Binary Logistic Regression

Aaron Avram

June 4 2025

Introduction

In this write up I will go through my derivation of the objective function and the optimization algorithm for my Multinomial 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 $y \in \{1, \ldots, K\}^n$ equal the vector of training labels, where there are K output classes. I will apply the shortcut used in the Binary Model Construction where each row of X is given an additional final entry with a 1, to account for the bias term. So X is now a $n \times (p+1)$ dimensional matrix. Where we can let $p+1 = d$ as our feature number. The model is parametrized by $K-1$ $p+1$ dimensional vectors $\theta^{(1)}, \ldots, \theta^{(k-1)}$, which I will collectively refer to by the flattened combined vector θ. Where the probabilities are given by:

$$\Pr(\kappa|x; \theta^{(\kappa)}) = \begin{cases} \frac{\exp((\theta^{(\kappa)})^T x)}{1 + \displaystyle\sum_{1 \leq i < K} \exp((\theta^{(i)})^T x)} & \kappa < K \\ \frac{1}{1 + \displaystyle\sum_{1 \leq i < K} \exp((\theta^{(i)})^T x)} & \kappa = K \end{cases}$$

The log likelihood function is then given by:

$$\mathcal{L}(\theta) = \sum_{1 \leq r \leq n} \log \Pr(y_r | x_r; \theta^{y_r})$$

Using a one hot encoding with the kronecker delta function we can write this as:

$$= \sum_{1 \leq r \leq n} \sum_{1 \leq s \leq K} \delta_{y_r s} \log \Pr(y_r | x_r; \theta^{(s)})$$

Consider the Likelihood function with respect to one input vector x_r and one parameter $\theta^{(j)}$:

$$\mathcal{L}_{rj}(\theta) = \log \Pr(y_r | x_r)$$

However we can condense this further, as if $y_r \neq j$ the numerator of the probability function is irrelevant to $\theta^{(j)}$, and if we decompose the fraction via the logarithm rules we can omit the numerator term and replace it with a kronecker delta. I.e Suppose $\Pr = N/D$ then $\log(\Pr) = \log(N) - \log(D)$ and so if N is independent of $\theta^{(j)}$ we can omit it. Thus:

$$\begin{aligned}\mathcal{L}_{rj}(\theta) &= \delta_{y_r j} \log(\exp((\theta^{(j)})^T x_r)) - \log(1 + \sum_{1\le i<K} \exp((\theta^{(i)})^T x)) \\ &= \delta_{y_r j} (\theta^{(j)})^T x_r - \log(1 + \sum_{1\le i<K} \exp((\theta^{(i)})^T x))\end{aligned}$$

Gradient Calculations

First lets take the partial derivative with respect to $\theta^{(j)}$ we find

$$\begin{aligned}\frac{\partial \mathcal{L}_{rj}}{\partial \theta^{(j)}} &= \delta_{y_r j} x_r - \frac{x_r \exp((\theta^{(j)})^T x_r)}{1 + \sum\limits_{1\le i<K} \exp((\theta^{(i)})^T x)} \\ &= x_r(\delta_{y_r j} - \Pr(j|x_i))\end{aligned}$$

Reusing some calculations from the previous write up it is clear that $\frac{\partial \mathcal{L}}{\partial \theta^{(j)}} = X(y - p_j)$ Which form the columns of the Jacobian matrix of $\mathcal{L}$.

Now let us take the second partial derivative of this expression with respect to $\theta^{(i)}$

$$\frac{\partial}{\partial \theta^{(i)}} \frac{\partial \mathcal{L}_{rj}}{\partial \theta^{(j)}} = \begin{cases} -x_r x_r^T \Pr(j|x_r)(1 - \Pr(j|x_r)) & i = j \\ x_r x_r^T \Pr(j|x_r) \Pr(i|x_r) & i \neq j \end{cases}$$

We can make this more compact utilizing the kronecker delta function:

$$\frac{\partial}{\partial \theta^{(i)}} \frac{\partial \mathcal{L}_{rj}}{\partial \theta^{(j)}} = x_r x_r^T \Pr(i|x_r)(\delta_{ij} - \Pr(j|x_r))$$

Now let $\Pr(j|x_r) = p_{jr}$ So the total second derivative of $\mathcal{L}$ is:

$$\begin{aligned}\frac{\partial}{\partial \theta^{(i)}} \frac{\partial \mathcal{L}}{\partial \theta^{(j)}} &= \sum_{1\le r\le n} x_r x_r^T p_{ir}(\delta_{ij} - p_{jr}) \\ &= \sum_{1\le r\le n} x_r x_r^T p_{ir}(\delta_{ij} - p_{jr})\end{aligned}$$

If we let $W_{ij} = \operatorname{diag}(p_{ir}(\delta_{ij} - p_{jr}))_r$ We can write this more compactly by observing that

$$\frac{\partial}{\partial \theta^{(i)}} \frac{\partial \mathcal{L}}{\partial \theta^{(j)}} = X^T W_{ij} X$$

Thus the Hessian of the Log-Likelihood is the block matrix H with blocks $H_{ij} = W_{ij}$, where this is the Hessian of the flattened vector θ made by combining all $\theta^{(1)}, \ldots, \theta^{(K-1)}$. Note that the Gradient of this flattened vector is the vector created by stacking the columns of the Jacobian.

Optimization

The objective function will be maximized via the Newton-Raphson Method. Reusing work from the previous write up, the update rule is given by:

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

Where H represents the full hessian matrix and $\nabla_\theta(\mathcal{L})$ is the flattened Jacobian where each column of the Jacobian is stacked (So that the gradients of each class parameter are kept together). If we let $g = \nabla_\theta(\mathcal{L})(\theta^{(old)})$ We can simplify the computations in each step of the method by solving for $\Delta\theta = \theta^{(new)} - \theta^{(old)}$. Where $H(\Delta\theta) = g$. H is a semi-positive definite matrix. If we apply some regularization we can ensure that H is a positive definite matrix, and so can use the conjugate gradient method to solve for $\Delta\theta$. This method leverages the property that given $u, v \in \mathbb{R}^m$ $(m := d \times (C-1))$, $u^T H v$ defines an inner product $\langle u, v\rangle_H := u^T H v$, a fact that is easy to check. By Gram-Schmidt we can always find a basis of orthogonal vectors $P_1, \ldots, P_m$ for the inner product space given by $\mathbb{R}^m$ with respect to this inner product. Thus, we can express $\Delta\theta$ as a linear combination of these vectors with coefficients $\alpha_1, \ldots \alpha_m$. Thus:

$$\begin{aligned} H(\Delta\theta) &= H(\sum_{i=1}^{m} \alpha_i p_i) \\ g &= \sum_{i=1}^{m} \alpha_i H p_i \end{aligned}$$

Now left multiplying by taking the dot product with p_k we have:

$$\begin{aligned} p_k^T g &= p_k^T \sum_{i=1}^{m} \alpha_i H p_i \\ &= \sum_{i=1}^{m} \alpha_i p_k^T H p_i \\ &= \sum_{i=1}^{m} \alpha_i \langle p_k, p_i\rangle_H \end{aligned}$$

By the orthonormality of the basis we have:

$$\begin{aligned} p_k^T g &= \alpha_k \langle p_k, p_k\rangle \\ \alpha_k &= \frac{p_k^T g}{\langle p_k, p_k\rangle_H} \end{aligned}$$

This gives us an algorithm for computing the coefficient of a basis vector. We can combine this with the Gram-Schmidt algorithm for computing this orthogonal basis and recover the coefficients at each step.

This can save us a lot of time as H is only used to compute the inner product, but since H is a block matrix and v is a block vector we can use this to compute Hv without ever constructing H, saving a lot of computing time. To do this consider the block vector Hv where H and v are viewed as a block matrix and a block vector respectively. You can think of v as a magtrix with each column i being an arbitrary parameter vector for class i. By the block matrix-vector multiplication rule we have:

$$\begin{aligned}(Hv)_i &= \sum_{j=1}^{C-1} H_{ij} v_j \\ &= \sum_{j=1}^{C-1} X^T W_{ij} X v_j \\ &= X^T \sum_{j=1}^{C-1} W_{ij} X v_j\end{aligned}$$

Fix P to be the matrix with its r-th row containing the probability vector of the r-th training sample. Now consider first Xv_j:

$$Xv_j = \begin{bmatrix} X_1^T v_j \\ \vdots \\ X_n^T v_j \end{bmatrix}$$

If we let $Z = XV$ clearly $Z_{:,j} = Xv_j$

Since W_{ij} is a diagonal matrix, multiplying Xv_j by it corresponds to elementwise scaling by the diagonal elements of W_{ij}.

$$W_{ij} X v_j = \begin{bmatrix} (W_{ij})_{11} X_1^T v_j \\ \vdots \\ (W_{ij})_{nn} X_n^T v_j \end{bmatrix}$$

$$= \begin{bmatrix} P_{1i}(\delta_{ij} - P_{1j}) X_1^T v_j \\ \vdots \\ P_{ni}(\delta_{ij} - P_{nj}) X_n^T v_j \end{bmatrix}$$

$$= \begin{bmatrix} P_{1i}\delta_{ij} X_1^T v_j \\ \vdots \\ P_{ni}\delta_{ij} X_n^T v_j \end{bmatrix} - \begin{bmatrix} P_{1i} P_{1j} X_1^T v_j \\ \vdots \\ P_{ni} P_{nj} X_n^T v_j \end{bmatrix}.$$

However since we are summing these vectors over all j the Kronecker Delta term will be equal to 1 exactly once, thus:

$$\begin{aligned}\overline{Z}_{:,i} &:= \sum_{j=1}^{C-1} W_{ij} X v_j \\ &= P_{:,i} \odot Z_{:,i} - (P_{:,i}) \odot \sum_{j=1}^{C-1} P_{:,j} \odot Z_{:,j}\end{aligned}$$

So $(Hv)_i = X^T Z_{:,i}$. If we view $(Hv)_i$ as the i-th column in a matrix we can write: $Hv = X^T Z$. And we can compute Z directly. Let $A := P \odot Z$ which is clearly the matrix with $A_{:,i} = P_{:,i} \odot Z_{:,i}$. Then if we consider $A(1_{C-1})$ this is the vector with the each entry corresponding to the sum $\sum_{j=1}^{C-1} P_{:,j} \odot Z_{:,j}$

Then if we take $P \odot A(1_{C-1})$ where we broadcast by copying the columns of the sum vector to form a matrix, we have $\overline{Z} = A - P \odot A(1_{C-1})$ and with this we have constructed Hv without actually building the explicit matrix, and can use this for the conjugate gradient method.