell(\beta) = \log L(\beta) = \sum_{i=1}^{n} \left[Y_i \log(\sigma(\beta^T X_i)) + (1 - Y_i) \log(1 - \sigma(\beta^T X_i)) \right]$$

Differentiating the Log-Likelihood

To find the MLE, we need to maximize the log-likelihood function, which involves taking the derivative with respect to β and setting it to zero.

Let's differentiate $\ell(\beta)$:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{n} \left[Y_i \frac{1}{\sigma(\beta^T X_i)} \frac{\partial \sigma(\beta^T X_i)}{\partial \beta} - (1 - Y_i) \frac{1}{1 - \sigma(\beta^T X_i)} \frac{\partial \sigma(\beta^T X_i)}{\partial \beta} \right]$$

We need to calculate $\frac{\partial \sigma(z)}{\partial z}$ where $z = \beta^T X$:

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

Now applying the chain rule:

$$\frac{\partial \sigma(\beta^T X_i)}{\partial \beta} = \frac{\partial \sigma(z)}{\partial z} \cdot \frac{\partial z}{\partial \beta} = \sigma(\beta^T X_i)(1 - \sigma(\beta^T X_i))X_i$$

Putting It All Together

Now we substitute this back into our expression for the derivative of the log-likelihood:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{n} \left[Y_i \frac{1}{\sigma(\beta^T X_i)} \cdot \sigma(\beta^T X_i)(1 - \sigma(\beta^T X_i))X_i - (1 - Y_i)\frac{1}{1 - \sigma(\beta^T X_i)} \cdot \sigma(\beta^T X_i)(1 - \sigma(\beta^T X_i))X_i \right]$$

Simplifying this gives us:

$$\frac{\partial \ell(\beta)}{\partial \beta} = \sum_{i=1}^{n} \left[(Y_i - \sigma(\beta^T X_i))X_i \right]$$

Setting this equal to zero for maximization leads us to the following condition:

$$\sum_{i=1}^{n} (Y_i - \sigma(\beta^T X_i))X_i = 0$$

This equation is used to find the MLE for β.

Conclusion

To summarize, we derived the log-likelihood function for logistic regression and differentiated it with respect to the parameter vector β. The differentiation steps highlighted the use of the chain rule, which is essential in this process.

Logistic Regression Pseudocode

The following pseudocode demonstrates how to estimate the coefficients B_0 and B_1 for a logistic regression model using gradient ascent.

```
# Initialize parameters
B0 <- 0
B1 <- 0
learning_rate <- 0.01
num_iterations <- 1000

# Function to calculate the sigmoid function
sigmoid <- function(z) {
    return(1 / (1 + exp(-z)))
}

# Function to calculate the log-likelihood
log_likelihood <- function(B0, B1, X, Y) {
    N <- length(Y)
    ll <- 0
    for (i in 1:N) {
        predicted_prob <- sigmoid(B0 + B1 * X[i])
        ll <- ll + Y[i] * log(predicted_prob) + (1 - Y[i]) * log(1 - predicted_prob)
```

```r
    }
    return(ll)
}

# Gradient Ascent Algorithm
for (iter in 1:num_iterations) {
    gradient_B0 <- 0
    gradient_B1 <- 0
    N <- length(Y)

    for (i in 1:N) {
        predicted_prob <- sigmoid(B0 + B1 * X[i])
        error <- Y[i] - predicted_prob

        # Calculate gradients
        gradient_B0 <- gradient_B0 + error
        gradient_B1 <- gradient_B1 + error * X[i]
    }

    # Update coefficients
    B0 <- B0 + learning_rate * gradient_B0 / N
    B1 <- B1 + learning_rate * gradient_B1 / N
}

# Output the estimated coefficients
cat("Estimated B0:", B0, "\n")
cat("Estimated B1:", B1, "\n")
```