Two Class Classification using Machine Learning

Sai Shanthan Venreddy
50315873
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
saishant@buffalo.edu

Abstract

In this project, we perform classification using machine learning. It is for a two-class problem. The features used for classification are pre-computed from images of a fine needle aspirate (FNA) of a breast mass. The task is to classify suspected FNA cells to Benign (class 0) or Malignant (class 1) using logistic regression as the classifier. The dataset in use is the Wisconsin Diagnostic Breast Cancer (wdbc dataset). Wisconsin Diagnostic Breast Cancer (WDBC) dataset will be used for training, validation and testing. The dataset contains 569 instances with 32 attributes (ID, diagnosis (B/M), 30 real-valued input features).

1 Introduction

Breast cancer is a major cause of concern in the United States today. At a rate of nearly one in three cancers diagnosed, breast cancer is the most frequently diagnosed cancer in women in the United States. The American Cancer Society projected that 211,300 invasive and 55,700 in situ cases would be diagnosed in 2003. Furthermore, breast cancer is the second leading cause of death for women in the United States and is the leading cause of cancer deaths among women ages 40—59. In this report, the data comes from Breast Cancer Wisconsin (Diagnostic) Data Set. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.

We use Logistic regression in this project. Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. Some of the examples of classification problems are Email spam or not spam, Online transactions Fraud or not Fraud, Tumor Malignant or Benign. Logistic regression transforms its output using the logistic sigmoid function to return a probability value. Logistic regression outputs a value between [0,1]. Therefore, it naturally functions as a two-class classifier. That is if the output is greater than or equal to .5 then we conclude that the input is in class 1 or else if output is less than .5 then it's in the class 0.

2 Dataset

The Data set that we are using to train, validate and test is the Wisconsin Diagnostic Breast Cancer (wdbc dataset). The dataset consists of 569 instances with 32 attributes (ID, diagnosis (B/M), 30 real-valued input features). Features are drawn from a digitized image of a fine needle aspirate (FNA) of a breast mass. Computed features describe the following features 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.

3 Pre-Processing

This data is saved in .CSV format and is processed or read by using pandas. Unnecessary data from the dataset like id are removed using drop function. Then this data is divided into 3 parts:80% of the dataset taken from wdbc is used for training, 10% is used for validation and the rest for testing purposes by using train_test_split function from sklearn.model_selection library. This data is normalized and so that it is efficient and easier to work with.

$$x' = \frac{x - \text{average}(x)}{\max(x) - \min(x)}$$

Figure 1: Mean Normalization

where x is an original value, x’ is the normalized value. There is another form of the mean normalization which is when we divide by the standard deviation which is also called standardization. Then this data is used for further implementation.

4 Architecture

Logistic Regression is a Machine Learning algorithm which is used for the classification problems, it is a predictive analysis algorithm and based on the concept of probability. We can call a Logistic Regression a Linear Regression model, but the Logistic Regression uses a more complex cost function, this cost function can be defined as the '**Sigmoid function**' or also known as the 'logistic function' instead of a linear function.

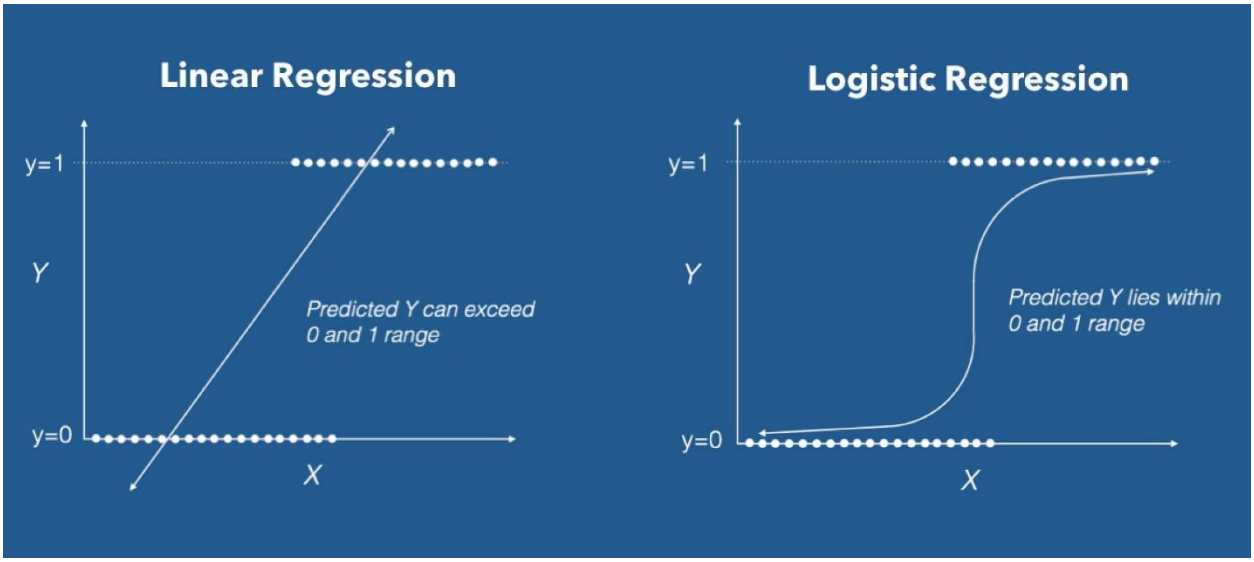

Figure 2: Linear vs Logistic regression

The hypothesis of logistic regression tends it to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.

$$0 \leq h_\theta(x) \leq 1$$

Figure 3: Logistic regression hypothesis expectation

Sigmoid Function

In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

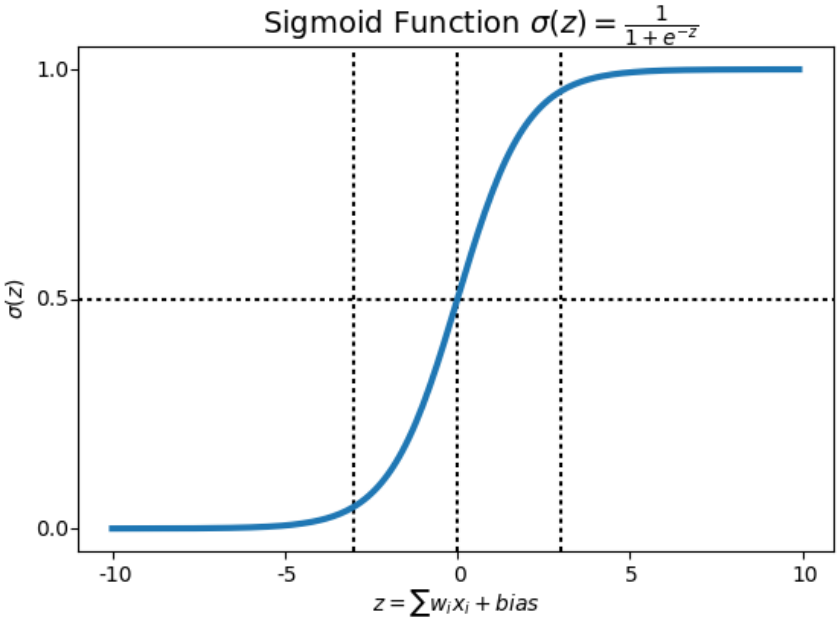

Figure 4: Sigmoid Function Graph

$$f(x) = \frac{1}{1+e^{-(x)}}$$

Figure 5: Formula of a sigmoid function

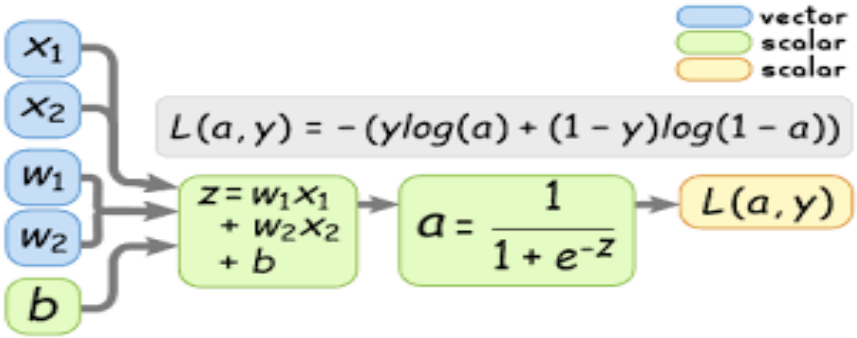

Figure 6: Computational graph of Logistic regression

here, w and x are vectors, whose size depend on the number of input features.

Cost Function:

Instead of Mean Squared Error, we use a cost function called Cross-Entropy, also known as Log Loss. Cross-entropy loss can be divided into two separate cost functions: one for y=1 and one for y=0.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\text{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \qquad \text{if } y = 1$$

$$\text{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \qquad \text{if } y = 0$$

Figure 7: Formula for Cost Function

Gradient-Descent:

To minimize our cost, we use Gradient Descent just like before in Linear Regression. There are other more sophisticated optimization algorithms out there such as conjugate gradient like BFGS, but you don't have to worry about these. Machine learning libraries like Scikit-learn hide their implementations so you can focus on more interesting things!

Gradient Descent

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

Want $\min_\theta J(\theta)$:

Repeat $\{$

$$\theta_j := \theta_j - \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

(simultaneously update all θ_j)

$\}$

Andrew Ng

Figure 8: Formula for Gradient Descent

Gradient

$$z = w_1 x_1 + w_2 x_2 + b \longrightarrow \hat{y} = a = \sigma(z) \longrightarrow L(\hat{y}, y)$$

$$\Leftrightarrow a = \hat{y}$$

$$w_1 \Rightarrow \frac{\partial(L)}{\partial w_1} = \frac{\partial L}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial(z)}{\partial w_1}$$

$$\frac{\partial L}{\partial a} = \frac{\partial}{\partial a}(-y \log a - (1-y)\log(1-a))$$

$$= -y\left(\frac{1}{a}\right) - (-1)\frac{(1-y)}{(1-a)}$$

$$\boxed{\frac{\partial L}{\partial a} = \left(\frac{-y}{a}\right) + \left(\frac{1-y}{1-a}\right)}$$

$$\boxed{\frac{\partial a}{\partial z} = a(1-a)}$$

$$\boxed{\frac{\partial z}{\partial w_1} = x_1}$$

Figure 9: Applying gradient descent .part -1

$$\frac{\partial L}{\partial w_1} = \left(\left(\frac{-y}{a} + \frac{(1-y)}{1-a}\right) \cdot (a)(1-a)\right) \cdot x_1$$

$$= (a-y) \cdot x_1$$

Update for w_1,

$$\frac{\partial L}{\partial w_1} = (a-y) \cdot x_1$$

Here, $(a-y) = \frac{\partial L}{\partial z}$

$$\boxed{w_1 = w_1 - \alpha \frac{\partial L}{\partial w_1}}$$

Similarly, for all parameters

$$\boxed{w_i = w_i - \alpha \frac{\partial L}{\partial w_i}} \qquad i = 1, 2, \ldots, m$$

m = no of parameters

$$\boxed{b = b - \alpha \frac{\partial L}{\partial b}}$$

Where, $\frac{\partial L}{\partial b} = (a-y)$

Figure 10: Applying gradient descent .part -2

5 Results:

Here, I trained the data with epoch as 500 and learning rate as 0.01 and the results are as follows:

```
cost after 480 iteration : 0.134490
cost after 490 iteration : 0.133538

Text(0,0.5,'Cost')
```

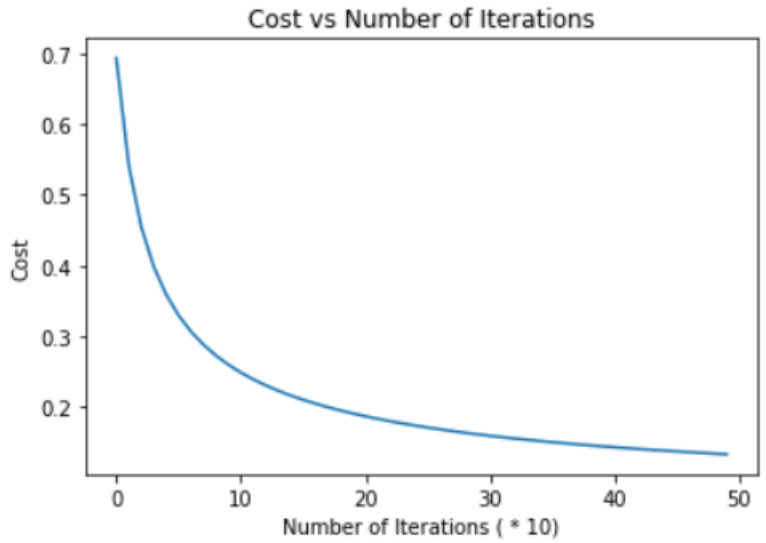

Figure 11: cost vs iterations graph for training data, learning rate is 0.01

```
Text(0,0.5,'predictions')
```

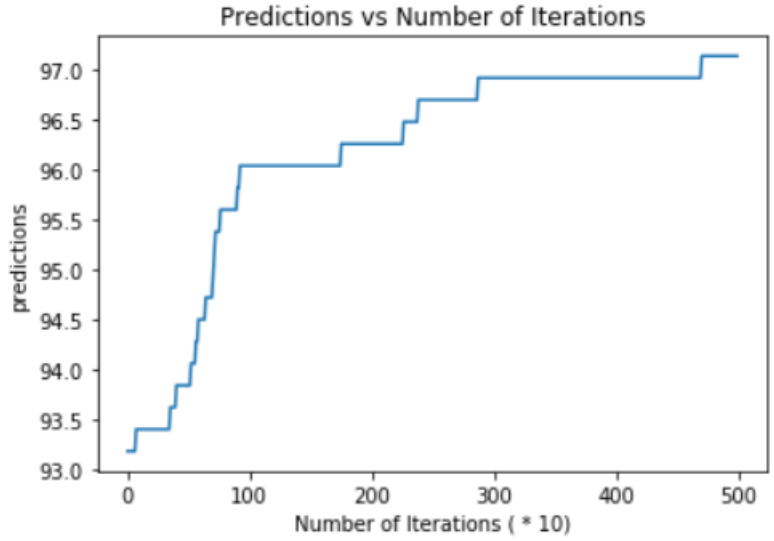

Figure 12: predictions vs iterations graph for training data, learning rate is 0.01

Validation Graphs

```
cost after 480 iteration : 0.095779
cost after 490 iteration : 0.094573

Text(0,0.5,'Cost')
```

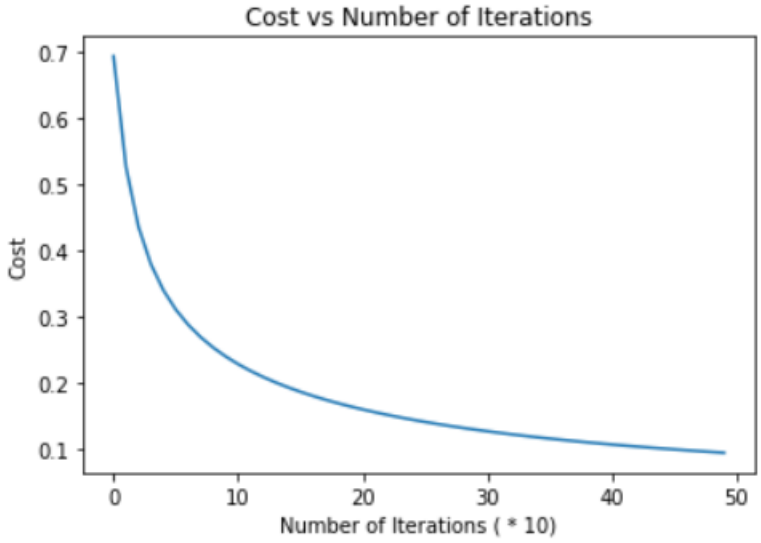

Figure 13: cost vs iterations graph for validation data, learning rate is 0.01

```
cost after 480 iteration : 0.032430
cost after 490 iteration : 0.031939

Text(0,0.5,'Cost')
```

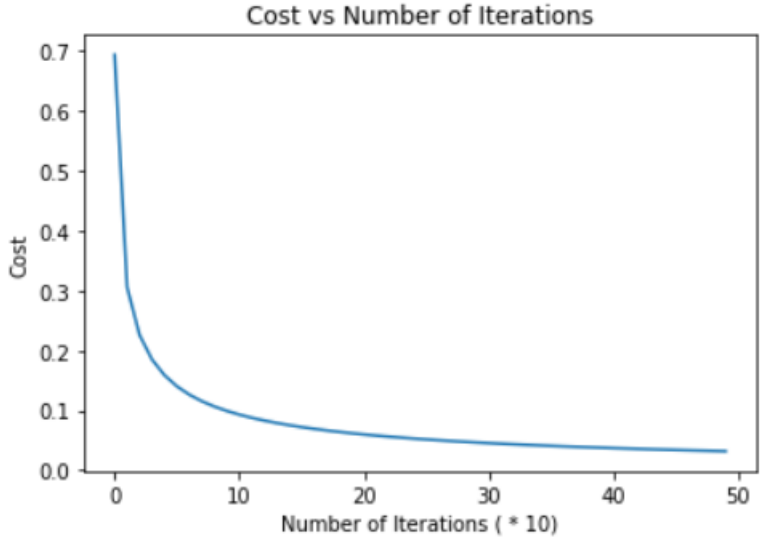

Figure 14: cost vs iterations graph for validation data, learning rate is 0.05

```
cost after 480 iteration : 0.020654
cost after 490 iteration : 0.020321

Text(0,0.5,'Cost')
```

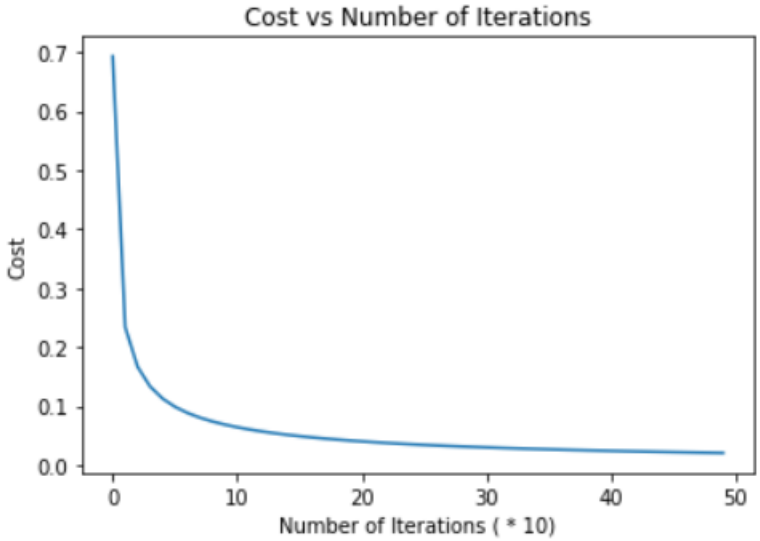

Figure 15: cost vs iterations graph for validation data, learning rate is 0.09

6 Conclusion:

I tested the dataset and got the following metrics:
Accuracy: 94.73684210526315

Precision: 83.33333333333334

Recall:100.0

References:

[1] https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

[2] http://ronny.rest/blog/post_2017_08_12_logistic_regression_derivative/

[3] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[4] https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

[5]https://medium.com/deep-math-machine-learning-ai/chapter-2-0-logistic-regression-with-math-e9cbb3ec6077