Binary Logistic Regression

Aaron Avram

June 2 2025

Introduction

In this write up I will go through my derivation of the objective function and the optimization algorithm for my Binary Logistic Regression model.

Construction

For my notation let $X \in \mathbb{R}^{n \times p}$ be the matrix with each of the n training inputs as rows. Where we define p to be the dimension of each input vector (or equivalently the number of features). Next let $y \in \{0,1\}^n$ equal the vector of training labels, where our two output classes are 0 and 1 for convenience.

To start, the general setup for Binary Logistic Regression is to model the probability of each of the two values in the target space as:

$$\Pr(x = 1; \beta, \beta_0) = \frac{\exp(x^T\beta + \beta_0)}{1 + \exp(x^T\beta + \beta_0)}$$

Or equivalently:

$$= \frac{1}{1 + \exp(-(x^T\beta + \beta_0))}$$

There are two convenient stylistic modifications we will make. First, let $\theta \in \mathbb{R}^{p+1}$ be defined as the vector with β as its first p entries and β_0 as its $p+1$th entry. Now let us extend X so that it becomes a $n \times p+1$ matrix where its new column is simply a column of ones. Next, define $\Pr(x;\theta) := \Pr(x = 1;\theta)$ and we will define $\Pr(x = 0;\theta)$ implicitly given by $1 - \Pr(x;\theta)$ as the total probability must be one.

Objective Function

Now we will construct the objective function. I decided to implement my model with a Log-likelihood function as is common for probabilistic models. Thus, the objective function is:

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} \log \Pr(x_i = y_i; \theta)$$

A more useful way to represent it would be:

$$= \sum_{i=1}^{n} y_i \log \Pr(x_i; \theta) + (1 - y_i) \log(1 - \Pr(x_i; \theta))$$

$$= \sum_{i=1}^{n} y_i \log \frac{\exp(x_i^T \theta)}{1 + \exp(x_i^T \theta)} + (1 - y_i) \log \frac{1}{1 + \exp(x_i^T \theta)}$$

$$= \sum_{i=1}^{n} y_i \log \exp(x_i^T \theta) - y_i \log(1 + \exp(x_i^T \theta)) - (1 - y_i) \log(1 + \exp(x_i^T \theta))$$

$$= \sum_{i=1}^{n} y_i x_i^T \theta - \log(1 + \exp(x_i^T \theta))$$

We want to maximize this function with respect to θ so that the likelihood of the correct training label being predicted is as high as possible.

Optimization

Now to maximize the objective function we want to find the value for θ at which the gradient of the objective is zero (Or practically, close to zero). The gradient of $\mathcal{L}$ with respect to θ is:

$$\nabla_\theta \mathcal{L}(\theta) = \nabla_\theta (\sum_{i=1}^{n} y_i x^T \theta - \log(1 + \exp(x_i^T \theta)))$$

$$= \sum_{i=1}^{n} y_i x_i - \frac{x_i \exp(x_i^T \theta)}{1 + \exp(x_i^T \theta)}$$

$$= \sum_{i=1}^{n} x_i (y_i - \Pr(x_i; \theta))$$

If we define $p \in \mathbb{R}^n$ as the column vector of probabilities for each x_i:

$$= X^T (y - p)$$

Now that we have our gradient, we can proceed one of two ways. Either, we use gradient descent to minimize the negative log-likelihood or apply the Newton-Raphson Method to the gradient of the log-likelihood. It becomes abundantly clear that the latter choice is the more computationally efficent method after we look at the specifics.

The Newton-Raphson method describes an iterative method for finding a zero of a function, where the update rule is given by:

$$x^{new} = x^{old} - J(f)(x^{old})^{-1}(f(x^{old}))$$

Where $J(f)(x^{old})^{-1}$ is the Jacobian of f at x^{old}. Applying this to $\nabla_\theta(\mathcal{L})$ as a function of θ we have:

$$\theta^{new} = \theta^{old} - H(\mathcal{L})(\theta^{old})^{-1}(\nabla_\theta(\mathcal{L})(\theta^{old}))$$

Where $H(\mathcal{L})(\theta^{old})$ is the Hessian of $\mathcal{L}$ at θ^{old}. Now we must compute the Hessian, which is the Jacobian of the gradient. I will do this in a couple of steps, first:

$$\begin{aligned}
\frac{\partial p(x_i;\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta}\Big(\frac{\exp(x_i^T\theta)}{1+\exp(x_i^T\theta)}\Big) \\
&= \frac{x_i(\exp(x_i^T\theta))(1+\exp(x_i^T\theta)) - x_i(\exp(x_i^T\theta)^2)}{(1+\exp(x_i^T\theta))^2} \\
&= \Big(\frac{x_i(\exp(x_i^T\theta))}{1+\exp(x_i^T\theta)}\Big)\Big(\frac{1+\exp(x_i^T\theta)-\exp(x_i^T\theta)}{\exp(x_i^T\theta)}\Big) \\
&= x_i \Pr(x_i;\theta)(1-\Pr(x_i;\theta))
\end{aligned}$$

For fixed j:

$$\begin{aligned}
\frac{\partial(\nabla_\theta(\mathcal{L}))}{\partial \theta_j} &= \frac{\partial}{\partial \theta_j}\Big(\sum_{i=1}^{n} x_i(y_i - \Pr(x_i;\theta))\Big) \\
&= \sum_{i=1}^{n} x_i \frac{\partial}{\partial \theta_j}(y_i - \Pr(x_i;\theta)) \\
&= -\sum_{i=1}^{n} x_i (x_i)_j \Pr(x_i;\theta)(1-\Pr(x_i;\theta))
\end{aligned}$$

Thus, the jth column of the Jacobian of the gradient of $\mathcal{L}$ (The Hessian) is the above expression, which when viewed in matrix notation gives:

$$H = -\sum_{i=1}^{n} x_i x_i^T \Pr(x_i;\theta)(1-\Pr(x_i;\theta))$$

If we let $W := \text{diag}(y-p)$ we have:

$$H = -X^T W X$$

Thus our Newton-Raphson step is given by:

$$\begin{aligned}
\theta^{new} &= \theta^{old} - H^{-1}(X^T(y-p)) \\
&= \theta^{old} + (X^T W X)^{-1}(X^T(y-p))
\end{aligned}$$

We have all the necessary components to implement the algorithm, although in practice having to compute the inverse of the Hessian is expensive and potentially unstable. Thus we can use a shortcut, define $\Delta\theta := \theta^{new} - \theta^{old}$. Then:

$$\begin{aligned}
\Delta\theta &= (X^T W X)^{-1}(X^T(y-p)) \\
X^T W X(\Delta\theta) &= X^T(y-p)
\end{aligned}$$

Since $H = X^T W X$ is positive semidefinite, it can be made positive definite with some minor regularization. Thus, a solution to this linear system of equations is guaranteed. And, solving this system of equations is much quicker and safer than performing the direct matrix inversion of the Hessian. Additionally, since H is positive definite post-regularization we can apply the Cholesky decomposition to write $H = LL^T$ where L is a lower triangular matrix. Then we can first solve $L(z) = X^T(y - p)$ with solution z^* for $z = L^T(\Delta\theta)$ then solve $L^T(\Delta\theta) = z^*$ and this two step process is even faster due to the simplicity of L.

From here we have all the necessary machinery to build the Binary Logistic Regression model!