Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**

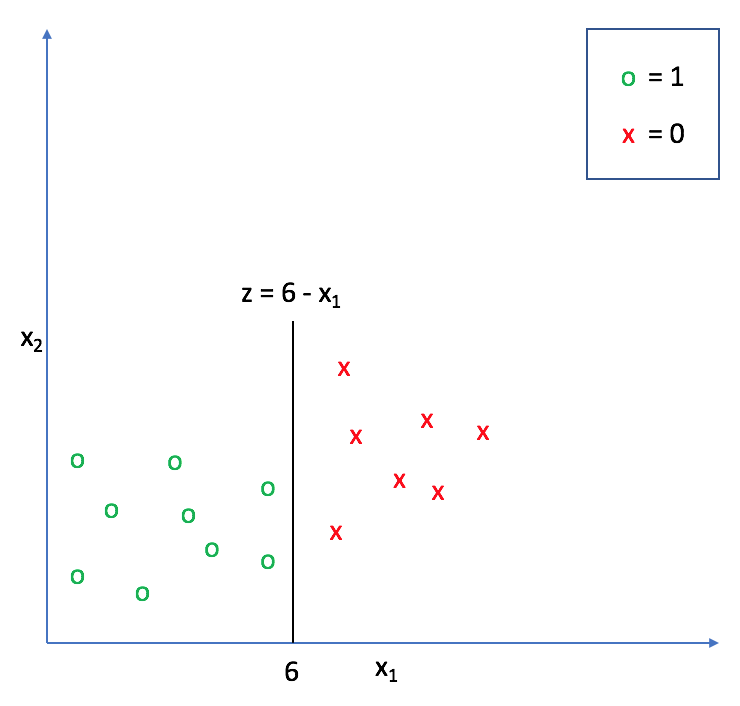
We can call a Logistic Regression a Linear Regression model but the Logistic Regression uses a more complex cost function, this cost function can be defined as the ‘**Sigmoid function**’ or also known as the ‘logistic function’ instead of a linear function.

The goal of logistic regression, as with any classifier, is to figure out some way to split the data to allow for an accurate prediction of a given observation's class using the information present in the features. (For instance, if we were examining the Iris flower dataset, our classifier would figure out some method to split the data based on the following: sepal length, sepal width, petal length, petal width.) In the case of a generic two-dimensional example, the split might look something like this.

**The decision boundary**

Let's suppose we define a line that is equal to zero along this decision boundary. This way, all of the points on one side of the line take on positive values and all of the points on the other side take on negative values.

For example, in the following graph, z=6−x1z=6−x1 represents a decision boundary for which any values of x1>6x1>6 will return a negative value for zz and any values of x1<6x1<6 will return a positive value for zz.



We can extend this decision boundary representation as any linear model, with or without additional polynomial features. (If this model representation doesn't look familiar, check out my post on [linear regression](https://www.jeremyjordan.me/linear-regression/)).

z(x)=θ0+θ1x1+θ2x2+...+θnxn=θTxz(x)=θ0+θ1x1+θ2x2+...+θnxn=θTx

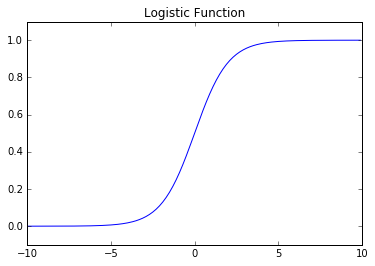
Adding [polynomial features](https://www.jeremyjordan.me/polynomial-regression/) allows us to construct nonlinear splits in the data in order to draw a more accurate decision boundary.

As we continue, think of z(x)z(x) as a measure of how far a certain data point exists from the decision boundary.

**The logistic function**

Next, let's take a look at the logistic function.

g(z)=11+e−zg(z)=11+e−z



As you can see, this function is asymptotically bounded between 0 and 1. Further, for very positive inputs our output will be close to 1 and for very negative inputs our output will be close to 0. This will essentially allow us to translate the value we obtain from zz into a prediction of the proper class. Inputs that are close to zero (and thus, near the decision boundary) signify that we don't have a confident prediction of 0 or 1 for the observation.

At the decision boundary z=0z=0, the function g(z)=0.5g(z)=0.5. We'll use this value as a cutoff establishing the prediction criterion:

hθ(xi)≥0.5→ypred=1hθ(xi)≥0.5→ypred=1

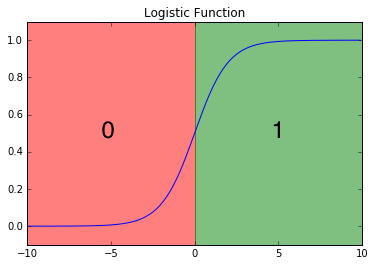
hθ(xi)<0.5→ypred=0hθ(xi)<0.5→ypred=0

where ypredypred denotes the predicted class of an observation, xixi, and hθ(x)hθ(x) represents the functional composition, hθ(x)=g(z(x))hθ(x)=g(z(x)).

More concretely, we can write our model as:

hθ(x)=11+e−θTxhθ(x)=11+e−θTx

Thus, the output of this function represents a binary prediction for the input observation's class.



Another way to interpret this output is to view it in terms of a probabilistic prediction of the true class. In other words, hθ(x)hθ(x) represents the estimated probability that yi=1yi=1 for a given input, xixi.

hθ(xi)=P(yi=1|xi;θ)hθ(xi)=P(yi=1|xi;θ)

Because the class can only take on values of 0 or 1, we can also write this in terms of the probability that yi=0yi=0 for a given input, xixi.

P(yi=0|xi;θ)=1−P(yi=1|xi;θ)P(yi=0|xi;θ)=1−P(yi=1|xi;θ)

For example, if hθ(x)=0.85hθ(x)=0.85 then we can assert that there is an 85% probability that yi=1yi=1 and a 15% probability that yi=0yi=0. This is useful as we can not only predict the class of an observation, but we can quantify the **certainty** of such prediction. In essence, the further a point is from the decision boundary, the more certain we are about the decision.

**The cost function**

Next, we need to establish a cost function which can grade how well our model is performing according to the training data. This cost function, J(θ)J(θ), can be considered to be a summation of individual "grades" for each classification prediction in our training set, comparing hθ(x)hθ(x) with the true class yiyi. We want the cost function to be large for incorrect classifications and small for correct ones so that we can minimize J(θ)J(θ) to find the optimal parameters.

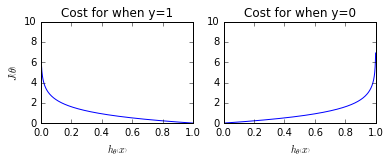
J(θ)=1m∑i=1mCost(hθ(xi),yi)J(θ)=1m∑i=1mCost(hθ(xi),yi)

In linear regression, we used the squared error as our grading mechanism. Unfortunately for logistic regression, such a cost function produces a nonconvex space that is not ideal for optimization. There will exist many local optima on which our optimization algorithm might prematurely converge before finding the true minimum.

Using the Maximum Likelihood Estimator from statistics, we can obtain the following cost function which produces a convex space friendly for optimization. This function is known as the **binary cross-entropy loss**.

Cost(hθ(x),y)={−log(hθ(x)) if y=1−log(1−hθ(x))if y=0}Cost(hθ(x),y)={−log(hθ(x)) if y=1−log(1−hθ(x))if y=0}

These cost functions return high costs for incorrect predictions.



More succinctly, we can write this as

Cost(hθ(x),y)=−ylog(hθ(x))−(1−y)log(1−hθ(x))Cost(hθ(x),y)=−ylog⁡(hθ(x))−(1−y)log⁡(1−hθ(x))

since the second term will be zero when y=1y=1 and the first term will be zero when y=0y=0. Substituting this cost into our overall cost function we obtain:

J(θ)=1m∑i=1m−yilog(hθ(xi))−(1−yi)log(1−hθ(xi))