CSE 574 – Logistic Regression

Rohit Lalchand Vishwakarma
Department of Computer Science
University at Buffalo – State University of New York
Buffalo, NY 14214
rohitlal@buffalo.edu

Abstract

We are developing a Logistic Regression model to predict the diagnosis of the breast cancer as malignant and benign based on the data of Breast Cancer Wisconsin (Diagnostic) Data Set. The data modelling is done in the python using the Jupyter notebook. Gradient Descent function is used to test the model whether it is not overfitted and we get the best accuracy with defined number of iterations and learning rate constant. Our model will help to predict the diagnosis of breast cancer based on the given feature parameters.

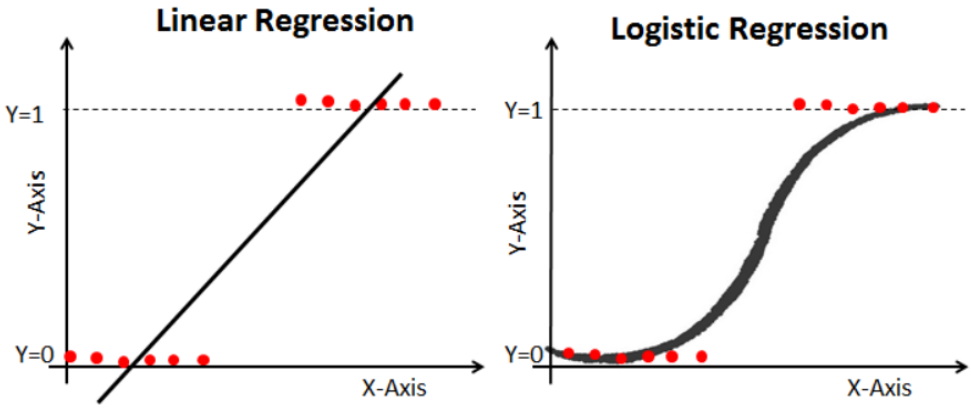
1 Introduction

"Hiding within those mounds of data is knowledge that could change the life of a patient or change the world."

– **Atul Butte**, Stanford University

1.1 About the Classification Model – Logistic Regression

The Term "Logistic regression" comes from two parameters the first is logistic which comes from the term **logit** as it uses logit function and second is regression as it uses the idea of Linear regression for prediction.



Classification problem is the problem when independent variables are continuous and dependent variable is in categorical form i.e. in classes like positive class (1) and negative class (0). For example, consider categorizing the mail as spam or not spam, or to categorize the tumor as malignant or benign which we are using in our model.

1.2 Using Logistic regression in our model

The data of Breast Cancer Wisconsin (Diagnostic) Data Set has the features which are pre-computed from images of a needle fine needle aspirate (FNA) of a breast mass. So here we are classifying the suspected FNA cells to Benign (Class 0) or Malignant (class 1) based on the modelling of the feature variables which are trained in the logistic model to predict the class 0 and class 1 accordingly.

2 Dataset

The data of Breast Cancer Wisconsin (Diagnostic) Data Set consists The dataset contains 569 instances with 32 attributes having 1st two columns with the id and Class 0/1(Benign/Malignant) and has 10 important characteristics : 1) radius (mean of distances from center to points on the perimeter) 2) texture (standard deviation of gray-scale values), 3) perimeter 4) area 5) smoothness (local variation in radius lengths), 6) compactness 7) concavity (severity of concave portions of the contour), 8) concave points (number of concave portions of the contour) 9) symmetry, 10) fractal dimension (“coastline approximation” - 1) which are further divided based on The mean, standard error, and \worst" or largest (mean of the three largest values) resulting in the total 30 features.

In our model the attributes header are provided based on its characteristics as 'id','diagnosis','radius_mean','texture_mean','perimeter_mean','area_mean','smoothness_mean','compactness_mean','concavity_mean','concave.points_mean','symmetry_mean','fractal_dimension_mean','radius_se','texture_se','perimeter_se','area_se','smoothness_se','compactness_se','concavity_se','concave.points_se','symmetry_se','fractal_dimension_se','radius_worst','texture_worst','perimeter_worst','area_worst','smoothness_worst','compactness_worst','concavity_worst','concave.points_worst','symmetry_worst','fractal_dimension_worst'.

3 Preprocessing

If we wish to build an impeccable predictive model, neither any programming language nor any machine learning algorithm can award it to you unless you perform data exploration. We need preprocessing so that our data behaves properly in our model i.e. we don’t need any data that gives us the biased results. We need to treat the outliers, check if the data is normal. Our data is already treated for the outliers and missing data. So we don’t have to do any data exploration here but we have to divide our data in such a way that major part of data is used for training the model and the second part is used for validating the training model and the 3rd part of the data is needed the predict the accuracy based on the model.

Here we are dividing the data into 80% for training data, 10% for validation data and another 10% for test data. We use the sklearn library of python to split the data which has the function train_test_split.

```
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size=0.80,test_size=0.20 , random_state=100)
x_test, x_val, y_test, y_val = train_test_split(x_test, y_test, train_size=0.50, random_state=100) #splitting
```

After the partition we need to scale the data between 0 and 1 for all the feature variables. This is important step because we want to predict the data for 0 and 1 so having values far larger than 1 will not make a good model and will not give the desired result.There are many functions in python to normalize the data but here we are using MinMaxScaler() method of the sklearn library. Once the data is normalized we can start with our training of the model

```
scaledata=preprocessing.MinMaxScaler()
X_train = scaledata.fit_transform(x_train)
X_test = scaledata.fit_transform(x_test)
X_val = scaledata.fit_transform(x_val)
```

4 Architecture

Logistic Regression uses Sigmoid Function which is given in the following image. The following equation generates values between 0 to 1 and here the z is the hypothesis which takes the value of weights, feature variable data and bias.

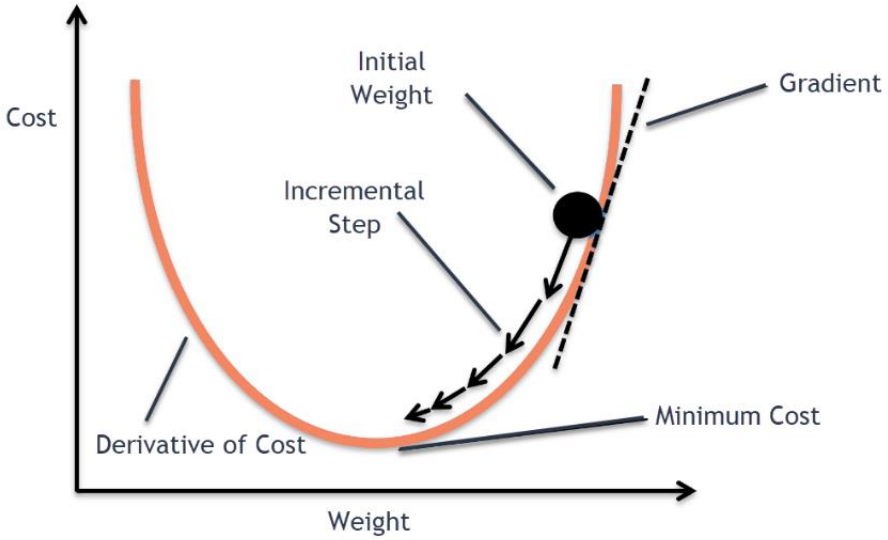
$$h(x) = g(\theta^T x)$$

$$z = \theta^T x$$

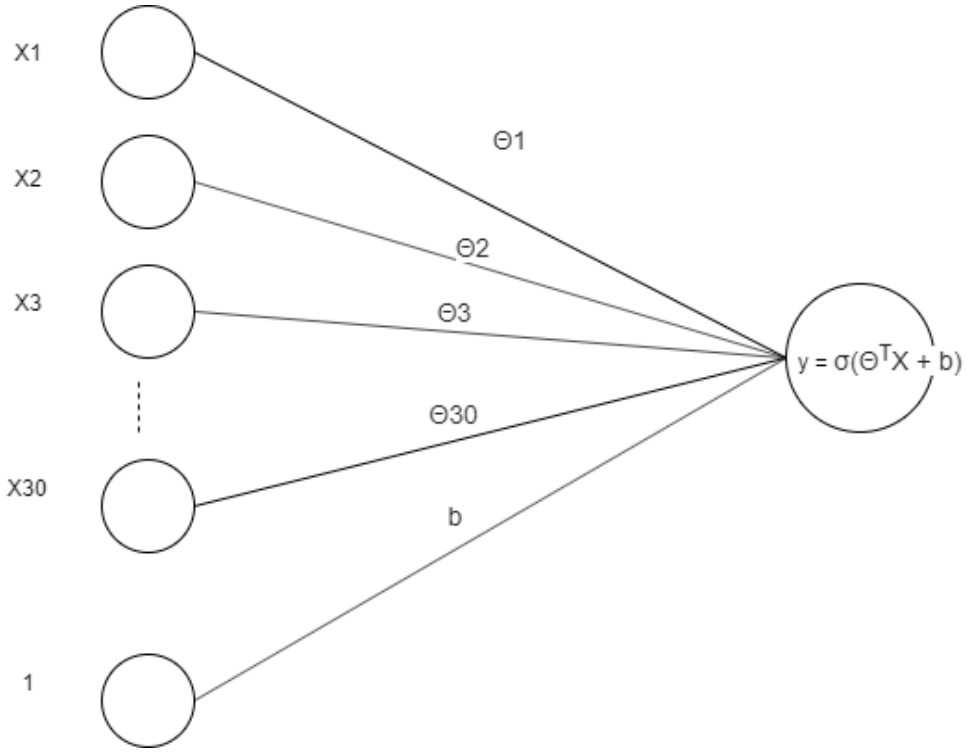
$$g(z) = \frac{1}{1+e^{-z}}$$

$$g(\theta^T x) = \frac{1}{1+e^{-(\theta^T x)}}$$

In our model we have created a gradient descent function which help us to get the weights and bias and we used this in the sigmoid function above to predict our response variable



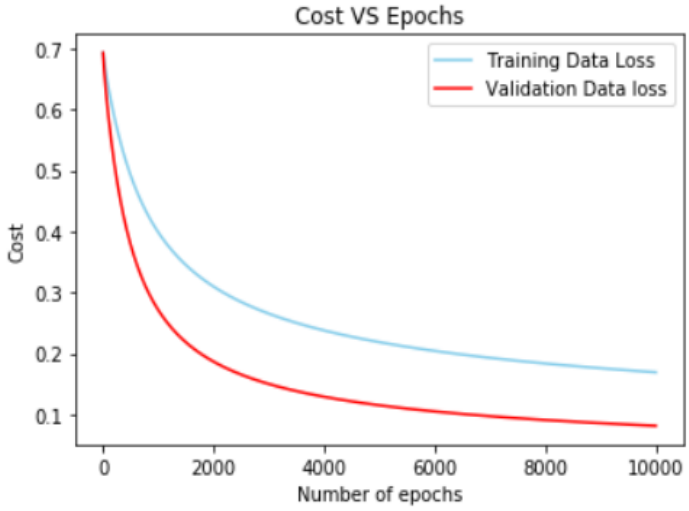
So ultimately we get our Logistic Function as per the following diagram where x1 is our feature data theta1 is our weights and y is our predicted function



5 Results

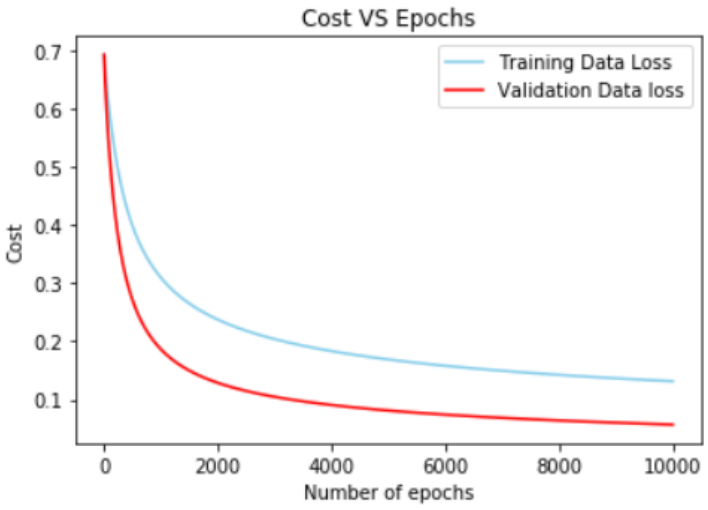
After the training with the number of iterations (epochs) and learning rate increasing as 0.02,0.04,0.06 we get weights(w) and bias (b) at each epochs value. The following is the corresponding result depicting for training data and validation data cost function with the number of epochs.(Epoch =10000) also the accuracy of the train and validation data has to be depicted using confusion matrix.

a) Learning Rate = 0.02



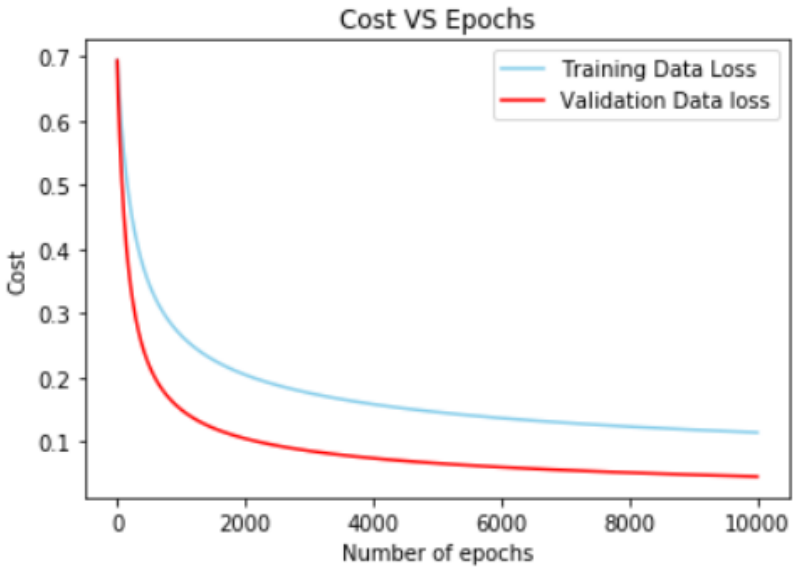

```
 ------- Training Data -------

[[290   2]
 [ 16 147]]

Accuracy
 : 0.9604395604395605

Precision
 : 0.9477124183006536

Recall
 : 0.9931506849315068

 ------- Validation Data -------

[[33  0]
 [ 1 23]]

Accuracy
 : 0.9824561403508771

Precision
 : 0.9705882352941176

Recall
 : 1.0
```

b) Learning Rate = 0.04




```
 ------- Training Data -------

[[290   2]
 [ 10 153]]

Accuracy
 : 0.9736263736263736

Precision
 : 0.9666666666666667

Recall
 : 0.9931506849315068

 ------- Validation Data -------

[[33  0]
 [ 1 23]]

Accuracy
 : 0.9824561403508771

Precision
 : 0.9705882352941176

Recall
 : 1.0
```

c) Learning rate =0.06




```
 ------- Training Data -------

[[291   1]
 [  8 155]]

Accuracy
 : 0.9802197802197802

Precision
 : 0.9732441471571907

Recall
 : 0.9965753424657534

 ------- Validation Data -------

[[33  0]
 [ 1 23]]

Accuracy
 : 0.9824561403508771

Precision
 : 0.9705882352941176

Recall
 : 1.0
```

Now we will use the updated weights(w) and bias (b) on our test data to get the predicted values and its accuracy

```
------- Test Data -------

[[31  1]
 [ 6 19]]

Accuracy
 : 0.8771929824561403

Precision
 : 0.8378378378378378

Recall
 : 0.96875
```

6 Conclusion

We can conclude that our model predicts with 87.71 % accuracy that for the given feature variables. This we can improve with increasing the learning rate and running the model for more number of iterations so that we get better weights and bias to train our data.