Performance of Single Layer perceptron on fertility dataset

Shrikant patro,

School of Computer Science and Engineering, VIT Chennai, Tamil Nadu, India 600127

Email:shrikantjagannath.2018@vitstudent.ac.in

Abstract – The classification of the data point into binary or multiple classes is the very common task in machine learning. Perceptron learning algorithm and pocket algorithm are basic classification algorithm. This paper will provide complete analysis of Perceptron learning in classification task. The dataset used here is woman fertility dataset from UCI repository. Towards end, we will check the performance achieved by the logistic regression over the perceptron algorithm on the csv file belonging to the woman fertility dataset containing records of patient donated blood and other detail. This model will not only predict the person can be possible blood donor or not but based on input attribute it will give clear classification as donor or not.

Introduction

This Paper introduces the one of the first algorithmically described machine learning algorithms for classification, the *perceptron*. We will start by implementing a perceptron step by step in Python and training it to classify different flower species in the Iris dataset. This will help us to understand the concept of machine learning algorithms for classification and algorithm is implemented in python in order to see performance perceptron learning algorithm over Woman Fertility dataset.

Trying to understand how the biological brain works to design artificial intelligence, Warren McCullock and Walter Pitts published the first concept of a simplified brain cell, the so-called McCullock-Pitts (MCP) neuron, in 1943 (W. S. McCulloch and W. Pitts. *A* Logical Calculus of the Ideas Immanent in Nervous Activity. The bulletin of mathematical biophysics, 5(4):115–133, 1943). Neurons are interconnected nerve cells in the brain that are involved in the processing and transmitting of chemical and electrical signals, which is illustrated in the following figure 1

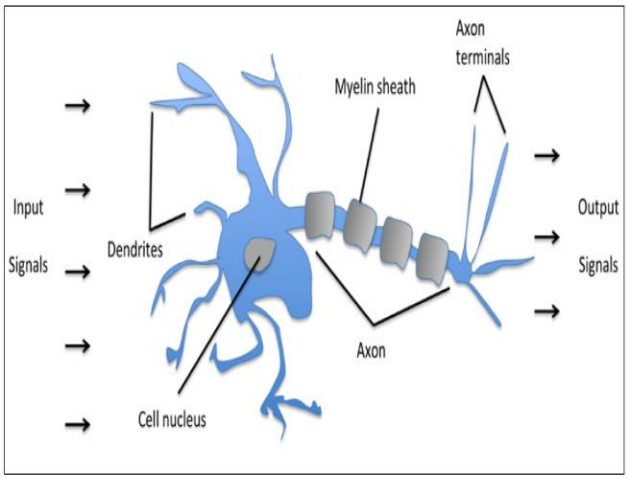

Fig 1

McCullock and Pitts described such a nerve cell as a simple logic gate with binary outputs; multiple signals arrive at the dendrites, are then integrated into the cell body, and, if the accumulated signal exceeds a certain threshold, an output signal is generated that will be passed on by the axon. Only a few years later, Frank Rosenblatt published the first concept of the perceptron learning rule

based on the MCP neuron model (F. Rosenblatt, The Perceptron, a Perceiving and Recognizing Automaton. Cornell Aeronautical Laboratory, 1957). With his perceptron rule, Rosenblatt proposed an algorithm that would automatically learn the optimal weight coefficients that are then multiplied with the input features in order to make the decision of whether a neuron fires or not. In the context of supervised learning and classification, such an algorithm could then be used to predict if a sample belonged to one class or the other.

Methodology

The problem as a binary classification task where we refer to our two classes as 1 (positive class) and -1 (negative class) for simplicity. We can then define an activation function () z_ that takes a linear combination of certain input values ***x*** and a corresponding weight vector ***w*** , where *z* is the so-called net input

$$\boldsymbol{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}, \quad \boldsymbol{x} = \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix}$$

Now, if the activation of a particular sample () *i****x***, that is, the output of () *z*, is greater than a defined threshold _ , we predict class 1 and class -1, otherwise, in the perceptron algorithm, the activation function () _ _is a simple unit step function, which is sometimes also called the Heaviside step function

$$\phi(z) = \begin{cases} 1 \ \text{if} \ z \geq \theta \\ -1 \ \text{otherwise} \end{cases}$$

For simplicity, we can bring the threshold (theta) to the left side of the equation and define a weight-zero as w_0= - (theta) and x_0=1, 1, so that we write **z** in a more compact form

$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \boldsymbol{w}^T \boldsymbol{x} \ \text{and} \ \phi(z) = \begin{cases} 1 \ \text{if} \ z \geq \theta \\ -1 \ \text{otherwise}. \end{cases}$$

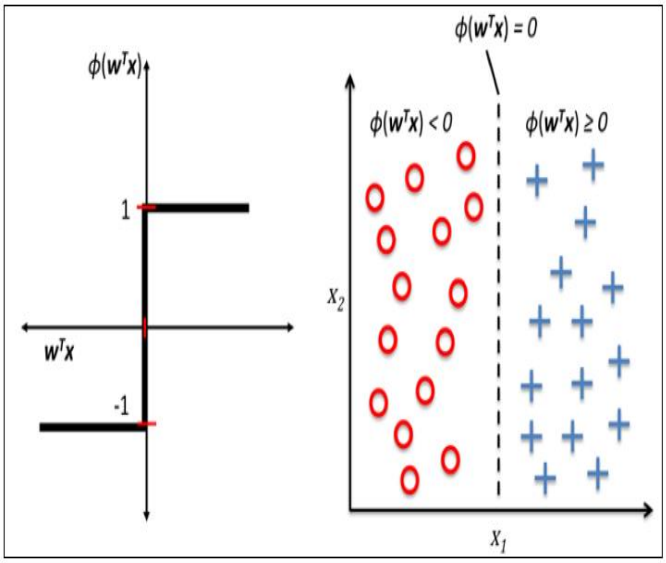

Fig 2

The output value is the class label predicted by the unit step function that we defined earlier, and the simultaneous update of each weight *w (j)* in the weight vector w can be more formally written as:

$$w_j := w_j + \Delta w_j .$$

$$\Delta w_j = \eta\left(y^{(i)} - \hat{y}^{(i)}\right)x_j^{(i)}$$

The weight is regularly updated using the above formulation.

Database- Woman's Fertility dataset

There are different CSV file for different phenomenon associated with woman fertility dataset. This dataset is collected by WHO with the help of 100 volunteers a semen sample analyzed according to WHO 2010 criteria. Sperm concentration are related to socio-demographic data, environmental factors, health status, and life habits. Dataset characteristics: Multivariate, Attribute characteristics: Real, Associated task: Classification, Regression, Number of instances: 100, Number of attribute: 10, Missing values: NA, Area: Life, Date Donated: 2013-01-17 and number of web hits: 142847.

The further description is specifically regarding the "Blood Transfusion Service Center". This include field like Recency (Months), Frequency (Times), Monetary (c.c. blood), Time (months) and donated. Donated is binary attributed stating in terms of 1 and 0 to represent whether donated or not.

Algorithm

In order to find the regression line we must follow the following steps:

Step 1:
Initialize the weights to 0 or small random numbers.

Step 2:
For each training sample $x^{(i)}$
perform the following steps:

Step 2.1 : Compute the output value $\hat{y}$
.

Step 4 :. Update the weights .

Conclusion:

References:

[1] Python Machine Learning – Sebastian Raschka.