Aldo Hernández

LOGISTIC REGRESSION WITH PYTHON

Abstract

Keywords ???

1. Introduction

Linear functions can be used not only for regression, but also for **classification**. In this case, the line—or surface—is called a *decision boundary*—also known as linear separator—and the data that admits this separator are called *linearly separable*. [2] In order to use these functions for classification, the first approach was to use the *threshold* function as the classification hypothesis

$$h_w(x) = Threshold(w \cdot x) \text{ where } Threshold(z) = 1 \text{ if } z \geq 0 \text{ and 0 otherwise}$$

$$w_i \leftarrow w_i + \alpha(y - h_w(x)) \times x_i \tag{1}$$

But this method has some pretty big disadvantages: even though it is *guaranteed* to converge with linearly separable data, it may take too many steps to do so, also, data may not be linearly separable always, so the algorithm (weight update rule shown in Equation 1) will fail to converge for a fixed learning rate α, unless it decays as $O(1/t)$ where t is the iteration number, then it is shown to converge to a minimum-error solution. [2]

Also, the hypothesis is not differentiable and is a discontinuous function of its inputs and weights, which makes learning with the **perceptron rule** shown in Equation 1 pretty complicated. Besides, this function will always return a confident prediction of 1 or 0, even if examples are too close to the boundary; in a lot of situations, this is not optimal since we will need more gradated predictions. [2]

It's because of these issues that *logistic regression* surges as a better alternative. We define a function—known as logistic or sigmoid—as follows

$$h_w(x) = Logistic(w \cdot x) = \frac{1}{1 + e^{-w \cdot x}}$$

This hypothesis gives a probability of 0.5 for every input at the center, and approaches 0 or 1 as we move away from the boundary [2]. The process of fitting the weights to this model to minimize the loss on a dataset is called **logistic regression** [2], there is no easy closed-form solution to find the optimal value of w but gradient descent computation is straightforward. Partially differentiating the L_2 function we get the following

$$\frac{\partial}{\partial w_i}(y - h_w(x))^2 = -2(y - h_w(x)) \times h_w(x)(1 - h_w(x)) \times x_i$$

Thus, the weight update for minimizing the loss is

$$w_i \leftarrow w_i + \alpha(y - h_w(x)) \times h_w(x)(1 - h_w(x)) \times x_i$$

This process is a **supervised algorithm** that can classify data into two states—binary, as shown earlier—or multiple tags [1]. Some common use cases are:

- Classify mail into spam or not.
- Classify tumors into benign or malignant.
- Classify the content of an article.

2. Methodology

2.1. Before typing code

First of all, we need to download this .csv file [1]

2.2. Data analysis

2.3. Creating the model

2.4. Model validation

2.5. Model visualization

3. Results

4. Conclusions

[1]

References

[1] Bagnato J.I.: *Aprende Machine Learning en Español*. Leanpub, 1.5st ed., 2020.

[2] Russell S., Norvig P.: *Artificial Intelligence: A Modern Approach*. Prentice Hall, 3rd ed., 2010.

Affiliations

Aldo Hernández
Universidad Autónoma de Nuevo León, San Nicolás de los Garza,
aldo.hernandezt@uanl.edu.mx

Received: ???
Revised: ???
Accepted: ???