Logistic classification with cross-entropy loss

Julián D. Arias Londoño

August 3, 2020

1 Definition

Logistic regression (LG) is one the basic models studied in Statistics and Machine Learning to solve bi-class classification problems. The intuition behind this model is to find a polynomial function capable of splitting the feature space into two parts, i.e. the polynomial function plays the role of decision boundary between the two classes. The aim of the training algorithm is to find the model's parameters (polynomial's weights) such that each part of the space contains samples from only one of the classes as long as it is possible. Figure 1 shows a scatter plot of a two-class toy problem and the boundary function.

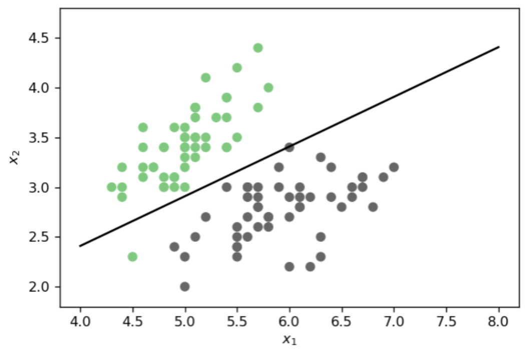

Figure 1: Scatter plot for a two-class problem and a linear decision function

Formally, given a dataset $\mathcal{D} = \{(\mathbf{x}_j, y_j)\}_{j=1}^N$, where $\mathbf{x}_j$ is a feature vector representing a sample j, and y_j is its corresponding target output, which can take one of two possible values $\{0, 1\}$, the aim is to build a function able to predict whether a new sample belongs to class 0 or 1. The LG model is composed of a polynomial function wrapped by a logistic function; it can be expressed as:

$$g(\mathbf{w}^T\mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})} \tag{1}$$

The logistic function is chosen because it is a derivable approximation of the sing function, and thus it can be used for gradient-based optimization methods. Figure 2 shows a graphic representation of the logistic function.

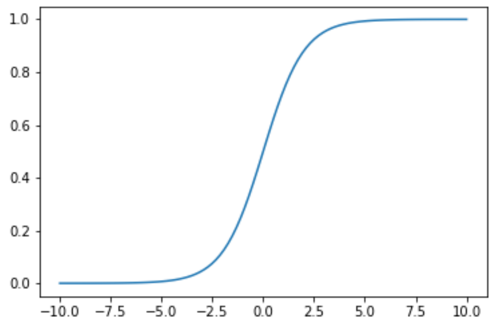

Figure 2: Logistic function

In order to train the model, we need to define a loss function that can be used for optimization purposes. Typically LG model uses the well-known cross-entropy function as cost function. Taking into account that logistic function provide a value in the interval $[0, 1]$, it can be interpreted as the probability of belonging to class 1. Therefore, we can use the Maximum Likelihood criterion applied to a Bernoulli distribution to come out with the function we want to optimize.

By assuming the samples are $i.i.d$ the log-likelihood function can be estimated as:

$$
\begin{aligned}
\arg\max_{\mathbf{w}} \mathcal{L} &= \log\left(\prod_{j=1}^{N} p_j^{y_j}(1-p_j)^{(1-y_j)}\right) \\
&= \log\left(\prod_{j=1}^{N} (g(\mathbf{w}^T\mathbf{x}_j))^{y_j}(1-g(\mathbf{w}^T\mathbf{x}_j))^{(1-y_j)}\right) \\
&= \sum_{j=1}^{N} \log\left((g(\mathbf{w}^T\mathbf{x}_j))^{y_j}\right) + \log\left((1-g(\mathbf{w}^T\mathbf{x}_j))^{(1-y_j)}\right) \\
&= \sum_{j=1}^{N} y_j \log\left(g(\mathbf{w}^T\mathbf{x}_j)\right) + (1-y_j)\log\left(1-g(\mathbf{w}^T\mathbf{x}_j)\right) \qquad (2)
\end{aligned}
$$

For the sake of numerical stability and for the use of a minimization algorithm instead of a maximization one, the final cross-entropy loss function is given by:

$$
J(\mathbf{w}) = -\frac{1}{N}\sum_{j=1}^{N} y_j \log\left(g(\mathbf{w}^T\mathbf{x}_j)\right) + (1-y_j)\log\left(1-g(\mathbf{w}^T\mathbf{x}_j)\right) \qquad (3)
$$

One of the most common optimization algorithms used for LG is the Gradient Descent which is based on applying iteratively the following rule:

$$\mathbf{w}(\tau) = \mathbf{w}(\tau - 1) - \eta \nabla J(\mathbf{w}) \tag{4}$$

In order to apply the former rule, the gradient of $J(\mathbf{w})$ must be estimated. The first step to get the gradient, is to estimate the derivative of the logistic function.

$$\begin{aligned}
\nabla_{\mathbf{w}} g(\mathbf{w}^T\mathbf{x}) &= \nabla_{\mathbf{w}} \left(\frac{1}{1 + \exp(-\mathbf{w}^T\mathbf{x})} \right) \\
&= \frac{\exp(-\mathbf{w}^T\mathbf{x})\mathbf{x}}{\left(1 + \exp(-\mathbf{w}^T\mathbf{x})\right)^2} \\
&= \frac{\exp(-\mathbf{w}^T\mathbf{x})}{1 + \exp(-\mathbf{w}^T\mathbf{x})} \frac{\mathbf{x}}{1 + \exp(-\mathbf{w}^T\mathbf{x})} \\
&= g(\mathbf{w}^T\mathbf{x})(1 - g(\mathbf{w}^T\mathbf{x}))\mathbf{x}
\end{aligned} \tag{5}$$

Based on the former result it is quite easy to estimate the gradient of the cross-entropy function as:

$$\begin{aligned}
\nabla_{\mathbf{w}} J(\mathbf{w}) &= \nabla_{\mathbf{w}} \left(-\frac{1}{N} \sum_{j=1}^{N} y_j \log\left(g(\mathbf{w}^T\mathbf{x}_j)\right) - (1 - y_j) \log\left(1 - g(\mathbf{w}^T\mathbf{x}_j)\right) \right) \\
&= -\frac{1}{N} \sum_{j=1}^{N} y_j \frac{\nabla_{\mathbf{w}} g(\mathbf{w}^T\mathbf{x}_j)}{g(\mathbf{w}^T\mathbf{x}_j)} + (1 - y_i) \frac{-\nabla_{\mathbf{w}} g(\mathbf{w}^T\mathbf{x}_j)}{1 - g(\mathbf{w}^T\mathbf{x}_j)}
\end{aligned} \tag{6}$$

By replacing Eq. (5) into Eq. (6) we obtain:

$$\begin{aligned}
\nabla_{\mathbf{w}} J(\mathbf{w}) &= -\frac{1}{N} \sum_{j=1}^{N} y_j \left(1 - g(\mathbf{w}^T\mathbf{x}_j)\right) \mathbf{x}_j - (1 - y_i) g(\mathbf{w}^T\mathbf{x}_j)\mathbf{x}_j \\
&= \frac{1}{N} \sum_{j=1}^{N} \left(g(\mathbf{w}^T\mathbf{x}_j) - y_j\right) \mathbf{x}_j
\end{aligned} \tag{7}$$

The expression in Eq. (7) is similar to the one that can be obtained for multiple regression using the Least Square Error cost function, that in turn can be derived from the maximum likelihood criterion but applied to a normal distribution instead [1].

References

[1] C. M. Bishop, *Pattern recognition and machine learning.* Springer, 2006.