1 Multiclass Logistic Regression

Recall that in logistic regression, we are tuning a weight vector $w \in \mathbb{R}^{d+1}$, which leads to a posterior distribution $Q_i \sim Bernoulli(p_i)$ over the binary classes 0 and 1:

$$P(Q_i = 1) = p_i = s(w^T x_i) = \frac{1}{1+e^{-w^T x_i}}$$
$$P(Q_i = 0) = 1 - s(w^T x_i) = \frac{e^{-w^T x_i}}{1+e^{-w^T x_i}}$$

Let's generalize this concept to **Multiclass Logistic Regression**, where there are K classes. Similarly to our discussion of the multi-class LS-SVM, it is important to note that there is no inherent ordering to the classes, and predicting a class in the continuous range from 1 to K would be a poor choice. To see why, recall our fruit classification example. Suppose 1 is used to represent "peach," 2 is used to represent "banana," and 3 is used to represent "apple." In our numerical representation, it would appear that peaches are less than bananas, which are less than apples. As a result, if we have an image that looks like some cross between an apple and a peach, we may simply end up classifying it as a banana.

The solution is to use a **one-hot vector encoding** to represent all of our labels. If the i-th observation has class k, instead of using the representation $y_i = k$, we can use the representation $y_i = e_k$, the k-th canonical basis vector. For example, in our fruit example, if the i-th image is classified as "banana", its label representation would be

$$y_i = [0\ 1\ 0]^T$$

Now there is no relative ordering in the representations of the classes. We must modify our weight representation accordingly to the one-hot vector encoding. Now, there are a set of $d+1$ weights associated with every class, which amounts to a matrix $W \in \mathbb{R}^{K\times(d+1)}$. For each input $x_i \in \mathbb{R}^{d+1}$, each class k is given a "score"

$$z_k = w_k^T x_i$$

Where w_k is the k-th row of the W matrix. In total there are K scores for an input x_i:

$$[w_1^T x_i \quad w_2^T x_i \quad \dots \quad w_K^T x_i]$$

The higher the score for a class, the more likely logistic regression will pick that class. Now that we have a score system, we must transform all of these scores into a posterior probability distribution Q. For binary logistic regression, we used the logistic function, which takes the value $w^T x_i$ and squashes it to a value between 0 and 1. The generalization to the the logistic function for the multi-class case is the **softmax function**. The softmax function takes as input all K scores (formally known as **logits**) and an index j, and outputs the probability that the corresponding softmax distribution takes value j:

$$\text{softmax}(j, \{z_1, z_2, \ldots, z_K\}) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

The logits induce a **softmax distribution**, which we can verify is indeed a probability distribution:

1. The entries are between 0 and 1.

2. The entries add up to 1.

On inspection, this softmax distribution is reasonable, because the higher the score of a class, the higher its probability. In fact, we can verify that the logistic function is a special case of the softmax function. Assuming that the corresponding weights for class 0 and 1 are w_0 and w_1, we have that:

$$P(Q_i = 1) = \frac{e^{w_1^T x_i}}{e^{w_0^T x_i} + e^{w_1^T x_i}} = \frac{e^{(w_1 - w_1)^T x_i}}{e^{(w_0 - w_1)^T x_i} + e^{(w_1 - w_1)^T x_i}} = \frac{1}{1 + e^{-(w_1 - w_0)^T x_i}} = s((w_1 - w_0)^T x_i)$$

$$P(Q_i = 0) = \frac{e^{w_0^T x_i}}{e^{w_0^T x_i} + e^{w_1^T x_i}} = \frac{e^{(w_0 - w_1)^T x_i}}{e^{(w_0 - w_1)^T x_i} + e^{(w_1 - w_1)^T x_i}} = \frac{e^{-(w_1 - w_0)^T x_i}}{1 + e^{-(w_1 - w_0)^T x_i}} = 1 - s((w_1 - w_0)^T x_i)$$

In the 2-class case, because we are only interested in the difference between w_1 and w_0, we just use a change of variables $w = w_1 - w_0$. We don't need to know w_1 and w_0 individually, because once we know $P(Q_i = 1)$, we know by default that $P(Q_i = 0) = 1 - P(Q_i = 1)$.

1.1 Multiclass Logistic Regression Loss Function

Let's derive the loss function for multiclass logistic regression, using the information-theoretic perspective. The "true" or more formally the **target distribution** in this case is $P(P_i = j) = y_i[j]$. In other words, the entire distribution is concentrated on the label for the training example. The estimated distribution Q comes from multiclass logistic regression, and in this case is the softmax distribution:

$$P(Q_i = j) = \frac{e^{w_j^T x_i}}{\sum_{k=1}^{K} e^{w_k^T x_i}}$$

Now let's proceed to deriving the loss function. The objective, as always, is to minimize the sum of the KL divergences contributed by all of the training examples.

$$
\begin{aligned}
W^*_{MCLR} &= \arg\min_W \sum_{i=1}^n D_{KL}(P_i||Q_i) \\
&= \arg\min_W \sum_{i=1}^n \sum_{j=1}^K P(P_i = j) \ln \frac{P(P_i = j)}{P(Q_i = j)} \\
&= \arg\min_W \sum_{i=1}^n \sum_{j=1}^K y_i[j] \ln \frac{y_i[j]}{\text{softmax}(j, \{w_1^T x_i, w_1^T x_i, \ldots, w_K^T x_i\})} \\
&= \arg\min_W \sum_{i=1}^n \sum_{j=1}^K y_i[j] \cdot \ln y_i[j] - y_i[j] \cdot \ln \big(\text{softmax}(j, \{w_1^T x_i, w_2^T x_i, \ldots, w_K^T x_i\}) \big) \\
&= \arg\min_W - \sum_{i=1}^n \sum_{j=1}^K y_i[j] \cdot \ln \big(\text{softmax}(j, \{w_1^T x_i, w_1^T x_i, \ldots, w_K^T x_i\}) \big) \\
&= \arg\min_W - \sum_{i=1}^n \sum_{j=1}^K y_i[j] \cdot \ln \left(\frac{e^{w_j^T x_i}}{\sum_{k=1}^K e^{w_k^T x_i}} \right) \\
&= \arg\min_W \sum_{i=1}^n H(P_i, Q_i)
\end{aligned}
$$

Just like binary logistic regression, we can justify the loss function with MLE as well:

$$
\begin{aligned}
W^*_{MCLR} &= \arg\max_W \prod_{i=1}^n P(Y_i = y_i) \\
&= \arg\max_W \prod_{i=1}^n \prod_{j=1}^K P(Q_i = j)^{y_i[j]} \\
&= \arg\max_W \sum_{i=1}^n \sum_{j=1}^K y_i[j] \ln P(Q_i = j) \\
&= \arg\max_W \sum_{i=1}^n \sum_{j=1}^K y_i[j] \ln \left(\frac{e^{w_j^T x_i}}{\sum_{k=1}^K e^{w_k^T x_i}} \right) \\
&= \arg\min_W - \sum_{i=1}^n \sum_{j=1}^K y_i[j] \ln \left(\frac{e^{w_j^T x_i}}{\sum_{k=1}^K e^{w_k^T x_i}} \right)
\end{aligned}
$$

We conclude that the loss function for multiclass logistic regression is

$$
L(W) = - \sum_{i=1}^n \sum_{j=1}^K y_i[j] \cdot \ln P(Q_i = j)
$$

2 Training Logistic Regression

The logistic regression loss function has no known analytic closed-form solution. Therefore, in order to minimize it, we can use gradient descent, either in batch form or stochastic form. Let's examine the case for batch gradient descent.

2.1 Binary Logistic Regression

Recall the loss function

$$L(w) = -\sum_{i=1}^{n} y_i \ln p_i + (1 - y_i)\ln(1 - p_i)$$

where

$$p_i = s(w^T x_i) = \frac{1}{1 + e^{-w^T x_i}}$$

$$\begin{aligned}
\nabla_w L(w) &= \nabla_w \left(-\sum_{i=1}^{n} y_i \ln p_i + (1 - y_i)\ln(1 - p_i) \right) \\
&= -\sum_{i=1}^{n} y_i \nabla_w \ln p_i + (1 - y_i) \nabla_w \ln(1 - p_i) \\
&= -\sum_{i=1}^{n} \frac{y_i}{p_i} \nabla_w p_i - \frac{1 - y_i}{1 - p_i} \nabla_w p_i \\
&= -\sum_{i=1}^{n} \left(\frac{y_i}{p_i} - \frac{1 - y_i}{1 - p_i} \right) \nabla_w p_i
\end{aligned}$$

Note that $\nabla_z s(z) = s(z)(1 - s(z))$, and from the chain rule we have that

$$\nabla_w p_i = \nabla_w s(w^T x_i) = s(w^T x_i)(1 - s(w^T x_i)) x_i = p_i(1 - p_i) x_i$$

Plugging in this gradient value, we have

$$\begin{aligned}
\nabla_w L(w) &= -\sum_{i=1}^{n} \left(\frac{y_i}{p_i} - \frac{1 - y_i}{1 - p_i} \right) \nabla_w p_i \\
&= -\sum_{i=1}^{n} \left(\frac{y_i}{p_i} - \frac{1 - y_i}{1 - p_i} \right) p_i(1 - p_i) x_i \\
&= -\sum_{i=1}^{n} \left(y_i(1 - p_i) - (1 - y_i)(p_i) \right) x_i \\
&= -\sum_{i=1}^{n} (y_i - p_i)\, x_i
\end{aligned}$$

We conclude that the gradient of the loss function with respect to the parameters is

$$\nabla_w L(w) = -\sum_{i=1}^{n} (y_i - p_i)\, x_i = -X^T (y - p)$$

where $y, p \in \mathbb{R}^n$. The gradient descent update is thus

$$w = w - \varepsilon \nabla_w L(w)$$

It does not matter what initial values we pick for w, because the loss function $L(w)$ is convex and does not have any local minima. Let's prove this, by first finding the Hessian of the loss function. The k,lth entry of the Hessian is the partial derivative of the gradient with respect to w_k and w_l:

$$\begin{aligned}
H_{kl} &= \frac{\partial^2 L(w)}{\partial w_k \partial w_l} \\
&= \frac{\partial}{\partial w_k} - \sum_{i=1}^{n} (y_i - p_i)\, x_{il} \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial w_k} p_i x_{il} \\
&= \sum_{i=1}^{n} p_i(1-p_i) x_{ik} x_{il}
\end{aligned}$$

We conclude that

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

To prove that $L(w)$ is convex in w, we need to show that $w^T H w \geq 0, \quad \forall w$:

$$w^T H w = w^T \sum_{i=1}^{n} p_i(1-p_i) x_i x_i^T w = \sum_{i=1}^{n} (w^T x_i)^2 p_i(1-p_i) \geq 0$$

2.2 Multiclass Logistic Regression

Instead of finding the gradient with respect to all of the parameters of the matrix W, let's find them with respect to one row of W at a time:

$$
\begin{aligned}
\nabla_{w_l} L(W) &= \nabla_{w_l}\left(-\sum_{i=1}^{n}\sum_{j=1}^{K} y_i[j]\cdot\ln\left(\frac{e^{w_j^T x_i}}{\sum_{k=1}^{K} e^{w_k^T x_i}}\right)\right) \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{K} y_i[j]\cdot\nabla_{w_l}\left(\ln\frac{e^{w_j^T x_i}}{\sum_{k=1}^{K} e^{w_k^T x_i}}\right) \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{K} y_i[j]\cdot\left(\nabla_{w_l} w_j^T x_i - \nabla_{w_l}\ln\sum_{k=1}^{K} e^{w_k^T x_i}\right) \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{K} y_i[j]\cdot\left(\mathbb{1}\{j=l\}x_i - \frac{e^{w_l^T x_i}}{\sum_{k=1}^{K} e^{w_k^T x_i}} x_i\right) \\
&= -\sum_{i=1}^{n}\sum_{j=1}^{K} y_i[j]\cdot\left(\mathbb{1}\{j=l\} - \frac{e^{w_l^T x_i}}{\sum_{k=1}^{K} e^{w_k^T x_i}}\right) x_i \\
&= -\sum_{i=1}^{n}\left(\mathbb{1}\{y_i=l\} - P(Q_i=l)\right) x_i \\
&= -X^T\left(\mathbb{1}\{y_i=l\} - P(Q=l)\right)
\end{aligned}
$$

Note the use of indicator functions: $\mathbb{1}\{j=l\}$ evaluates to 1 if $j=l$, otherwise 0. Also note that since y_i is a one-hot vector encoding, it evaluates to 1 only for one entry and 0 for all other entries. We can therefore simplify the expression by only considering the j for which $y_i[j]=1$. The gradient descent update for w_l is

$$w_l = w_l - \varepsilon\nabla_{w_l} L(W)$$

Just as with binary logistic regression, it does not matter what initial values we pick for W, because the loss function $L(W)$ is convex and does not have any local minima.