Logistic Regression

Rohit Amarnath

October 20, 2023

1 Overview

The following notes serve as an introduction to logistic regression. It begins with a discussion on the intuition behind the cross-entropy loss, and then derives the loss function for a logistic regression model through both MLE and MAP. It ends with a final discussion of logistic regression in multi-class conditions.

2 Cross Entropy Loss

Recall that the *information content* of a particular outcome x is the uncertainity associated with the outcome happening. It is calculated by computing $\log_2\left(\frac{1}{p(x)}\right)$ where $p(x)$ is the probability of the outcome x. The *entropy* is defined as the expected uncertainty across all possible outcomes of a particular system. Given a discrete random variable X, which takes values in the sample space $\mathcal{X}$, the entropy is defined as

$$H(X) := -\sum_{x\in\mathcal{X}} p(x)\log(p(x)) = \mathbb{E}[-\log p(x)]$$

In a classification problem, it is common to measure the *dissimilarity* between the model's output distribution and the true output distribution. The Kullback-Leibler Divergence is a distance metric that measures the difference between two probability distributions P and Q over a sample space $\mathcal{X}$. It is defined as

$$D_{KL} = \sum_{x\in X} P(x)\log\left(\frac{P(x)}{Q(x)}\right) = \sum_{x\in X} P(x)\log P(x) - P(x)\log Q(x)$$

The goal of any classification model is to learn a mapping from the input x to the output distribution. For $(x, y) \in \mathcal{D}$, we can one hot encode y to form a discrete distribution over the sample space of classes $\mathcal{K}$. For example, if we had 3 classes, and $y = 2$, then the true output distribution would be $(0, 1, 0)$.

For simplicitly, let each y be in the dataset be one hot encoded. That is, if k is the true class of sample input x, then $y_i = 0, i \neq k$ and $y_k = 1$. Note that classification models output the probability of a particular sample being each class (and we pick the class with largest probability).

To derive a loss expression for such a model, we must first consider what P and Q are. Let P be the true output distribution, meaning that $P(k) = y_k$. Similarly, let Q be the model's output distribution, meaning that $Q(k) = f(x)_k$ where f is the model function and the notation $f(x)_k$ refers to the k-th element of $f(x)$. We sum over the KL divergence over $\mathcal{K}$ for every $(x, y) \in \mathcal{D}$

$$\begin{aligned}
\mathcal{L} &= \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} D_{KL}(P, Q) \\
&= \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}\sum_{k\in\mathcal{K}}[y_k\log y_k - y_k\log f(x)_k] \\
&= \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}\sum_{k\in\mathcal{K}} y_k\log y_k - \sum_{(x,y)\in\mathcal{D}}\sum_{k\in\mathcal{K}} y_k\log f(x)_k
\end{aligned}$$

Note that the first expression is constant for a dataset. Hence, we only need to minimize the second term. Thus, our loss term is

$$\mathcal{L} = -\frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}}\sum_{k\in\mathcal{K}} y_k\log f(x)_k$$

This is the loss function that is used for classification tasks. We will use cross-entropy loss in logistic regression as well.

3 Model definition

We define a classic logistic regression model to be

$$f_w(x) = \sigma(-wx) = \frac{1}{1+\exp(-wx)}$$

Generally, when we think of the classic logistic regression model, we think of binary classification. This means the probability of one class is $P(y)$ and the other class is $1 - P(y)$. Let $(x^{(i)}, y^{(i)}) \in \mathcal{D}$. The loss function is the cross-entropy loss averaged over each sample point in the batch size N.

$$\mathcal{L}(w) = -\frac{1}{N}\sum_{i=1}^{N}\Big[y^{(i)}\log\Big(f_w(x^{(i)})\Big) + (1-y^{(i)})\log\Big(1-f_w(x^{(i)})\Big)\Big]$$

Observe that

$$\frac{\partial}{\partial x}\sigma(x) = \sigma(x)(1-\sigma(x))$$

Now, we find the gradient descent update for $\mathcal{L}(w)$.

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial w} &= -\frac{1}{N}\sum_{i=1}^{N}\left[y^{(i)}\frac{f_w(x^{(i)})(1-f_w(x^{(i)})}{f_w(x^{(i)})}x^{(i)} + (1-y^{(i)})\frac{-f_w(x^{(i)})(1-f_w(x^{(i)}))}{1-f_w(x^{(i)})}x^{(i)}\right]\\ &= -\frac{1}{N}\sum_{i=1}^{N}\Big(y^{(i)}(1-f_w(x^{(i)})) - (1-y^{(i)})f_w(x^{(i)})\Big)x^{(i)}\\ &= -\frac{1}{N}\sum_{i=1}^{N}\Big(y^{(i)} - y^{(i)}f_w(x^{(i)}) - f_w(x^{(i)}) + y^{(i)}f_w(x^{(i)}\Big)x^{(i)}\\ &= \frac{1}{N}\sum_{i=1}^{N}\Big(f_w(x^{(i)}) - y^{(i)}\Big)x^{(i)}\end{aligned}$$

If w is a vector indexed by j, the update would be

$$\frac{\partial \mathcal{L}}{\partial w_j} = \frac{1}{N}\sum_{i=1}^{N}\Big(f_w(x^{(i)}) - y^{(i)}\Big)x_j^{(i)}$$

4 MLE Perspective

As suggested earlier, logistic regression makes the following assumption:

$$y_i | x_i \sim \text{Bern}(\sigma(-wx_i))$$

For any sample input x_i, the probability of y_i given x_i is a Bernoulli random variable with $p = \sigma(-wx_i)$. In other words, the logistic regression model defines the probability that x_i is either of the two classes (since we are considering binary logistic regression).

Remember that the likelihood is the probability of the data occurring within a model given a certain set of parameters. If $|\mathcal{D}| = N$ and w are the weights of the model, we write the likelihood as

$$P(x^{(1)}, \ldots, x^{(N)}, y^{(1)}, \ldots, y^{(N)} | w)$$

We seek to find w such that this probability is maximized. Note that a sample data point $(x^{(i)}, y^{(i)}) \in \mathcal{D}$ occurs independently; thus, the probability is also independent of each other.

$$\begin{aligned}
&\operatorname*{argmax}_w P(x^{(1)}, \dots, x^{(N)}, y^{(1)}, \dots, y^{(N)}|w) \\
&\operatorname*{argmax}_w \prod_{i=1}^{N} P(x^{(i)}, y^{(i)}|w) \\
&\operatorname*{argmax}_w \prod_{i=1}^{N} P(y^{(i)}|x^{(i)}, w)P(x^{(i)}|w) \\
&\operatorname*{argmax}_w \prod_{i=1}^{N} P(y^{(i)}|x^{(i)}, w)P(x^{(i)})
\end{aligned}$$

Note that $P(x^{(i)}$ does not depend on w. Thus, we do not need to consider in the maximization. As illustrated earlier, note that $P(y^{(i)}|x^{(i)}, w) = f_w(x^{(i)})$. Since this is binary classification, $y^{(i)} \in \{0, 1\}$. Note that

$$\begin{aligned}
P(y^{(i)} = 1 \,|\, x^{(i)}, w) &= f_w(x^{(i)}) \\
P(y^{(i)} = 0 \,|\, x^{(i)}, w) &= 1 - f_w(x^{(i)})
\end{aligned}$$

Together, $P(y^{(i)}|x^{(i)}, w) = f_w(x^{(i)})^{y^{(i)}}(1 - f_w(x^{(i)})^{(1-y^{(i)})}$. Thus, we want to maximize

$$\operatorname*{argmax}_w \prod_{i=1}^{N} f_w(x^{(i)})^{y^{(i)}}(1 - f_w(x^{(i)})^{(1-y^{(i)})}$$

Since it is difficult to maximize a product, we consider the log likelihood. Thus,

$$\begin{aligned}
&\operatorname*{argmax}_w \ \ln \prod_{i=1}^{N} f_w(x^{(i)})^{y^{(i)}}(1 - f_w(x^{(i)})^{(1-y^{(i)})} \\
&\operatorname*{argmax}_w \sum_{i=1}^{N} \Big[y^{(i)} f_w(x^{(i)} + (1 - y^{(i)})(1 - f_w(x^{(i)}))\Big]
\end{aligned}$$

Note that this was is the cross entropy loss function. Thus, the MLE confirms our earlier derivation of the logistic regression loss function.

5 MAP Perspective

Unlike MLE, Maximum a Posteriori (MAP) maximizes

$$P(x^{(1)}, \dots, x^{(N)}, y^{(1)}, \dots, y^{(N)}|w)P(w)$$

by assuming a prior probability $P(w)$ on the weights. The prior for the weights tends to be a Gaussian Distribution (as it usually does for most models):

$$w \sim \mathcal{N}(0, \sigma^2 I)$$

As we did with MLE, we find the log likelihood using the MAP estimator now. Notice that the first probability directly comes from the MLE estimate. More often than not, MAP estimates tend to follow their MLE counterparts with a prior belief on the initial values of the weights.

$$\begin{aligned}
&\operatorname*{argmax}_w \ln\Big(P(x^{(1)}, \dots, x^{(N)}, y^{(1)}, \dots, y^{(N)}|w)P(w)\Big) \\
&\operatorname*{argmax}_w \ln\Big(P(x^{(1)}, \dots, x^{(N)}, y^{(1)}, \dots, y^{(N)}|w)\Big) + \ln P(w)
\end{aligned}$$

Consider the second term. Note that since each w_i of vector w comes from a Gaussian Distribution, we know that

$$P(w) = \prod_i \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{-w_i^2}{2\sigma^2}\right)$$

Taking the natural log, we have

$$\begin{aligned}\ln P(w) &= \sum_i \ln\left(\frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{w_i^2}{2\sigma^2}\right)\right) \\ &= \sum_i \ln \frac{1}{\sqrt{2\pi}\sigma} + \sum_i -\frac{w_i^2}{2\sigma^2}\end{aligned}$$

Observe that the first summation has no w term, meaning we do not need to consider it in our maximization. Let w be a vector of size M. Altogether, we have the following:

$$\begin{aligned}&\underset{w}{\operatorname{argmax}} \ln\Big(P(x^{(1)}, \dots, x^{(N)}, y^{(1)}, \dots, y^{(N)}|w)\Big) + \ln P(w) \\ &\underset{w}{\operatorname{argmax}} \sum_{i=1}^{N} \Big[y^{(i)} f_w(x^{(i)} + (1-y^{(i)})(1-f_w(x^{(i)}))\Big] - \sum_{i=1}^{M} \frac{w_i^2}{2\sigma^2} \\ &\underset{w}{\operatorname{argmax}} \sum_{i=1}^{N} \Big[y^{(i)} f_w(x^{(i)} + (1-y^{(i)})(1-f_w(x^{(i)}))\Big] - \frac{1}{2\sigma^2} \sum_{i=1}^{M} w_i^2\end{aligned}$$

In practice, $\lambda = \frac{1}{\sigma^2}$. Hence, we reformulate the above as

$$\underset{w}{\operatorname{argmax}} \sum_{i=1}^{N} \Big[y^{(i)} f_w(x^{(i)} + (1-y^{(i)})(1-f_w(x^{(i)}))\Big] - \frac{\lambda}{2} \sum_{i=1}^{M} w_i^2$$

Furthermore, maximizing tends to be computationally harder; instead, we negate the above expression to find the minimum with respect to w. We also average the summation term as it makes more sense to compute the average loss (rather than the total loss across all data points). Thus, we have

$$\mathcal{L}(w) = -\frac{1}{N} \sum_{i=1}^{N} \Big[y^{(i)} f_w(x^{(i)} + (1-y^{(i)})(1-f_w(x^{(i)}))\Big] + \frac{\lambda}{2} \sum_{i=1}^{M} w_i^2$$

In practice, this is referred to as the cross-entropy loss (as mentioned earlier) for a batch size N with a L2 regularization term. Oftentimes, we may want a sparser representation in w, where a L1 regularization term would be stronger. Then, we would have a Laplace prior for the weights like so:

$$w \sim \text{Laplace}(0, b)$$

meaning that $P(w_i) = \frac{1}{2b} \exp\left(-\frac{|w_i|}{b}\right)$. This would give us the L1 regularization term.

Lastly, λ is inversely proportional to the variance of our prior belief since $\lambda = \frac{1}{\sigma^2}$. Furthermore, a smaller λ leads to overfitting while a larger λ leads to an underfitting of the data. Evidently, then, λ is an important hyperparameter of the model and must be carefully chosen for optimal results.

6 Multiclass Logistic Regression

Let K be the number of classes of the task. The idea behind multiclass logistic regression is to have K sets of weight vectors, and rather than use the sigmoid function to translate wx between 0 and 1 and interpret that as a probability distribution, we use the softmax function to build this distribution with K outcomes.

The softmax function is similar to the sigmoid function in that they both convert their input to some type of output distribution. However, unlike the sigmoid function, the softmax function will take in K values and interpret these K values as the probability that a given data point x can be a particular class.

Let W be a $D \times K$ matrix, where D is the size of each weight vector and K is the number of classes. Additionally, let x be an arbitrary data point in the dataset.

$$W = \begin{bmatrix} w_1^{(1)} & w_2^{(2)} & \dots & w_C^{(1)} \\ \vdots & \vdots & \vdots & \vdots \\ w_1^{(D)} & w_2^{(D)} & \dots & w_C^{(D)} \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 & \dots & x_D \end{bmatrix}$$

The multiclass logistic regression model is of the form:

$$f_W(x) = \begin{bmatrix} \frac{\exp\left(W_1^T x\right)}{\sum_{i=1}^K \exp\left(W_i^T x\right)} \\ \frac{\exp\left(W_2^T x\right)}{\sum_{i=1}^K \exp\left(W_i^T x\right)} \\ \vdots \\ \frac{\exp\left(W_{K-1}^T x\right)}{\sum_{i=1}^K \exp\left(W_i^T x\right)} \\ \frac{\exp\left(W_K^T x\right)}{\sum_{i=1}^K \exp\left(W_i^T x\right)} \end{bmatrix}$$

where W_i^T is the i-th row of W^T. The above column vector is the result of applying the softmax function on the vector

$$\begin{bmatrix} W_1^T x \\ W_2^T x \\ \vdots \\ W_{K-1}^T x \\ W_K^T x \end{bmatrix}$$

Softmax can be thought of as taking a vector of values and converting it to a probability distribution of K values by using the relative magnitude of each value in the original vector. A larger magnitude would result in a larger probability and vice versa. Furthermore, this also means that

$$P(y = C|x) = \frac{\exp\left(W_C^T x\right)}{\sum_{i=1}^K \exp\left(W_i^T x\right)}$$

Consider the log likelihood from earlier. Using the above, we have

$$\sum_{i=1}^N \ln P(y^{(i)}|x^{(i)}, W)$$

$$\sum_{i=1}^N \ln \frac{\exp\left(W_{y^{(i)}}^T x\right)}{\sum_{i=1}^K \exp\left(W_i^T x\right)}$$

$$\sum_{i=1}^N \left[W_{y^{(i)}}^T - \ln \sum_{i=1}^K \exp\left(W_i^T x\right) \right]$$