Logistic Regression on Breast Cancer Wisconsin (Diagnostic) Dataset

Gaurav Madhav Pai
Department of Computer Science
University at Buffalo (SUNY)
Buffalo, NY 14226
gmadhavp@buffalo.edu

Abstract

This project is an attempt at using Logistic Regression, coded from scratch, to learn from the Wisconsin Breast Cancer dataset in order to predict whether the data represent the characteristics of the cell nuclei is benign or malignant. Gradient Descent for optimization of weights and bias. An accuracy of 97%, recall 1 and precision of 96% is obtained on the test set.

1. Introduction

Breast cancer is cancer that forms in the cells of the breasts. After skin cancer, breast cancer is the most common cancer diagnosed in women in the United States. Breast cancer can occur in both men and women. In this project, given the data relating to the fine needle aspirate of a breast mass, an attempt is made to predict whether a person has breast cancer or not.

2. Dataset

The dataset contains 569 observations, each of which corresponds features extracted from a digitized image of a fine needle aspirate (FNA) of a breast mass. There are 32 features for each of the observation which.

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 ($perimeter^2/area - 1.0$)
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)

Figure 1 – Description of features

The above table (Figure 1) gives the details about some of the features and operations like, mean, standard error etc on these features were done to give a set of 30 features for each observation.

3. Preprocessing

Since, the target variable has values M and B representing Malignant & Benign respectively, it has to be converted to 1 and 0. After the conversion, these target values are separated and stored in a new variable with dimensionality 596x1. The following code is

used to perform the conversion.

```
df[1] = df[1].apply(lambda x: 1 if x=="M" else 0)
```

Further the column that represents the ID of the images is dropped since it does not provide any insight into the problem that is being tackled. The result of this operation is a dataset with the dimensionality 596x30. Observing the min and max values of each of the feature, it is noticed that the ranges of the features vary vastly. The Fig 2 shows the statistic of two variable with vastly different range.

5	6
569.000000	569.000000
654.889104	0.096360
351.914129	0.014064
143.500000	0.052630
420.300000	0.086370
551.100000	0.095870
782.700000	0.105300
2501.000000	0.163400

Figure 2 Statistic of two features

Hence, the features are normalized using a MinMaxScaler that results in the range of all data between 0,1. Fig3 represents the code to perform normalization of data.

```
scaler = MinMaxScaler()
x_train_lr = scaler.fit_transform(x_train)
x_test_lr = scaler.transform(x_test)
```

Figure 3 – Code for normalization of features

4. Architecture

4.1 Logistic Regression

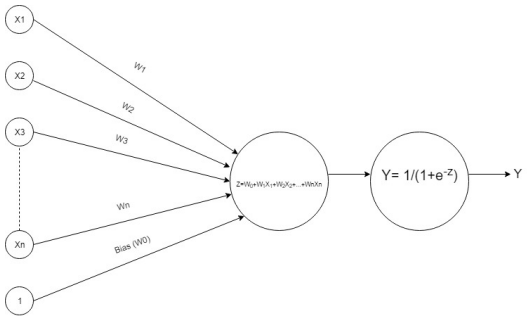


Figure – Computational Graph for Logistic Regression

In logistic regression classification we pass the linear equation through a sigmoid function in order to squeeze the output of the linear equation between 0 and 1. The linear

equation that incorporates the weights and bias to give a prediction is given in Figure 4. Figure 5 shows the Sigmoid function

$$Z = WX + B$$

Figure 4 – Linear Equation

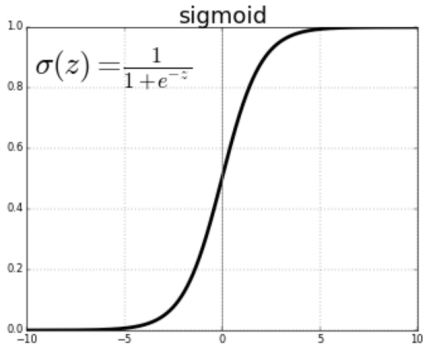


Figure 5 - Sigmoid Function

4.2 Loss Function

The loss function gives us a single number representing the how bad is the classification performance of the algorithm. Figure 6 represents the Cross Entropy loss function that is used by the Gradient Descent for optimization of weights and bias.

$$Cost(h_\theta(x), y) = -ylog(h_\theta(x)) - (1 - y)log(h_\theta(x))$$

Figure 6 Loss Function

4.3 Gradient Descent

In this project, Gradient Descent is used for the optimization of weights and bias. The gradient of the loss function is calculated and accordingly each weight is reduced by the product of the gradient and learning rate. The learning rate (α) is used to control the descent of the weights. Figure 6 and 7 represent the steps involved in updating the weights and bias using Gradient Descent

$$\frac{\partial}{\partial w} J(w) = \nabla_w J$$
$$\frac{\partial}{\partial b} J(w) = \nabla_b J$$

Figure 6 – Calculating the Gradient (First Derivative)

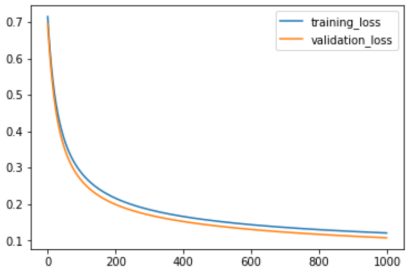
$$w = w - \alpha \nabla_w J$$
$$b = b - \alpha \nabla_b J$$

Figure 7 – Updating the weights and bias

5. Results

5.1 Training the model

Initially Logistic Regression Model was trained for 1000 epochs and learning rate = 0.5. Figure 8 Shows the result. Since the validation loss was lesser than the

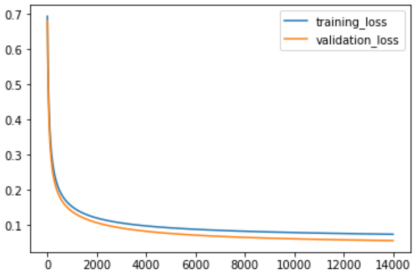



```
Training loss    0.12030805950983116
Validation loss    0.10699995426492176
```

Figure 8 – Results with epochs = 1000 and $\alpha = 0.5$

5.2 Tuning the hyperparameters of the model

To tune the hyper-parameters (epochs, learning rate α) the following values were used Epochs = [4000, 8000, 14000] and $\alpha = [0.25, 0.12, 0.06]$. Training the algorithm for all combinations of epochs and α it was found that the learning rate of 14000 and α 0.25 gave the lowest validation loss and highest validation accuracy. Figure 9 shows the corresponding results The evaluation metric "accuracy" is explained in section 5.4.




```
Training loss    0.07432668229506091
Validation loss    0.05626663330105721

Validation accuracy   0.989010989010989
Training accuracy   0.9807692307692307
```

Figure 9 – Epochs = 14000 and $\alpha = 0.25$

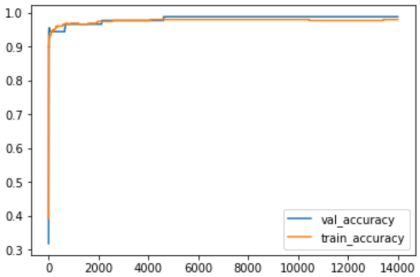


Figure 10 – Validation and Training accuracy for final training with best hyperparametes

5.3 Testing the model

The above mentioned tuning results seemed promising and hence the values of w (weights), b (bias) and epochs = 14000 and α = 0.25 was used on the testing set to obtain a accuracy, recall and precision on testing set of 0.973, 0.925 and 1.

5.4 Evaluation Metrics

This following Figure A give the formulae for computation of the evaluation metrics i.e Accuracy, Precision and Recall.

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

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

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

Figure A : Accuracy , Precision and Recall

6. Conclusion

Since the validation loss was lower than the training loss, it can confidently said that the model is neither overfitting and not underfitting. Further, on unseen data of the testing set the model gave an accuracy, precision and recall of 97.4% , 100% and 92.5% respectively. Therefore, if the model is deployed in the real-world it can be conclusively said that it will perform with an accuracy of 97.4%.