1 Logistic Regression

Logistic regression is a discriminative classification technique that has a direct probabilistic interpretation. We will first present the binary class case, and then we can easily extend the logic to the multiclass case.

1.1 Binary Logistic Regression

Suppose that we have the binary classification problem where classes are represented by 0 and 1. Note that we instead of using $-1/+1$ labels (as in LS-SVM), in binary logistic regression we use $0/1$ labels. Logistic regression makes more sense this way because it directly outputs a probability, which belongs in the range of values between 0 and 1.

In binary logistic regression, we would like our model to output the probability that a data point is in class 0 or 1. We can start with the raw linear "score" $\mathbf{w}^\top \mathbf{x}$ and convert it to a probability between 0 and 1 by applying a sigmoid transformation $s(\mathbf{w}^\top \mathbf{x})$, where $s(z) = \frac{1}{1+e^{-z}}$. To classify an arbitrary point $\mathbf{x}$, we use the sigmoid function to output a probability distribution $P(\hat{Y})$ over the classes 0 and 1:

$$P(\hat{Y} = 1 \mid \mathbf{x}, \mathbf{w}) = s(\mathbf{w}^\top \mathbf{x}), \quad P(\hat{Y} = 0 \mid \mathbf{x}, \mathbf{w}) = 1 - s(\mathbf{w}^\top \mathbf{x})$$

we classify $\mathbf{x}$ as the class with the maximum probability:

$$\hat{y} = \max_k P(\hat{Y} = k \mid \mathbf{x}, \mathbf{w}) = \begin{cases} 1 & \text{if } s(\mathbf{w}^\top \mathbf{x}) \geq 0.5 \\ 0 & \text{otherwise} \end{cases}$$

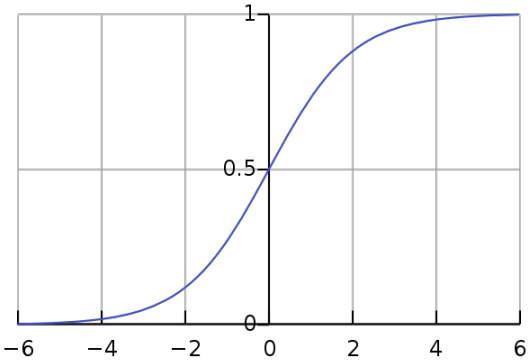

Figure 1: Logistic function. For our purposes, the horizontal axis is the output of the linear function $\mathbf{w}^\top \mathbf{x}_i$ and the vertical axis is the output of the logistic function, which can be interpreted as a probability between 0 and 1.

Equivalently, we classify $\mathbf{x}$ as

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{w}^\top \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

1.1.1 Loss Function

Suppose we are given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. In order to train our model, we need a loss function to optimize. One possibility is least squares:

$$\arg\min_{\mathbf{w}} \sum_{i=1}^{n} \|y_i - s(\mathbf{w}^\top \mathbf{x}_i)\|^2 + \lambda \|\mathbf{w}\|^2$$

However, this may not be the best choice. Ordinary least squares regression has theoretical justifications such as being the maximum likelihood objective under Gaussian noise. Least squares for this classification problem does not have a similar justification.

Instead, the loss function we use for logistic regression is called the log-loss, or **cross entropy**:

$$L(\mathbf{w}) = \sum_{i=1}^{n} y_i \ln\left(\frac{1}{s(\mathbf{w}^\top \mathbf{x}_i)}\right) + (1 - y_i) \ln\left(\frac{1}{1 - s(\mathbf{w}^\top \mathbf{x}_i)}\right)$$

If we define $p_i = s(\mathbf{w}^\top \mathbf{x}_i)$, then using the properties of logs we can express this as

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

For each $\mathbf{x}_i$, p_i represents our predicted probability that its corresponding class is 1. Because $y_i \in \{0, 1\}$, the loss corresponding to the i'th data point is

$$L_i(\mathbf{w}) = \begin{cases} -\ln p_i & \text{when } y_i = 1 \\ -\ln(1 - p_i) & \text{when } y_i = 0 \end{cases}$$

Intuitively, if $p_i = y_i$, then we incur 0 loss. However, this is never actually the case. The logistic function can never actually output a value of exactly 0 or 1, and we will therefore always incur some loss. If the actual label is $y_i = 1$, then as we lower p_i towards 0, the loss for this data point approaches infinity.

The loss function can be derived from a maximum likelihood perspective or an information-theoretic perspective. First let's present the maximum likelihood perspective. We view each observations y_i as an independent sample from a Bernoulli distribution $\hat{Y}_i \sim \text{Bern}(p_i)$ (technically we mean $\hat{Y}_i \mid \mathbf{x}_i, \mathbf{w}$, but we remove the conditioning terms for notational brevity), where p_i is a function of $\mathbf{x}_i$. Thus our observation y_i, which we can view as a "sample," has probability

$$P(\hat{Y}_i = y_i) = \begin{cases} p_i & \text{if } y_i = 1 \\ 1 - p_i & \text{if } y_i = 0 \end{cases}$$

One convenient way to write the likelihood of a single data point is

$$P(\hat{Y}_i = y_i) = p_i^{y_i}(1-p_i)^{(1-y_i)}$$

which holds no matter what y_i is.

We need a model for the dependency of p_i on $\mathbf{x}_i$. We have to enforce that p_i is a transformation of $\mathbf{x}_i$ that results in a number from 0 to 1 (ie. a valid probability). Hence p_i cannot be, say, linear in $\mathbf{x}_i$. One way to do achieve the 0-1 normalization is by using the sigmoid function

$$p_i = s(\mathbf{w}^\top \mathbf{x}_i) = \frac{1}{1+e^{-\mathbf{w}^\top \mathbf{x}_i}}$$

Now we can estimate the parameters $\mathbf{w}$ via maximum likelihood. We have the problem

$$\hat{\mathbf{w}}_{\text{LR}} = \arg\max_{\mathbf{w}} P(\hat{Y}_1 = y_1, \ldots, \hat{Y}_n = y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w})$$

$$= \arg\max_{\mathbf{w}} \prod_{i=1}^{n} P(\hat{Y}_i = y_i \mid \mathbf{x}_i, \mathbf{w})$$

$$= \arg\max_{\mathbf{w}} \prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}$$

$$= \arg\max_{\mathbf{w}} \ln\left[\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{(1-y_i)}\right]$$

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

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

which exactly matches the cross-entropy formulation from earlier. The logistic regression loss function can also be justified from an information-theoretic perspective. To motivate this approach, we introduce **Kullback-Leibler (KL) divergence** (also called **relative entropy**), which measures the amount that one distribution diverges from another. Given *any* two discrete random variables P and Q, the KL divergence from Q to P is defined as

$$D_{\text{KL}}(P \parallel Q) = \sum_x P(x) \ln \frac{P(x)}{Q(x)}$$

Note that D_{KL} is not a true distance metric, because it is not symmetric, ie. $D_{\text{KL}}(P \parallel Q) \neq D_{\text{KL}}(Q \parallel P)$ in general. It also does not satisfy the triangle inequality. However, it is always positive, ie. $D_{\text{KL}}(P \parallel Q) \geq 0$, with equality iff $P = Q$.

In the context of classification, if the class label y_i is interpreted as the probability of being class 1, then logistic regression provides an estimate p_i of the probability that the data is in class 1. The true class label can be viewed as a sampled value from the "true" distribution $Y_i \sim \text{Bern}(y_i)$. $P(Y_i)$ is not a particularly interesting distribution because all values sampled from it will yield y_i:

$P(Y_i = y_i) = 1$. Logistic regression yields a distribution $\hat{Y}_i \sim \text{Bern}(p_i)$, which is the posterior probability that estimates the true distribution $P(Y_i)$.

The KL divergence from $P(\hat{Y}_i)$ to $P(Y_i)$ provides a measure of how closely logistic regression can match the true label. We would like to minimize this KL divergence, and ideally we would try to choose our parameters so that $D_{\text{KL}}(P(Y_i) \parallel P(\hat{Y}_i)) = 0$. Again, this is impossible for two reasons. First, if we want $D_{\text{KL}}(P(Y_i) \parallel P(\hat{Y}_i)) = 0$, then we would need $p_i = y_i$, which is impossible because p_i is the output of a logistic function that can never actually reach 0 or 1. Second, even if we tried tuning the parameters so that $D_{\text{KL}}(P(Y_i) \parallel P(\hat{Y}_i)) = 0$, that's only optimizing one of the data points – we need to tune the parameters so that we can collectively minimize the totality of all of the KL divergences contributed by all data points.

Therefore, our goal is to tune the parameters $\mathbf{w}$ (which indirectly affect the p_i values and therefore the estimated distribution $P(\hat{Y}_i)$), in order to minimize the total sum of KL divergences contributed by all data points:

$$\begin{aligned}
\hat{\mathbf{w}}_{\text{LR}} &= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} D_{\text{KL}}(P(Y_i) \parallel P(\hat{Y}_i)) \\
&= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} y_i \ln \frac{y_i}{p_i} + (1 - y_i) \ln \frac{(1 - y_i)}{(1 - p_i)} \\
&= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} y_i(\ln y_i - \ln p_i) + (1 - y_i)(\ln(1 - y_i) - \ln(1 - p_i)) \\
&= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (-y_i \ln p_i - (1 - y_i) \ln(1 - p_i)) + (y_i \ln y_i + (1 - y_i) \ln(1 - y_i)) \\
&= \arg\min_{\mathbf{w}} -\sum_{i=1}^{n} y_i \ln p_i + (1 - y_i) \ln(1 - p_i) \\
&= \arg\min_{\mathbf{w}} \sum_{i=1}^{n} H(P(Y_i), P(\hat{Y}_i))
\end{aligned}$$

Note that the $y_i \ln y_i + (1 - y_i) \ln(1 - y_i)$ component of the KL divergence is a constant, independent of our changes to p_i. Therefore, we are effectively minimizing the sum of the **cross entropies** $H(P(Y_i), P(\hat{Y}_i))$. We conclude our discussion of KL Divergence by noting the relation between KL divergence and cross entropy:

$$D_{\text{KL}}(P(Y_i) \parallel P(\hat{Y}_i)) = H(P(Y_i), P(\hat{Y}_i)) - H(P(Y_i))$$

where:

1. $D_{\text{KL}}(P(Y_i) \parallel P(\hat{Y}_i))$ is the KL divergence from $P(\hat{Y}_i)$ to $P(Y_i)$:

$$D_{\text{KL}}(P(Y_i) \parallel P(\hat{Y}_i)) = y_i \ln \frac{y_i}{p_i} + (1 - y_i) \ln \frac{(1 - y_i)}{(1 - p_i)}$$

2. $H(P(Y_i), P(\hat{Y}_i))$ is the cross entropy between Y_i and $\hat{Y}_i$:

$$H(P(Y_i), P(\hat{Y}_i)) = -y_i \ln p_i - (1 - y_i) \ln(1 - p_i)$$

3. $H(Y_i)$ is the entropy of Y_i:

$$H(Y_i) = -y_i \ln y_i - (1 - y_i) \ln(1 - y_i)$$

Since the parameters $\mathbf{W}$ do not affect the entropy, we can optimize the cross entropy instead of the KL divergence.

1.2 Multiclass Logistic Regression

Let's generalize logistic regression to the case 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 $\mathbf{y}_i = \mathbf{e}_k$, 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

$$\mathbf{y}_i = [0 \; 1 \; 0]$$

(Be careful to distinguish between the class label $y_i \in \mathbb{R}$ and its one-hot encoding $\mathbf{y}_i \in \mathbb{R}^K$). Now there is no relative ordering in the representations of the classes. We must modify our parameter representation accordingly to the one-hot vector encoding. Now, there are a set of $d+1$ parameters associated with every class, which amounts to a matrix $\mathbf{W} \in \mathbb{R}^{K \times (d+1)}$. For each input $\mathbf{x}_i \in \mathbb{R}^{d+1}$, each class k is given a "score"

$$z_k = \mathbf{w}_k{}^\top \mathbf{x}_i$$

Where $\mathbf{w}_k \in \mathbb{R}^{d+1}$ is the k'th row of the $\mathbf{W}$ matrix. In total there are K raw linear scores for an arbitrary input $\mathbf{x}$:

$$\begin{bmatrix} \mathbf{w}_1{}^\top \mathbf{x} & \mathbf{w}_2{}^\top \mathbf{x} & \dots & \mathbf{w}_K{}^\top \mathbf{x} \end{bmatrix}$$

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 $P(\hat{Y})$. For binary logistic regression, we used the logistic function, which takes the value $\mathbf{w}^\top \mathbf{x}$ 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:

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

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.

Using the softmax function, we generate probabilities for each class:

$$\hat{\mathbf{y}} = \sigma(\mathbf{W}\mathbf{x}) = \begin{bmatrix} \sigma(\mathbf{W}\mathbf{x})_1 & \sigma(\mathbf{W}\mathbf{x})_2 & \dots & \sigma(\mathbf{W}\mathbf{x})_K \end{bmatrix}$$

and our prediction is the class with the maximum probability:

$$\hat{y} = \max_k \sigma(\mathbf{W}\mathbf{x})_k$$

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 used in the binary case is a special case of the softmax function used in the multiclass case. Assuming that the corresponding parameters for class 0 and 1 are $\mathbf{w}_0$ and $\mathbf{w}_1$, we have that:

$$P(\hat{Y}_i = 1 \mid \mathbf{x}_i, \mathbf{W}) = \sigma(\mathbf{W}\mathbf{x}_i)_1 = \frac{e^{\mathbf{w}_1{}^\top \mathbf{x}_i}}{e^{\mathbf{w}_0{}^\top \mathbf{x}_i} + e^{\mathbf{w}_1{}^\top \mathbf{x}_i}} = s((\mathbf{w}_1 - \mathbf{w}_0)^\top \mathbf{x}_i)$$

$$P(\hat{Y}_i = 0 \mid \mathbf{x}_i, \mathbf{W}) = \sigma(\mathbf{W}\mathbf{x}_i)_0 = \frac{e^{\mathbf{w}_0{}^\top \mathbf{x}_i}}{e^{\mathbf{w}_0{}^\top \mathbf{x}_i} + e^{\mathbf{w}_1{}^\top \mathbf{x}_i}} = 1 - s((\mathbf{w}_1 - \mathbf{w}_0)^\top \mathbf{x}_i)$$

In the 2-class case, because we are only interested in the difference between $\mathbf{w}_1$ and $\mathbf{w}_0$, we just use a change of variables $\mathbf{w} = \mathbf{w}_1 - \mathbf{w}_0$. We don't need to know $\mathbf{w}_1$ and $\mathbf{w}_0$ individually, because once we know $P(\hat{Y}_i = 1)$, we know by default that $P(\hat{Y}_i = 0) = 1 - P(\hat{Y}_i = 1)$.

1.2.1 Loss Function

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

$$P(\hat{Y}_i = j) = \frac{e^{\mathbf{w}_j{}^\top \mathbf{x}_i}}{\sum_{k=1}^K e^{\mathbf{w}_k{}^\top \mathbf{x}_i}}$$

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.

$$\begin{aligned} \hat{\mathbf{W}}_{\text{MCLR}} &= \arg\min_{\mathbf{W}} \sum_{i=1}^n D_{\text{KL}}(P(Y_i) \parallel P(\hat{Y}_i)) \\ &= \arg\min_{\mathbf{W}} \sum_{i=1}^n \sum_{j=1}^K P(Y_i = j) \ln \left(\frac{P(Y_i = j)}{P(\hat{Y}_i = j)} \right) \end{aligned}$$

$$
\begin{aligned}
&= \arg\min_{\mathbf{W}} \sum_{i=1}^{n} \sum_{j=1}^{K} \delta_{j,y_i} \ln \left(\frac{\delta_{j,y_i}}{\sigma(\mathbf{W}\mathbf{x}_i)_j} \right) \\
&= \arg\min_{\mathbf{W}} \sum_{i=1}^{n} \sum_{j=1}^{K} \delta_{j,y_i} \cdot \ln \delta_{j,y_i} - \delta_{j,y_i} \cdot \ln \left(\sigma(\mathbf{W}\mathbf{x}_i)_j \right) \\
&= \arg\min_{\mathbf{W}} - \sum_{i=1}^{n} \sum_{j=1}^{K} \delta_{j,y_i} \cdot \ln \left(\sigma(\mathbf{W}\mathbf{x}_i)_j \right) \\
&= \arg\min_{\mathbf{W}} - \sum_{i=1}^{n} \sum_{j=1}^{K} \delta_{j,y_i} \cdot \ln \left(\frac{e^{\mathbf{w}_j{}^\top \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\mathbf{w}_k{}^\top \mathbf{x}_i}} \right) \\
&= \arg\min_{\mathbf{W}} \sum_{i=1}^{n} H(P(Y_i), P(\hat{Y}_i))
\end{aligned}
$$

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

$$
\begin{aligned}
\hat{\mathbf{W}}_{\text{MCLR}} &= \arg\max_{\mathbf{w}} P(\hat{Y}_1 = y_1, \ldots, \hat{Y}_n = y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{w}) \\
&= \arg\max_{\mathbf{w}} \prod_{i=1}^{n} P(\hat{Y}_i = y_i \mid \mathbf{x}_i, \mathbf{w})
\end{aligned}
$$

$$
\begin{aligned}
&= \arg\max_{\mathbf{W}} \prod_{i=1}^{n} \prod_{j=1}^{K} P(\hat{Y}_i = j \mid \mathbf{x}_i, \mathbf{W})^{\delta_{j,y_i}} \\
&= \arg\max_{\mathbf{W}} \sum_{i=1}^{n} \sum_{j=1}^{K} \delta_{j,y_i} \ln P(\hat{Y}_i = j \mid \mathbf{x}_i, \mathbf{W}) \\
&= \arg\max_{\mathbf{W}} \sum_{i=1}^{n} \sum_{j=1}^{K} \delta_{j,y_i} \ln \left(\frac{e^{\mathbf{w}_j{}^\top \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\mathbf{w}_k{}^\top \mathbf{x}_i}} \right) \\
&= \arg\min_{\mathbf{W}} - \sum_{i=1}^{n} \sum_{j=1}^{K} \delta_{j,y_i} \ln \left(\frac{e^{\mathbf{w}_j{}^\top \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\mathbf{w}_k{}^\top \mathbf{x}_i}} \right)
\end{aligned}
$$

We conclude that the loss function for multiclass logistic regression is

$$
L(\mathbf{W}) = - \sum_{i=1}^{n} \sum_{j=1}^{K} \delta_{j,y_i} \cdot \ln P(\hat{Y}_i = j \mid \mathbf{x}_i, \mathbf{W})
$$

1.3 Training

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.

1.3.1 Binary Case

Recall the loss function

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

where

$$p_i = s(\mathbf{w}^\top \mathbf{x}_i) = \frac{1}{1 + e^{-\mathbf{w}^\top \mathbf{x}_i}}$$

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

Note that $\nabla_{\mathbf{z}} s(\mathbf{z}) = s(\mathbf{z})(1 - s(\mathbf{z}))$, and from the chain rule we have that

$$\nabla_{\mathbf{w}} p_i = \nabla_{\mathbf{w}} s(\mathbf{w}^\top \mathbf{x}_i) = s(\mathbf{w}^\top \mathbf{x}_i)(1 - s(\mathbf{w}^\top \mathbf{x}_i))\mathbf{x}_i = p_i(1 - p_i)\mathbf{x}_i$$

Plugging in this gradient value, we have

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

The gradient descent update is thus

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \epsilon \sum_{i=1}^{n} (y_i - p_i)\, \mathbf{x}_i$$

It does not matter what initial values we pick for $\mathbf{w}$, because the loss function $L(\mathbf{w})$ 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 $\mathbf{w}_k$ and $\mathbf{w}_\ell$:

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

We conclude that

$$H(\mathbf{w}) = \sum_{i=1}^{n} p_i(1-p_i)\mathbf{x}_i\mathbf{x}_i^\top$$

To prove that $L(\mathbf{w})$ is convex in $\mathbf{w}$, we need to show that $\forall \mathbf{v}, \mathbf{v}^\top H(\mathbf{w})\mathbf{v} \geq 0$.

$$\mathbf{v}^\top H(\mathbf{w})\mathbf{v} = \mathbf{v}^\top \sum_{i=1}^{n} p_i(1-p_i)\mathbf{x}_i\mathbf{x}_i^\top \mathbf{v} = \sum_{i=1}^{n} (\mathbf{v}^\top \mathbf{x}_i)^2 p_i(1-p_i) \geq 0$$

1.3.2 Multiclass Case

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

$$\begin{aligned} \nabla_{\mathbf{w}_\ell} L(\mathbf{W}) &= \nabla_{\mathbf{w}_\ell} - \sum_{i=1}^{n}\sum_{j=1}^{K} \delta_{j,y_i} \cdot \ln\left(\frac{e^{\mathbf{w}_j{}^\top \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\mathbf{w}_k{}^\top \mathbf{x}_i}}\right) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{K} \delta_{j,y_i} \cdot \nabla_{\mathbf{w}_\ell}\left(\ln\frac{e^{\mathbf{w}_j{}^\top \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\mathbf{w}_k{}^\top \mathbf{x}_i}}\right) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{K} \delta_{j,y_i} \cdot \left(\nabla_{\mathbf{w}_\ell} \mathbf{w}_j{}^\top \mathbf{x}_i - \nabla_{\mathbf{w}_\ell} \ln \sum_{k=1}^{K} e^{\mathbf{w}_k{}^\top \mathbf{x}_i}\right) \\ &= -\sum_{i=1}^{n}\sum_{j=1}^{K} \delta_{j,y_i} \cdot \left(\delta_{\ell,y_i} - \frac{e^{\mathbf{w}_\ell{}^\top \mathbf{x}_i}}{\sum_{k=1}^{K} e^{\mathbf{w}_k{}^\top \mathbf{x}_i}}\right)\mathbf{x}_i \\ &= -\sum_{i=1}^{n}\left(\delta_{\ell,y_i} - P(\hat{Y}_i = \ell)\right)\mathbf{x}_i \end{aligned}$$

The gradient descent update for $\mathbf{w}_\ell$ is thus

$$\mathbf{w}_\ell^{(t+1)} = \mathbf{w}_\ell^{(t+1)} + \epsilon \sum_{i=1}^{n} \left(\delta_{\ell,y_i} - P(\hat{Y}_i = \ell) \right) \mathbf{x}_i$$

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