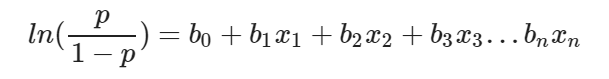
**Lab 3**

**To implement logistic regression**

In statistics logistic regression is used to model the probability of a certain class or event. Logistic regression is similar to linear regression because both of these involve estimating the values of parameters used in the prediction equation based on the given training data. Linear regression predicts the value of some continuous, dependent variable. Whereas logistic regression predicts the probability of an event or class that is dependent on other factors. Thus, the output of logistic regression always lies between 0 and 1. Because of this property it is commonly used for classification purpose.

# Logistic Model

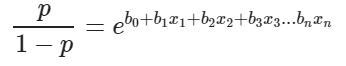
Consider a model with features x1, x2, x3 … xn. Let the binary output be denoted by Y, that can take the values 0 or 1. Let p be the probability of Y = 1, we can denote it as p = P(Y=1).  
The mathematical relationship between these variables can be denoted as:



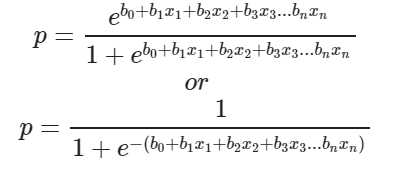
Here the term p/(1−p) is known as the odds and denotes the likelihood of the event taking place. Thus ln(p/(1−p)) is known as the log odds and is simply used to map the probability that lies between 0 and 1 to a range between (−∞,+∞). The terms b0, b1, b2… are parameters (or weights) that we will estimate during training.

So we simplify the equation to obtain the value of p:

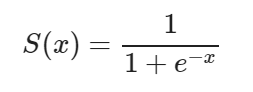
1. The log term *ln* on the LHS can be removed by raising the RHS as a power of *e*:

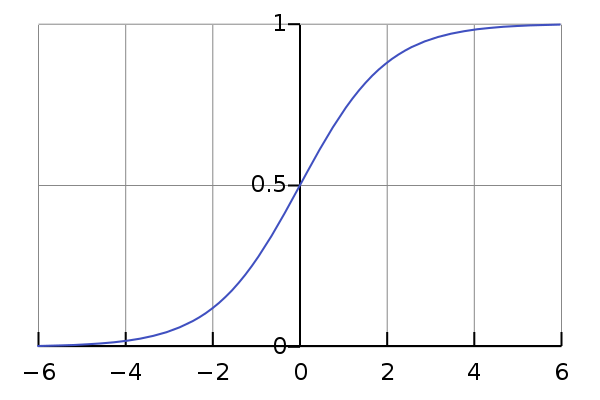


2. Now we can easily simplify to obtain the value of *p* :



This actually turns out to be the equation of the *Sigmoid Function* which is widely used in other machine learning applications. The *Sigmoid Function* is given by:



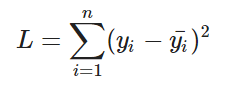


Now we will be using the above derived equation to make our predictions. Before that we will train our model to obtain the values of our parameters *b0, b1, b2…* that result in least error. This is where the error or loss function comes in.

**Loss Function**

The loss is basically the error in our predicted value. In other words it is a difference between our predicted value and the actual value. We will be using the L2 Loss Function to calculate the error. Theoretically you can use any function to calculate the error. This function can be broken down as:

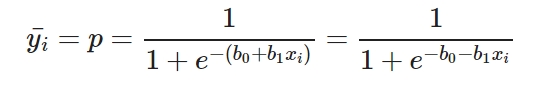
1. Let the actual value be yᵢ. Let the value predicted using our model be denoted as ȳᵢ. Find the difference between the actual and predicted value.
2. Square this difference.
3. Find the sum across all the values in training data.



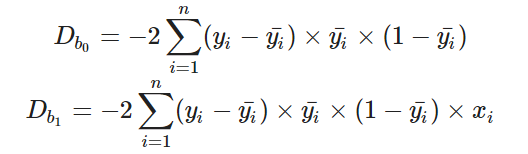
Now that we have the error, we need to update the values of our parameters to minimize this error. This is where the “learning” actually happens, since our model is updating itself based on it’s previous output to obtain a more accurate output in the next step. Hence with each iteration our model becomes more and more accurate. We will be using the Gradient Descent Algorithm to estimate our parameters. Another commonly used algorithm is the Maximum Likelihood Estimation.

# The Gradient Descent Algorithm

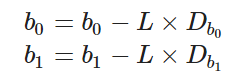
You might know that the partial derivative of a function at it’s minimum value is equal to 0. So gradient descent basically uses this concept to estimate the parameters or weights of our model by minimizing the loss function. For simplicity, assume that our output depends only on a single feature x. So we can rewrite our equation as:

(1)

Thus we need to estimate the values of weights b0 and b1 using our given training data.

1. Initially let b0=0 and b1=0. Let L be the learning rate. The learning rate controls by how much the values of b0 and b1 are updated at each step in the learning process. Here let L=0.001.
2. Calculate the partial derivative with respect to b0 and b1. The value of the partial derivative will tell us how far the loss function is from it’s minimum value. It is a measure of how much our weights need to be updated to attain minimum or ideally 0 error. In case you have more than one feature, you need to calculate the partial derivative for each weight b0, b1 … bn where n is the number of features. (2)

3. Next we update the values of b0 and b1:

(3)

4. We repeat this process until our loss function is a very small value or ideally reaches 0 (meaning no errors and 100% accuracy). The number of times we repeat this learning process is known as iterations or epochs.

**Steps:**

1. Read and visualize the database

2. Divide the data to training set and test set (80:20)

3. Perform data normalization

4. Compute the method to make predictions using equation (1)

5. Evaluate the method to train the model using equations (2) and (3). Obtain b0 and b1

6. Train the model using training data

7. Make the predictions

8. Calculate accuracy