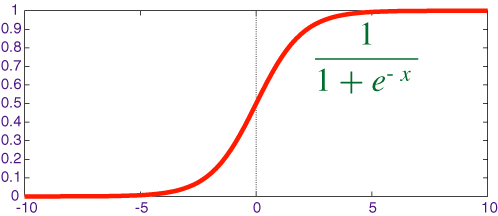
# Logistics Regression an overview

Logistics Regression is majorly based on sigmoid function. Linear regression tries to define a boundary between the entities of your dataset and based on that line predicts the value of y for given value of x. But lets suppose you need to predict something that only can have two answers yes or no. Now the problem here is that for x in the domain of the graph will surely give a appreciable result but it will fail if x is given such that its out of domain of the graph like a very high value of x or negative values of x which is not acceptable since our answer can only lie between 0 or 1. Thats how logistics regression was born.

Logistics regression revolves around the sigmoid function to get the values between 0 and 1. Since Decision making only has value either 0 or one using sigmoid is no brainer. Sigmoid function is given as

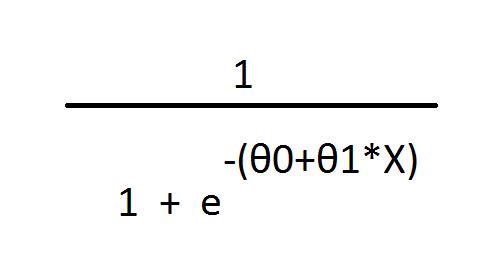
1 / 1 + e^-x

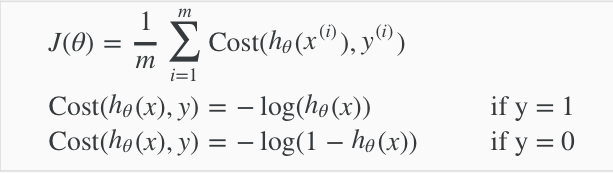
Which when plotted looks like this:

Thus if we take the hypothesis of linear regression and plug it in the sigmoid function we get now make sure that out output always remain between 0 and 1.

The Cost Function

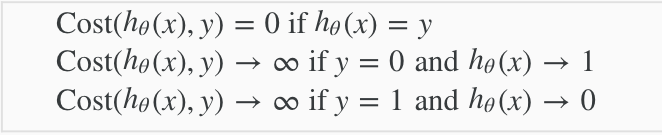
Now lets discuss about the cost function. The hypothesis for logistics regression is obtained by just substituting the hypothesis of linear regression in the sigmoid function as:

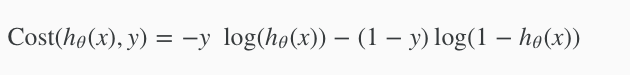
hypothesis = 1 / 1 + e^ ( θ0+θ1\*x )

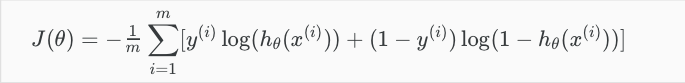
Thus the cost function of Logistics regression is obtained by taking the log of the cost function used in linear regression. It is thus stated as:

Where hθ(x) = θ0 + θ1 \* x

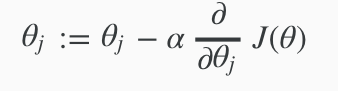
Thus now if out cost function tends to infinity in both cases where either hθ(x) tends to zero or y = 1 and y = 0 and hθ(x) tends to 1.



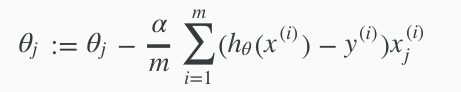
Thus our cost function can thus be combined to form a single expression as:

Hence our final cost function can be represented as:

Optimisations:

Now since we have our cost function we need to optimise our model using gradient descent. The standard equation for gradient descent is given as:

Now working out the partial derivative for our logistics regression cost function we obtain:



Thus by simultaneously updating the garden descent for θ0 and θ1 we ensure that the cost function reaches a minima.

Decision Boundary:

Decision boundary is that threshold value where we decide whether the prediction is true or false or 0 or 1 , which is designated for logistics regression since it predicts only two cases true or false. Now looking back at the graph of the sigmoid function, we notice that it cuts the y axis at 0.5 ; Thus we can define the threshold value at 0.5 like:

1. If value is >= 0.5 then classify it as 1
2. If value is < 0.5 then classify it as 0

Why Logistics is better than linear regression for classification tasks :

Logistics regression was built for classification tasks and in every way works better than linear regression. The major problems that arise when using linear regression for classification tasks is the decision boundary. Classification tasks only predict values as either 0 or 1 but linear regression will predict values based on the linear decision boundary which is not guaranteed to give a output bounded between 0 and 1; This is no brainer since linear regression predicts values on a continuous basis and on the other hand logistics regression has only few outputs which are guaranteed to be bounded between 0 and 1. Thus logistics regression works flawlessly for classification tasks.

Having seen the theory lets move to the coding part. We shall be coding logistic regression from scratch and also as the thumb rule says never reinvent the wheel, we also shall be looking into how to use logistic model directly on data using sklearn. ( Its no brainer to make sure you have python (v 3.x) , sklearn , numpy, matplotlib and pandas libraries installed ).

Logistic Regression Using sklearn:

Please refer the Jupyter notebook.

Logistic Regression from scratch:

While using sklearn is always recommended its also important to know how to code it from scratch to get a better understanding of how the model is actually trained. We shall be starting by finding the cost function and then minimising it using gradient descent. Then we shall be finding the required parameters ( theta ) to find the values of theta required by our model and then we shall be using the test data to find the accuracy of our model. It should match with the accuracy obtained using the standard method since we are not touching the dataset.