Project 1- Logistic Regression

Rohtih Suresh
Rohithsu@buffalo.edu

Abstract

A Classification model has been built for dataset of Wisconsin Diagnostic Breast Cancer was done using Machine Learning. The model used here is Logistic Regression for a two class problem. The model trained is able to predict if the given data is Malicious or Benign i.e if the cancer is positive or negative when the data is given.

1 Introduction

Logistic Regression is a statistical model used in Machine Learning for classification. By classification, the model goes through the features in the given data and based on calculation, classifies the data into predefines classes. The model gives a probability value for each dataset based on which classification must be done. The model normally has a accuracy value that dictates how accurate it is in prediction the class of the data.

Here, we have a two-class problem. This means that there are two classes that the data can be classified into that is Malicious or Benign. We are handling features calculated from the images of FNA(Fine Needle Aspirate) of Breast Cancer. These features are the given dataset. We have to pass the features into the logistic Regression model to classify then into the 2 classes. We specify parameters and hyperparameters for the model and then fit the Features into the model. Then, this model is set to run for a certain number of epochs after which the model will be able to predict the class of a given data to a certain accuracy. The accuracy here is determined by the parameters like Weights and Bias and hyperparameters like epochs and learning rate. After the model runs for the fixed number of epochs, the model will have learned and will be able to predict with a certain accuracy. So, we can feed the model the validation and testing data set and see with how much accuracy the model can classify the data. The accuracy we get here will prove how well the model is trained.

2 Dataset

The Dataset we are using here is the features which are precomputed from FNA of Breast Cancer. These features are computed from digitized images of cell nuclei if the image having features like radius, texture, area, parameter etc. The mean, standard and worst of each of these given features are calculated to give 30 final features. For this project, we are given a dataset of dimensions 529 x 32. This dataset has to be used for all purposed i.e training, validation and testing.

First we have to read the given data set and import it into the python environment. Then we can remove the first column that contains the Id which is essentially useless for our model. Then the dataset has to be partitioned into Training, Validation and Testing at 80%, 10%, 10% respectively. This has to be done randomly. Then we have to separate the 3 parts into theirs X and Y parts. The new first column after removing the id has the Y values. The Y values are the classes M and B, M being Malignant and B being Benign. You have to conver these classes to binary values(0,1) for out model. This can be done directly to the .csv file or in the python program. After we get the X and Y values, we have to normalize the X values. This is done as the given features are very big and can have different metrics or measurements which can cause issues. So, after we normalize the data we do not need to

	0	1	2	3	...	26	27	28	29
338	10.05	17.53	64.41	310.8	...	0.1055	0.06499	0.2894	0.07664
427	10.80	21.98	68.79	359.9	...	0.1927	0.07485	0.2965	0.07662
406	16.14	14.86	104.30	800.0	...	0.2310	0.11290	0.2778	0.07012
96	12.18	17.84	77.79	451.1	...	0.0498	0.05882	0.2227	0.07376
490	12.25	22.44	78.18	466.5	...	0.1230	0.06335	0.3100	0.08203

[5 rows x 30 columns]

Figure 1- Example of X of training dataset

worry about the differences in the same features. The normalization is done with the following formula-

$$X(\textit{nomalized}) = (X - \min(X)) / (\max(X) - \min(X))$$

Here, the min(X) is the smallest value of the respective Feature column and max(X) is the highest value of the respective features. After this, the dataset is ready and we can fit it into the model.

	338	427	406	...	359	192	559
0	0.017694	0.019014	0.028415	...	0.016613	0.017113	0.020264
1	0.030863	0.038697	0.026162	...	0.032254	0.032077	0.042130
2	0.113398	0.121109	0.183627	...	0.105317	0.106919	0.131197
3	0.547183	0.633627	1.408451	...	0.490493	0.507218	0.710387
4	0.000177	0.000155	0.000167	...	0.000178	0.000122	0.000163

[5 rows x 455 columns]

Figure 2- Example of normalized X training dataset

3 Architecture

The Logistic Regression model uses a few parameters to train and classify data.

The main parameters what are computed at the end of all the epochs are the Weights and Bias. These parameters are updated at the end of each epoch and at the end of the training, the value we receive can be used for classification of unseen data.

For this, first we can initialize the weights and biases to random values or zero/ones. In my model I chose random values. We also have to initialize the hyperparameters. In the case of logical regression, the hyperparameters are Epochs and Learning rate. The learning rate value need to be chosen carefully, as a bigger value can overshoot the global minima and a very small value can make the training extremely slow.

After the equations are initialized, we have to calculate the linear regression value by using the weights(W^{T}) and bias(W_o) in the formula-

$$Z = W^{T}X + W_o$$

After we get the value Z, we have to run this through the sigmoid function to get the values between 0 and 1. Only when we get this done properly can we do the classification. The formula for sigmoid function is –

$$H = 1/(1 + e^{-z})$$

This value H can be then used to update the Weight with the formula;

$$W = W - ((-(Y-H)X^T / M) \text{ x } \eta$$

Where η is the learning rate

The bias also has to be update with a similar equation;

$$W_o = W_o - ((-(Y-H) / M) \text{ x } \eta$$

These two updates have to be done at the end of each epoch to improve the parameters. And the formula

We also have to generate the binary values(1,0) at the end of each epoch. This will be used to compare and get the accuracy of the training and validation. This accuracy test can be done at the end of each epoch too. That way we can monitor improvement in accuracy.

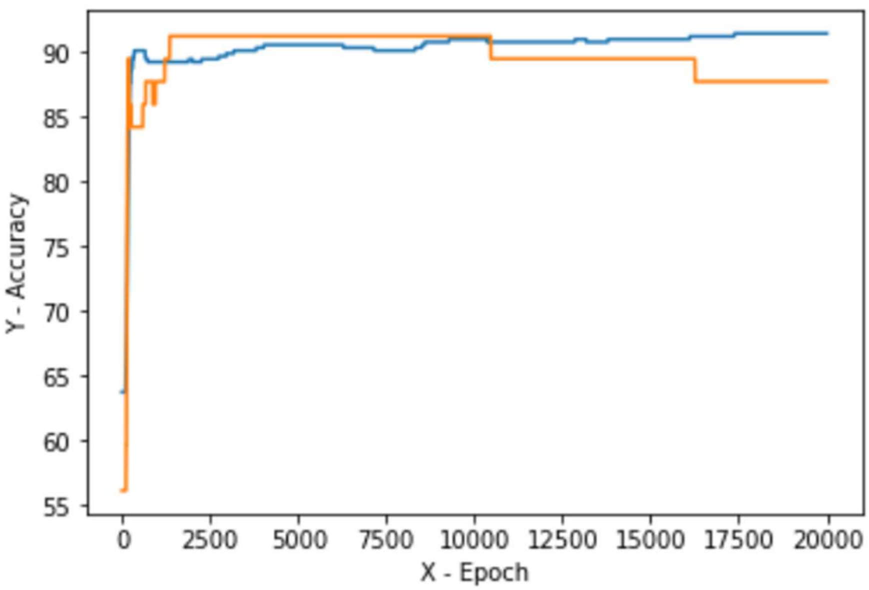

Figure 3- Epoch vs Accuracy for train(blue) and validation(yellow) data

We can also find the loss function for each epoch. The loss function used here is Cross Entropy function. We can monitor the loss function at the end of each epoch and observer the function reach the Global minima. To put it in crude way, the loss function calculates the difference between the input and the predicted values. As the model keeps learning, we can see the loss function value decreasing to reach a Global minima. This show the improvement of the model.

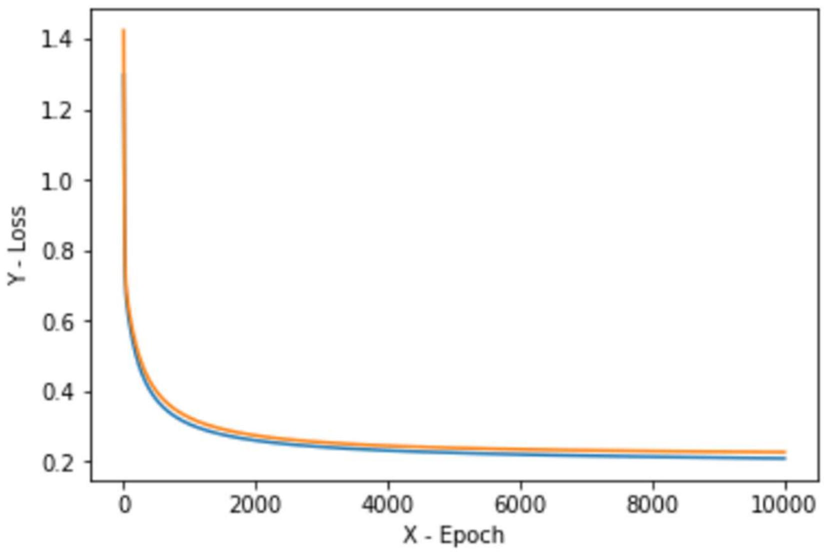

Figure 4- Epoch vs Loss Function value

4 Result

After 20000 epochs, my accuracy value for training data is around 90% and validation value is around 87%.

```
print(train_accuracy)
print(val_accuracy)
```

```
91.42857142857143
87.71929824561403
Text(0, 0.5, 'Y - Accuracy')
```

This shows that the model is trained. Now, after using this model for the testing date I get the following values

```
print("Test Accuracy -", accuracy)
print("Precision -", precision)
print("Recall -", recall)
```

```
Test Accuracy - 0.9824561403508771
Precision - 0.9722222222222222
Recall - 1.0
```

Hence, we can see that our model is trained.

5 Conclution

We can see from the above mentioned that, using logical regression, a classifier has been built. For the given dataset, after training and validation, it has a good accuracy and precision value even on unseen data. This shows that the built model is successful and works properly.