Binary Logistic Regression using Python

Uneeth Akula
50317480
Dept. of Computer Science
University at Buffalo
Buffalo, NY 14221
uneethak@buffalo.edu

Abstract

Breast Cancer is the most common cancer among women. An early diagnosis will expediate the treatment of this ailment. The accuracy of visually diagnosed breast fine needle aspirates, however, is less. It is, therefore necessary to minimize the subjectivity with machine learning techniques. In this project the presence of malignant tumor cells in breasts will be predicted for the Wisconsin Diagnostic Breast Cancer Data. This project employs a supervised Machine Learning technique, "Logistic Regression". The performance of model will be tested by the accuracy, precision and recall.

1 Introduction

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value. A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a reasonable way.

In unsupervised learning, no labels are given to the learning algorithm, leaving it on its own to group similar inputs through clustering or density estimates of high-dimensional data that can be visualized effectively.

Supervised Machine Learning comprises of 2 training techniques, Linear Regression and Logistic Regression. Linear Regression predicts a continuous valued output whereas Logistic Regression, more commonly known as Classification predicts a discrete valued output.

Analyzing the dependent variable Y and the independent variables X, the goal is to find the value of W and b, also known as the Weights vector and Bias so that we can fit the model on our data, find the decision boundary and the malignancy of the tumor.

The cost function is used as a measurement parameter of logistic regression model. We must keep changing the value of ϴ to minimize the cost. Gradient Descent is used to minimize the cost. The value of weights ϴ and Bias B is updated until the cost is minimized, and we arrive at the global minimum. After finding the final Weights vector ϴ based on the logistic function, decision boundary and optimum learning rate we estimate the probability of a patient's tumor being malignant or benign.

2 Logistic Regression

This Project employs Logistic Regression to model the probabilities for classifying the two possible outcomes. Regression is a statistical process that estimates the relationship between the dependent and independent variables. Logistic Regression is a supervised machine learning technique, employed in classification jobs (for predictions based on training data). Binary (0,1) outcomes can be predicted from the independent variables. The outcome of dependent variable is discrete. Logistic Regression uses a simple equation which shows the linear relation between the independent variables. These independent variables along with their coefficients are united linearly to form a linear equation that is used to predict the output.

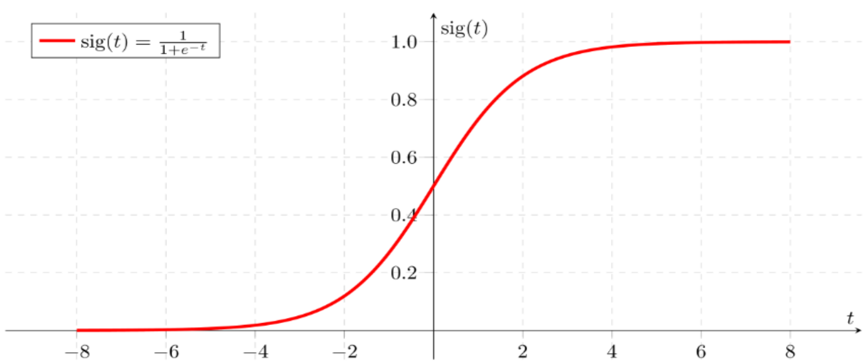
2.1 Logistic Function

The logistic regression model uses the Logistic function to limit the output of a linear equation between 0 and 1.

The logistic function is defined as:

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

For large positive values of z, the sigmoid function should be close to 1, while for large negative values of z, the sigmoid function will be close to 0.



The hypothesis function for logistic regression:

$$h_\theta(\mathbf{x}) = g(\theta^\top \mathbf{x})$$

2.2 Decision Boundary

The line which line /curve /boundary which separates the region where y=1 from y=0 is the decision boundary.

After estimating the parameters, we get a decision boundary which approximately separates the data into 2 classes, (0,1).

If y=1 this means basis function calculated with sigmoid >0.5.

Likewise, if y=0 this means basis function < 0.5.

Decision Boundary is the property of hypothesis of parameters not the data set.

2.3 Cost Function

The cost function used in this project is Binary Cross Entropy Loss. It is a sigmoid activation and a cross entropy loss. The cost function helps us to find the right value of Θ, b in the best possible time so that our decision boundary fits our case. We can predict the value of dependent variable from independent variables. Starting with Θ and bias values as zero, we find that difference between actual and predicted value. So, the Cost function is used as a measurement parameter of our logistic regression model. Cost function is defined as below.

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))]$$

2.4 Learning Rate

Learning Rate is the hyper parameter which has to be tuned in order to minimize the cost function. After training the model with the train data set, the validation data set is used to tune the hyper parameters to get optimum results.

3 Data Set

Wisconsin Diagnostic Breast Cancer dataset was used for training, validation and testing. The dataset contains 569 instances with 32 attributes (ID, diagnosis (B/M), 30 real-valued input features). Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. Computed features describe the following characteristics of the cell nuclei present in the image:

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 (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)

The mean, standard error, and “worst” or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features.

4 Preprocessing

The Data given was split into three sets: Training data, Test data and Validation data with a ratio of 8:1:1. The train_test_split function from the sklearn library in python was used to split the data. Each set was then scaled and normalized to get optimum results.

The first column contain the patient ID’s was dropped from the data set as it is neither dependent nor independent variable. The first column containing the Diagnosis data is assigned to Vector Y. The remaining Columns i.e. 2:31 is assigned to X which is a vector of independent variables x(i).

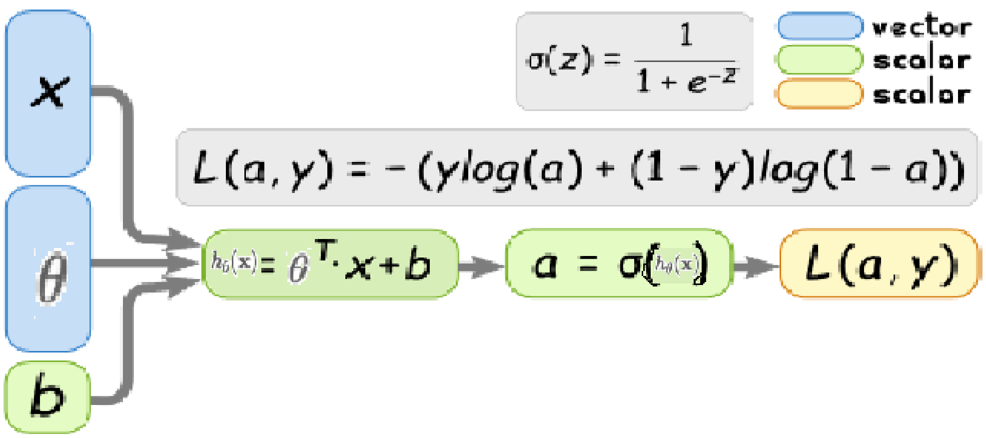
4.1 Feature Scaling

This method is widely used for normalization in many machine learning algorithms. The general method of calculation is to determine the distribution mean and standard deviation for each feature. Next, we subtract the mean from each feature. Then we divide the values (mean is already subtracted) of each feature by its standard deviation.

Where $x' = x - \bar{x} / \sigma$

Where x is the original feature vector, $\bar{x} = average\ (x)$ is the mean of that feature vector, and σ is its standard deviation.

5 Architecture



The hypothesis function for the logistic regression, is given by

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

Where g(z) is the sigmoid function defined as

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

We must choose the best parameters Θ's in the equation above to minimize the errors. Here Θ is not a single parameter. We will minimize the cost function by finding the best possible values of Θ. The minimization will be performed by gradient descent algorithm, whose task is to parse the cost function output until it finds the lowest output.

For logistic regression the cost function is defined as:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} [y^{(i)} \log(h_\theta(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_\theta(x^{(i)}))]$$

Where m is the no. of training examples.

We must minimize the cost function, which will output the best set of parameters Ө. The way we are going to minimize the cost function is using the gradient descent. To minimize the cost function, we must run the gradient descent function on each parameter.

$$\Theta = [\, \theta o, \theta 1, \cdots \theta n]$$

Repeat until convergence {

$$\theta j := \theta j - \alpha \frac{\partial}{\partial \theta j} J(\theta) \qquad \text{------------- (1)}$$

}

Where j= 0,1, 2…n

Let a=hӨ(x) = sigmoid (Ө.T X+B), then

$$a = \sigma(z)$$

$$z = \theta^T x + B$$

$$J = \frac{-1}{m}\left\{ y \log \sigma(z) + (1-y) \log(1-\sigma(z)) \right\}$$

$$\frac{\partial J}{\partial \theta_1} = \frac{-1}{m} \frac{\partial}{\partial \theta_1}\left\{ \underbrace{y \log \sigma(\theta^T x + B)}_{(1)} + \underbrace{(1-y)\log(1-\sigma(\theta^T x + B))}_{(2)} \right\}$$

Let us solve Term (1)

$$\frac{\partial}{\partial \theta_1} \cdot y \log \sigma(\theta^T x + B)$$

$$\Rightarrow y * \frac{1}{\sigma(\theta^T x + B)} \cdot \frac{\partial}{\partial \theta_1}\left[1 + e^{-(\theta^T x + B)}\right]^{-1}$$

$$\Rightarrow y * \frac{1}{\sigma(\theta^T x + B)} * (-1)\left[1 + e^{-(\theta^T x + B)}\right]^{-2} * (-1) * e^{-(\theta^T x + B)} * [x_1]$$

$$\Rightarrow y * \frac{1}{\sigma(\theta^T x + B)} * \frac{e^{-\theta^T x + B}}{1 + e^{-(\theta^T x + B)}} * \frac{1}{1 + e^{-(\theta^T x + B)}} \times x_1$$

$$\Rightarrow y * \frac{1}{\sigma(\theta^T x + B)} * (1 - \sigma(\theta^T x + B)) \times \sigma(\theta^T x + B) * x_1$$

$$\Rightarrow y * (1 - \sigma(\theta^T x + B)) * x_1$$

$$(1) \Rightarrow y * (1-a) * x_1$$

$$\text{Solving } (2) \Rightarrow -(1-y) * a * x_1$$

$$\therefore \frac{\partial J}{\partial \theta_1} = \frac{-1}{m}\left\{ y - \sigma(z) \right\} x_1$$

Deriving with respect to bias we get,

$$\frac{\partial J}{\partial B} = \frac{-1}{m}\left\{ y - \sigma(z) \right\} * 1$$

The obtained gradients or derivatives are then substituted in equation (1) for k no. of iterations to minimize the cost function.

Finally, we get optimum values for ϴ and Bias which are the best combinations of weights and Bias. We use these values in the hypothesis equation to predict the output of the test data.

4 Results

Four performance metrics namely accuracy, precision, recall and f1-score are used to evaluate the performance of the trained models.

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

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

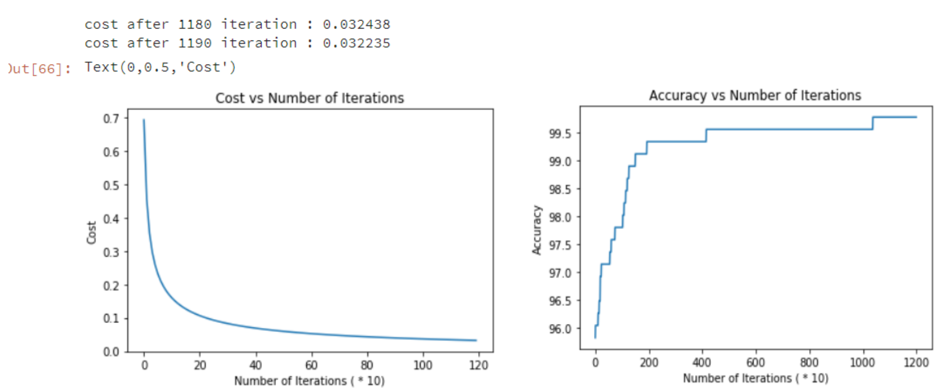
$$Recall = \frac{TP}{TP + FN}$$

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Where

	Actual Positives	Actual Negatives
Positive Predictions	True Positives (TP)	False Positives (FP)
Negative Predictions	False Negatives (FN)	True Negatives (TN)

First the model is trained with the training data set and the cost function is minimized by the forward and backward propagation by calculating the gradients. The cost vs no. of epochs and accuracy vs no. of epochs for the training data is as shown in Fig(1) and Fig(2).

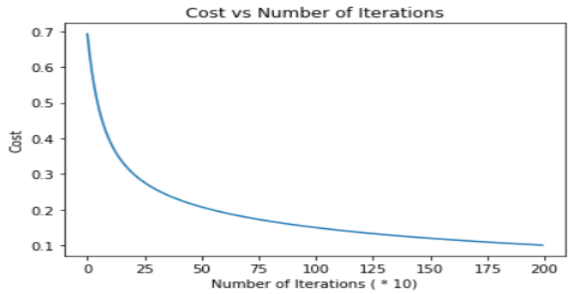


Fig(1) Fig(2)

After training, the validation data was plugged in to find the minimum cost function value by tuning the learning rate.

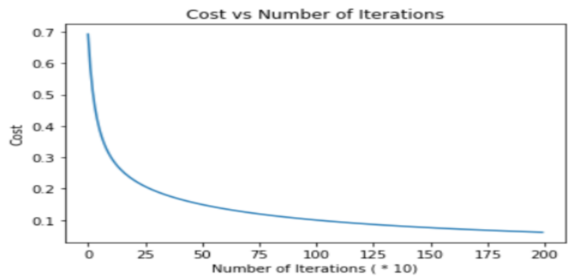
The cost function vs iterations for learning rate = 0.003

```
cost after 1980 iteration : 0.100668
cost after 1990 iteration : 0.100340
Text(0,0.5,'Cost')
```



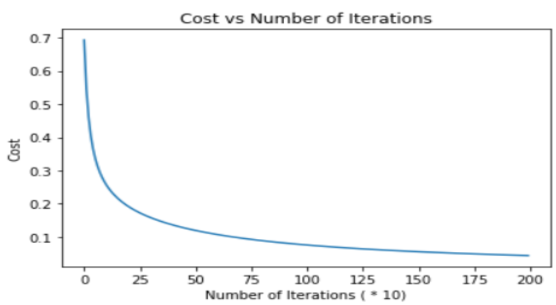
The cost function vs iterations for learning rate = 0.006

```
cost after 1980 iteration : 0.061365
cost after 1990 iteration : 0.061125
Text(0,0.5,'Cost')
```



The cost function vs iterations for learning rate = 0.009

```
cost after 1980 iteration : 0.044141
cost after 1990 iteration : 0.043954
Text(0,0.5,'Cost')
```



Finally, with the learning = 0.009 and epochs = 1200 the test data set was plugged in to predict the diagnosis with the obtained weights and bias. The results for the test dataset is shown

```
accuracy = accuracy_numerator / accuracy_denominator *100
precision = precision_numerator / precision_denominator *100
recall = recall_numerator / recall_denominator *100
fmeasure = 2 * (recall *precision) / (recall + precision)

print('Test Data Accuracy: ', accuracy)
print('Precision: ', precision)
print('Recall: ', recall)
print('f_measure: ', fmeasure)
```

```
Test Data Accuracy:  94.73684210526315
Precision:  84.21052631578947
Recall:  100.0
f_measure:  91.42857142857142
```

6 Conclusion

Successfully trained the model using the given Wisconsin Diagnostic Breast Cancer Data, validated and tested the model using the testing data with an accuracy of 94.73%. The recall obtained for the test data is 100% and the precision is 84%. We do not want any affected patient to be classified as not affected without giving much heed to, if the patient is being wrongfully diagnosed with cancer. This is because, the absence of cancer can be detected by further medical diseases, but the presence of the disease cannot be detected in an already rejected candidate. Hence in such applications, the recall value must be high and the precision value must be low. With this model we have obtained an 100% recall which is a perfect fit for such applications.

7 References

[1]https://towardsdatascience.com/logistic-regression-from-very-scratch-ea914961f320
Logistic Regression

[2]https://www.internalpointers.com/post/cost-function-logistic-regression
The cost function in logistic Regression

[3]https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc
Logistic Regression detailed overview

[4]http://ronny.rest/blog/post_2017_08_12_logistic_regression_derivative/
Derivatives calculation step by step

[5]https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html
Pandas.DataFrame