Logistic Regression of Wisconsin Breast Cancer Database

Yash Sanjiv Patange
Department of Computer Science
UB Person No: 50319943
University at Buffalo
Buffalo, NY 14221
yashsanj@buffalo.edu

Abstract

The following report presents Logistic Regression performed on the Wisconsin Breast Cancer Database. Logistic Regression was performed to predict, based on different features of the cancer cell, whether the cancer is malignant or benign.

1 Introduction

In Machine Learning, prediction problems are very interesting and prevalent. As the name suggests, with the help of Machine Learning Algorithms we can predict value of anything provided we have certain initial knowledge. The most common example is loan defaulting. This is used by many banks and financial institutions to predict, if a loanee can pay off the loan in the stipulated time period. For this, the prior information needed is the customers financial details, like salary, savings, assets, credit score, has he/she paid previous loans on time, have they defaulted anytime in the past. This information is fed into a predictive model, and the model predicts whether the customer will default or not. To predict this, we use regression analysis. In statistics, regression analysis gives us the relationship between variables. Now regression analysis is of different types. Of which, linear regression is the simplest one. In linear regression, the linear relationship between variables is found out. If a value of one variable is x at some point, what will be the value of the other variable at the same point. It is also called linear regression because, the relationship between the variables is always linear. Now linearity always, is not necessarily a good thing. Suppose there are many variables and predictions are all over the place. To correctly predict and fit the model we need to have some kind of non-linearity to achieve this. And as complexity of problem increases, number of inputs increases, it becomes difficult to maintain linearity. Thus, there needs to be some way to introduce non-linearity in a linear model. This is achieved by using the sigmoid function, which is an important component of Logistic Regression. Linear Regression is used to predict the value of a variable. When we apply sigmoid function to output of Linear regression we get the probable value that the output belongs to a certain class or not. This is also known as classification problem. Logistic Regression is used to classify the dataset into 2 classes. Thus, Logistic Regression is binary classifier. Logistic Regression successfully predicts if the output belongs to one class or not.

2 Dataset

The dataset used for this project is the Wisconsin Breast Cancer Database(wdbc). This dataset consists of features precomputed from the images of Fine Needle Aspirate of a breast mass. It consists of 569 instance of cells and 32 attributes. These 32 attributes contain ID,

diagnosis(B/M) and 30 real-valued features. The mean, standard error and largest of these features were computed for each image which resulted in 30 features. The characteristic used to form the features are:

1	Radius (mean of distances from center to points on the perimeter)
2	Texture
3	Perimeter
4	Area
5	Smoothness (local variation in radius lengths)
6	Compactness (perimeter2/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)

3 Preprocessing

It is very important to understand exactly what the dataset is about, or at least what features are to be used. This is done in the preprocessing stage. The data is preprocessed and understood or made less complicated. This allows for correct output. In preprocessing the data is also split into 3 different sets. The training set, validation set and testing set. The training set is used for training the weights and bias of the model. Validation set is used for tuning the hyperparameters, namely, learning rate and the number of epochs. Testing set is the final set, where we apply logistic regression with our tuned weight, biases and hyperparameters to the dataset. In preprocessing it is also necessary to normalize the dataset. Normalization implies that the values of the feature are in a comparable range to each other. If one is very high and the other is low, the model won’t be properly trained.

4 Architecture

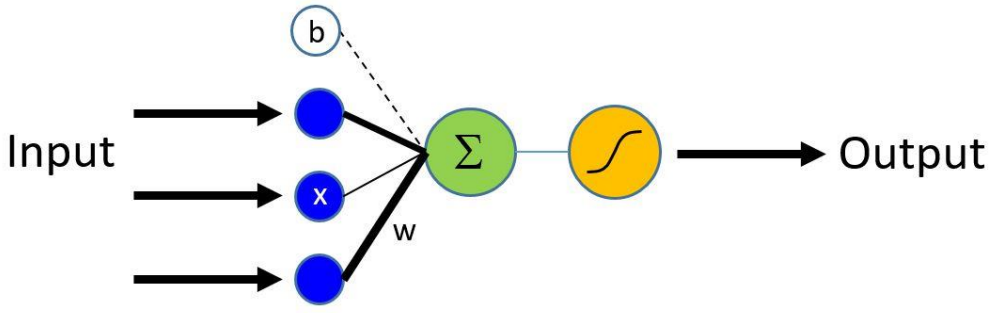

Fig 1. [1]

Logistic Regression is regression analysis technique that classifies the output into two classes. Thus, Logistic Regression is 2 class classifier or binary classifier. Logistic Regression is also a linear model. As shown in Figure 1, feature inputs and a bias input is

provided to summation function, which is basically similar to linear regression.The output of linear regression is applied to a sigmoid function which then tells which class it belongs to. The sigmoid function is an activation function that converts any value and maps it between 0 and 1. A threshold is decided (usually 0.5). If below threshold, it belongs to class 0. If above threshold it belongs to class 1. The sigmoid function is given as:

$$w^T.x + b = z$$

Fig2. [2]

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

Fig3. [3]

In figure 2 we have *w* as the weights, that are transposed, multiplied with *x* which is the input or feature set. *b* is the initial bias and *z* is the output. Now this *z* is used in the sigmoid function show in figure 3. Here sigmoid function is given by g(z). This function will map the value of *z* between 0 and 1. We need to have a proper set of weights and bias for our model to be accurate enough. Initially these weights and biases are assigned values 0. But we need them to a certain value to perform accurately. This is where gradient descent algorithm helps us. In gradient descent we iteratively choose the next lower value till we arrive a minimum. Thus, we iteratively change our weights and bias. The formula for gradient descent is arrived after partially differentiating the loss or cost function. The loss function is measure of how good our model is performing. It is given by: -

$$L(p, y) = -(ylogp + (1 - y)\, log(1 - p))$$

Fig 4. [4]

Where *p* is our predicted values set, and y is our actual correct output. To change our weights and biases we need to partially different this Loss function, separately, with respect to weight and bias respectively. Thus, we arrive at:

$$\frac{\partial L(p,y)}{\partial w} = \frac{\partial L(p,y)}{\partial z}\frac{\partial z}{\partial w} = dz\,\frac{\partial}{\partial w}(w^T.x + b)$$

$$\frac{\partial L(p,y)}{\partial w} = dw = \frac{1}{\underbrace{m}_{training\ example}} X dz^T$$

Fig5. [5]

$$\frac{\partial L(p,y)}{\partial b} = \frac{\partial L(p,y)}{\partial z}\frac{\partial z}{\partial b} = dz\,\frac{\partial}{\partial b}(w^T.x + b)$$

$$\frac{\partial L(p,y)}{\partial b} = db = \frac{1}{\underbrace{m}_{training\ example}} \sum_{k=1}^{m} dz^k$$

Fig 6. [6]

Figure 5 shows us the weight update equation. Figure 6 shows us the bias update situation. Here *m* is the number of training examples, i.e. training set inputs. This is done iteratively on every epoch number for all weights and bias. After weights and bias, there are 2 other parameters namely, epochs and learning rate. Now, epochs are the number of iterations that we perform gradient descent algorithm for. Generally, higher the number of epochs, better is the model fit. Learning rate is a hyperparameter that decides how fast are the new values overriding the older values [7]. If it is too low, the new values won't get overridden quickly. If it is too high then, it might get quickly overridden and our model will be overfitted. Hence, learning must always hit that sweet spot.

5 Results

The results were obtained after running the program quite a number of times. Each time a different parameter was altered and the result was arrived at. In this fine tuning of hyperparameters, training set and the validation set was used. Training set was used to find the weights and biases for the dataset. Based on the accuracy of validation set, the following parameter tweaking was performed. The following figures will explain how the hyperparameters were chosen:

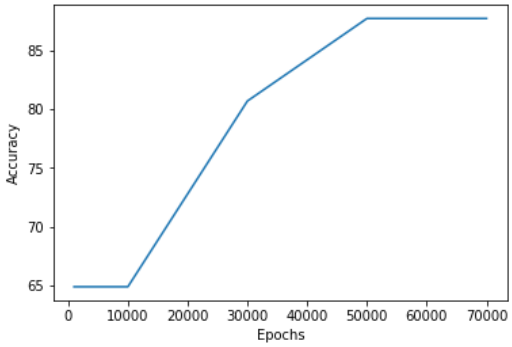

Fig 7.

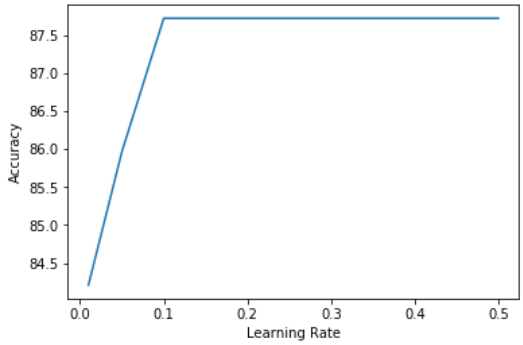

Fig 8.

Figure 7 graph shows the "epochs vs accuracy" graph. Epochs value chosen was 1000, 10000, 30000, 50000, 60000 and 70000. A very wide range of set was used. As the number of epochs increases we get a higher accuracy. But also, the higher your number of epochs, more time the program takes to run. Thus, time taken can be sacrificed for a higher rate of accuracy. Figure 8 shows us the "learning rate vs the accuracy" graph. Similarly, here too different values of learning rates were used. They were 0.01, 0.05, 0.1, 0.5. This shows that the accuracy stagnates for values greater than 0.1. Hence based on these graphs our hyperparameters can be set as epochs = 70,000 with a learning rate of 0.1. The gradient descent for this set of hyperparameters is:

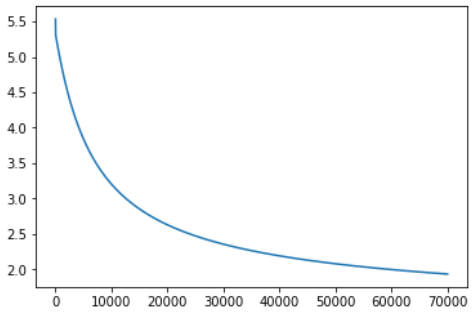

Fig 9.

Based on the hyperparameters, weights and bias, Logistic Regression was performed on the Testing set. This yielded commendable results. The Accuracy of the model was 89.47%. The Precision of the model was 88.9% and the Recall rate was 88.9%.

6 Conclusion

Thus, Logistic regression was performed on the Wisconsin Breast Cancer Database. Logistic Regression is a 2-class problem and helps us to classify which class will the output belong to. Sigmoid function is a helpful function that makes classification possible by mapping the values between 0 and 1. Gradient Descent algorithm is performed iteratively over the weights and bias to obtain the optimum values.

References

[1]. https://gallery.azure.ai/Tutorial/Cognitive-Toolkit-101-Logistic-Regression-and-ML-Primer

[2], [3], [4], [5], [6] https://towardsdatascience.com/logistic-regression-from-very-scratch-ea914961f320

[7]. https://en.wikipedia.org/wiki/Learning_rate