
Prediction of Breast Cancer using Logistic Regressor

Neelesh K Bhajantri
Department of Computer Science
University at Buffalo
Buffalo, New York 14260
neeshb@buffalo.edu

Abstract

In this project we have gone through logistic regression using gradient descent. We are given a two-class problem to determine whether FNA cells belong to class benign or malignant. We have performed logistic regression on “Wisconsin diagnostic breast cancer” and have trained the model based on 80% of the data set and the remaining 20% for testing and validation. The model gives us 77% accuracy with the test set.

1 Introduction

Logistic regression is a widely used statistical classification model. We build a model and keep iterating the model until we get a minimum loss for the model. That means we are continuing to run the model on different learning rate and iterations so that we can lower down the cost. Since we have two classes here, we use the sigmoid function to determine which class the values lie in. the sigmoid function maps the values between 0 and 1 and we are assigning the benign and malignant values to 0 and 1 and determine which class they belong to. The data set we have, has the preprocessed images of FNA of a breast mass. To minimize the loss function in this project, we are using gradient descent. By using gradient descent, we find out optimal parameters that result in optimizing the sigmoid function. Gradient descent is based on the observation that if a multi-variate function ‘F’ is defined and differentiable in a neighborhood of a point x, then the function ‘F’ decreases fastest if one goes from the point x in the direction of the negative gradient of F at the point x. By repeating the process of minimizing the loss function we can determine the values of benign and malignant with maximum accuracy and precision.

2 Dataset

The dataset of Wisconsin diagnostic breast cancer has 33 columns and 569 rows. The first column is just indexing and we remove that by just giving the header as “False” and the “id” column is just the patient id which is unnecessary for us to build a model. The ten features namely, radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, fractal dimension are spread to 30 features and we are performing logistic regression these 30 features. The diagnosis feature is taken as the dependent variable ‘y’ and all the remaining 30 features become the independent features to find out ‘y’. We first separate the dataset into training, validation and testing with 80, 10, 10 percentages respectively. We use the training data set to build a data model and the validation set to change the epochs and learning rate. Then we use this model to test on the test data that we divided earlier. Partitioning the data randomly helps us to build a very robust model in determining the prediction of the future values. With the help of validation data we are able to tune the hyperparameters into determining a more accurate model for the sake of prediction. We iteratively update the weights and bias on the training data only until we achieve a desirable graph and then we pass the validation data to tune the hyperparameters ie

the epoch and learning rate values. After all this process we dump the test data to test how our model works.

3 Preprocessing

There are many ways to preprocess the data that we have but before getting into the process lets discuss why preprocessing the data is necessary. preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing. The values that we have of the features are distributed randomly and they are very much distinct. Suppose we are trying to plot the values on a 2D graph they would be scattered everywhere. So, for our convenience we distribute the values such that they are around a certain attribute. Now, in order to do this, there are many ways. Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to a common scale, without distorting differences in the ranges of values. For machine learning, every dataset does not require normalization. It is required only when features have different ranges. There is a library in python called "sklearn" which directly preprocesses the data using "sklearn.preprocessing.normalize" or we can do it manually as I did it in our project. We distribute the values around the mean of the features. For every feature we take the mean of that feature and we distribute the minimum and maximum values around that mean and we repeat this process for all the 30 features. This is only needed and done for the input/independent variables. We don't have to normalize the dependent variables as they are randomly scattered according to the given input values.

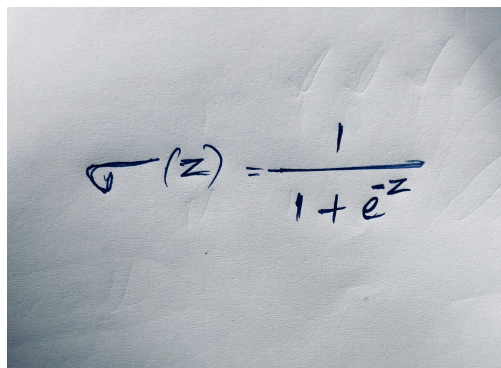
4 Architecture

4.1 Bias and weights function

We have to initially set up the weights and bias. We initialize the bias value to 1 and the weight with 1 with the dimension of the training dataset since we will be performing arithmetic operations with the training dataset in order to calculate the sigmoid function. We write this down as a function and not as an initialization since we have different dimension for training and testing data set and initializing every time is not optimal hence we write down a function and call this function every time we use the weights and bias.

4.2 Sigmoid Function

The sigmoid values determine the class of the feature attributes. Since the sigmoid returns only in the region of 0 and 1 and we are using a 2-class model it will be perfect for us to implement the sigmoid function in this case. The sigmoid function is given as


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

4.3 Loss Function

We need to obtain the cost and the loss of the model in order to progress in building a sustainable model. Here cost is nothing but the mean values of the loss values. The cost helps us determine the flow of graph, how the cost is varying for every iteration and if its decreasing or increasing across the epochs. To calculate the loss function, we use

$$\begin{aligned}\text{let } a &= \sigma(z) \\ L &= \frac{-(y \log a + (1-y) \log(1-a))}{n} \\ &= \frac{-1}{n} (y \log a + (1-y) \log(1-a)) \\ &= \frac{-1}{n} (y \log(\sigma(z)) + (1-y) \log(1-\sigma(z)))\end{aligned}$$

Here we call the sigmoid function that we have written earlier and compute the sigmoid for weights and the training dataset along with the bias. From the equation above, the 'y' is the training parameter 'y' and z is nothing but the sigmoid of the training independent values and weights and bias. Here we are calling the 'L' value as cost value for that certain attribute of the training set. Now we have to compute the changes in weights and bias by using partial derivatives.

$$\begin{aligned}\theta &= \theta - \eta \Delta \theta \\ b &= b - \eta \Delta b \\ \Delta \theta_1 &= \frac{\partial L}{\partial \theta_1} = \frac{-1}{n} \frac{\partial}{\partial \theta_1} \{ y \log \sigma(z) + (1-y) \log (1-\sigma(z)) \} \\ &= \frac{-1}{n} \left\{ y \cdot \frac{1}{\sigma(z)} \cdot \frac{\partial}{\partial \theta_1} \sigma(z) + (1-y) \frac{1}{1-\sigma(z)} \frac{\partial}{\partial \theta_1} (1-\sigma(z)) \right\} \\ &= \frac{-1}{n} \{ y (1-\sigma(z)) x_1 + (1-y) \sigma(z) (-x_1) \} \\ &= \frac{-x_1}{n} \{ y - y \sigma(z) - \sigma(z) + y \sigma(z) \} \\ \Delta \theta_1 &= \frac{-1}{n} \{ y - \sigma(z) \} x_1\end{aligned}$$

We perform the similar partial derivate on the equation of bias and we get a similar equation like this

$$\Delta b = \frac{-1}{n} \{y - \sigma(z)\}$$

4.4 Learning the model

After we calculate the cost values of every attribute we need to maintain a list of all cost values such that we can plot a graph across the epochs to see how the costs are varying, so we call the loss function in learning model and we obtain the weights and cost and append every cost to the cost list. Here I update the weights and the bias values

$$w = w - \alpha \Delta w$$

$$b = b - \alpha \Delta b$$

In this function I additionally store indexes for every 10 cost values just so that would help me scale the graph while plotting. We use “matplotlib.pyplot” to plot the graph and we are observing the graph of cost list against the number of iterations that we are performing. The training data set of dependent and independent values are passed to the loss function from the learning model itself. After performing the changes in weights and biases the function returns the parameters and the derived weights and biases and the cost list.

4.5 Gradient Descent

The most basic and vastly used optimization technique to minimize the above function is Gradient Descent. It is an iterative optimization algorithm to find the minimum of a function. To find the local minimum using gradient descent, steps proportional to the negative of the gradient of the function at the current point are taken. The standard gradient descent algorithm is defined as follows where η is the learning rate.

```
while true,  
     $\theta_t = \theta_{t-1} - \eta_t \Delta(\theta_{t-1})$   
    if  $||\theta_t - \theta_{t-1}|| < \epsilon$  then break;
```

Gradient Descent Algorithm

After the last iteration the above algorithm gives the best values of θ for which the function J is minimum. The disadvantage of this algorithm is that in every iteration m gradients have to be computed accounting to m training examples. If the training set is very huge, the above

algorithm is going to be memory inefficient and might crash if the training set doesn't fit in the memory. In such cases, the Stochastic Gradient Descent algorithm is going to be helpful.

4.6 Performing the prediction

After we build the model and we are satisfied with the given iterations and learning parameter we predict the values based on our training set. Here we take the independent values of the test set and perform a sigmoid function to determine their class. Let's initialize the prediction to all zeros for now and for every value of the sigmoid we divide the values to 0 and 1 if they are less than 0.5 and greater than 0.5 respectively. The prediction function returns us the predicted values of given test data values.

4.7 Logistic Regression

Now all the previous sub function are called or called-via from the logistic regression function. This helps understand the process of logistic regression that is going to take place in order to predict the values for the given test data set. First, we need to know the dimensions of the training data set so that we can create a similar size for the initial weight. Here we are going to call the weights and bias initialize function. To get the loss function and the cost list we just need to call learning model function since it has the cost function call inside it. After we obtain the parameters, we are just going perform prediction on the test dataset. Our logistic regressor return the predicted values of the given test set so that we can further compute the accuracy, precision and recall for the constructed model.

4.8 Calculating Accuracy, Precision and Recall

First, we need to obtain the true positive, true negative, false positive and false negative parameters in order to calculate the accuracy, precision and recall.

TP = The predicted value is malignant and the original value is malignant

TP = The predicted value is benign and the original value is benign

TN = The predicted value is malignant and the original value is benign

FP = The predicted value is benign and the original value is malignant

The formulae for accuracy, precision and recall are given by

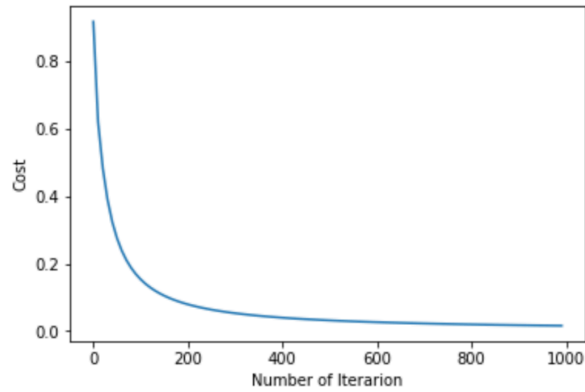
$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

$$Precision = \frac{TP}{TP+FP}$$

$$Recall = \frac{TP}{TP+FN}$$

5 Results

After performing several iterations with the training data set and changing the hyper parameters while using the validation dataset we obtain the learning parameter as 0.3 and the number of iterations as 1000 and we obtain the following graph for cost list versus epochs



6 Conclusion

We have successfully built a model on the Wisconsin diagnostic breast cancer data set which predicts whether the tumor is malignant or benign. To perform this project, we used the logistic regression technique and with the help of sigmoid function we could predict the output of the data by building a model over the data set. Since this is a two class problem we could perform using sigmoid function. In future we can perform prediction of any two class problem with this model.