Two Class Classification using Logistic Regression in Python

Sai Charith Reddy Pasula
50320452
saichari@buffalo.edu
Department of Computer Science
University at buffalo

Abstract

In this project, we use machine learning to classify a type of cancer as benign or malignant. For this we train a model using the machine learning algorithm logistic regression. Logistic regression is used in classifying a dataset into different classes. Our problem deals with two class classification. The dataset that we use is the Wisconsin Diagnostic Breast Cancer(wdbc.dataset).

1 Introduction

Breast cancer is one of the most ubiquitous cancer that occurs in woman. If not detected at an early stage, this can lead to life threatening situation. Hence in this project we try to classify a type of tumor as benign or malignant. Benign tumor refers to the tumor that isn't cancer. i.e. the tumor doesn't spread with time. Malignant tumor refers to the tumor that is cancer i.e. this tumor spreads with time and is life threatening.

Classifying a tumor as benign or malignant refers to a two-class classification problem. We use logistic regression to classify the data into benign or malignant. This type of machine learning problem refers to supervised learning. i.e. you feed both the input value and the class label to the model and the model figures out the features of each class and classifies the incoming data into the respective class. The other type of machine learning problem we deal with is unsupervised learning in which the model itself forms clusters of the input data based on the features of the data.

Logistic regression deals with the use of a sigmoid function on the input hypothesis to compress the data between 0 and 1. We will look at logistic regression in much detail in the below sections.

I was successfully able to build a model and train it to classify the input into benign or malignant tumor. I was able to get an accuracy of 96% for the model.

1.1 Logistic Regression

Linear regression cannot be used for classification problem. Linear regression deals with continuous values of data and as we deal with discrete data, the linear regression algorithm must be edited to accommodate discrete values. For this reason we use logistic regression. Logistic regression applies a sigmoid function to the hypothesis of the linear function.

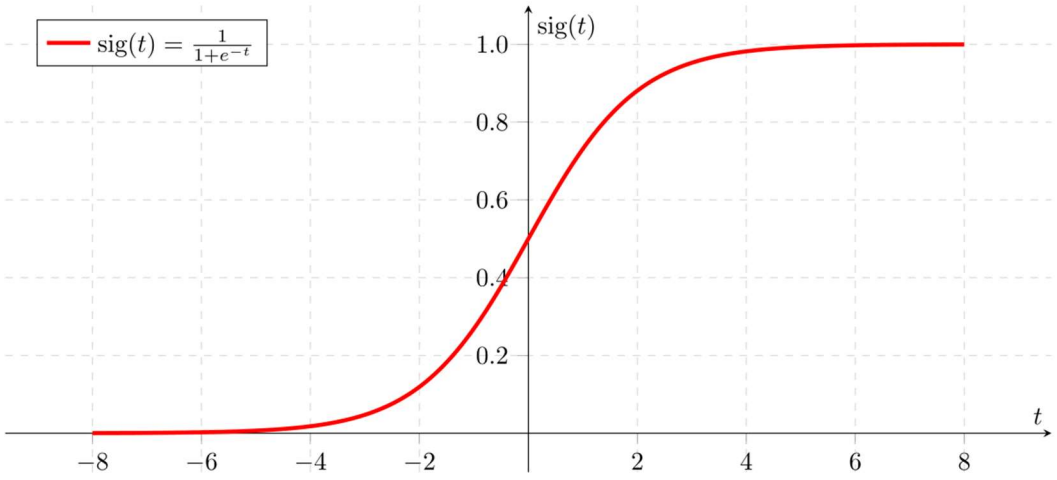


Figure 1: Sigmoid Activation Function.

As we can observe from figure 1, the sigmoid function "squeezes" the hypotheses so that the output is between zero and one. The hypothesis is in the form of Z=W.T.X + b. where W.T is the transpose of the weight matrix and b is the bias. As Z goes to infinity, Y(Predicted) will become 1 and if 'Z' goes to negative infinity, Y(Predicted) will become 0.

1.1.1 Decision Boundary

To predict to which class our predicted output belongs, a threshold can be set. For example, if the predicted value >= 0.5, we can classify the data as malignant else it is benign. Decision boundary is linear if we use a polynomial basis function. If we use a gaussian basis function, it can be non linear. Linear regression uses mean squared error as its cost function. If this is used for logistic regression, then it will be a non-convex function

1.1.2 Cost function

Linear regression uses mean squared error as its cost function. If this is used for logistic regression, then it will be a non-convex function of the parameters. Gradient descent only converges into global minimum only if the function is convex. For this reason, we use a log based cost function which is derived from binary cross entropy. The cost function of logistic regression looks like :

$$\text{Cost}(h\theta(x),y)=\begin{cases}-\log(h\theta(x)) & \text{if } y = 1\\ -\log(1-h\theta(x)) & \text{if } y = 0\end{cases}$$

This formulae can be simplified to the below form:

$$\text{Cost}(h\theta(x),y)=1/m\{-y\log(h\theta(x))-(1-y)\log(1-h\theta(x))\}$$

1.1.3 Gradient descent

Gradient descent is a first-order iterative optimization algorithm for finding the minimum of a function. In machine learning, we use gradient descent to update the parameters of our

model.

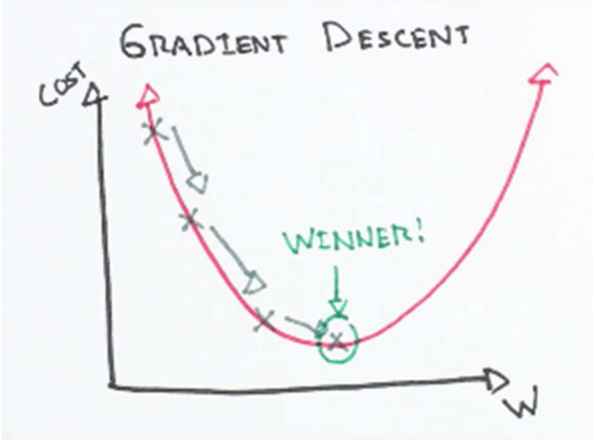


Figure 2: Gradient descent visulization.

We calculate the first order derivative of the function at a particular point that is equal to the tangent of the curve at a particular point in figure 2. By using the slope of the function, we can decide in which direction our point has to move. If the slope is negative, we move rightwards and vise versa. The step size that we take is called the learning rate. If the learning rate is too high, the point may "overshoot" and never converge to the global minimum. A low learning rate is better but it takes more time to converge.

1.1.4 Epoch

An epoch is defined as one iteration through the dataset. For each iteration, we update the weights and bias.

2 Dataset

The dataset provided to us is the Wisconsin Diagnostic Breast Cancer (WDBC) dataset. This 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 describes 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 ($perimeter^2/area - 1.0$)
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 5: Dataset representation

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 Preprocessing

I copied the entire dataset into a pandas data frame using the readcsv function. The Dataset given to us contains id's in the first column that can be removed. I dropped this column using the drop function included in the data frame. The dataset also contains the Benign and Malignant values(the output values) as B and M respectively. I used the pandas replace function to replace B as 0 and M as 1 to facilitate easy calculation. I divided the data set into three parts, training data(80%), validation data(10%) and testing data(10%) using the train_test_split function included in the sklearn.model_selection library. The training input parameters are present in the column 3 to 32 and the output parameters are present in the 2nd column. I segregated this using the loc function. I finally used the standard scaler present in the scikitlearn library to normalize the data. This helps in removing the mean and scaling to unit variance. It does so by calculating (X-U)/S where U is the mean of the training samples and S is the standard deviation of the training sample.

Second level headings are lower case (except for first word and proper nouns), flush left, bold and in point size 10. One line space before the second level heading and ½ line space after the second level heading.

4 Architecture

The computational graph of logistic regression can be visualized as follows

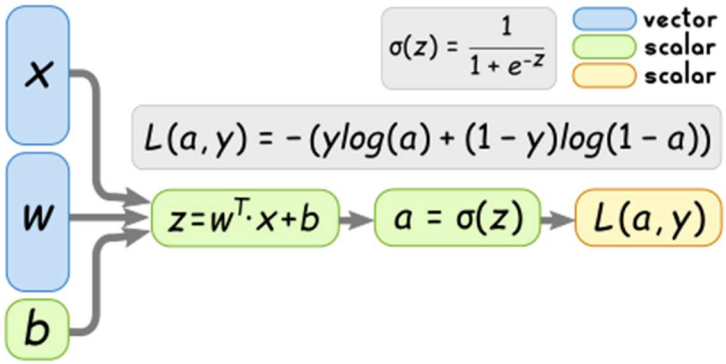


Figure 6: Computational graph of Logistic regression

Where x is the input vector and w is the weight vector. B is the bias in the model.

We apply gradient descent to above loss function to calculate:

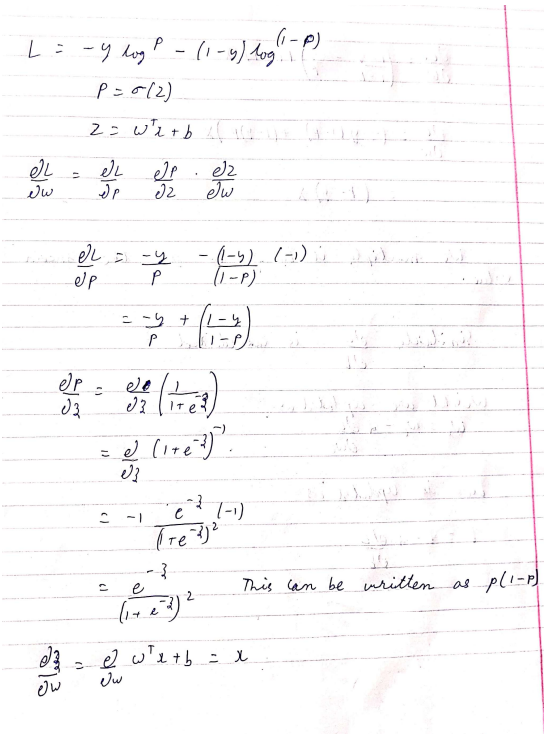


Figure 7.1: applying gradient descent to Logistic regression

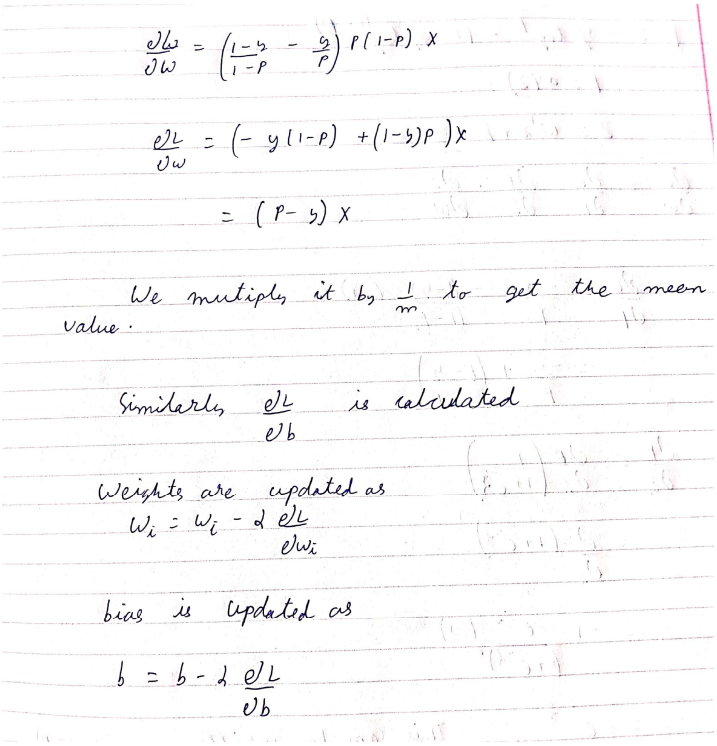


Figure 7.2: Computational graph of Logistic regression

As we can see in the above figure, the gradient of loss function to the input unit takes the

simple for of (p - y)*x where a is the predicted value, y is the expected output and x is the input data point. Further the change in weights and bias can be easily calculated with a few simple formulae. After updating the weights and bias, the model is again trained with the entire dataset for the next epoch. This continues until the convergence is reached or the number of epochs is exhausted.

The entire process of logistic regression can be condensed into a single figure shown below

for epoch in range(epochs):

$$z = w^T . x + b$$
$$p = s(z)$$
$$dz = p - y$$
$$dw = \frac{1}{m} X dz^T$$
$$db = \frac{1}{m} \sum_{k=1}^{m} dz^k$$
$$b := b - \alpha db$$
$$w := w - \alpha dw$$

Figure 8: Entire process of logistic regression.

5 Results

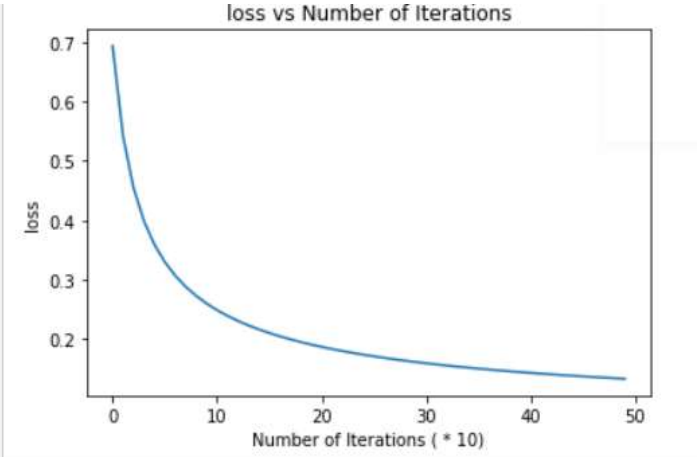
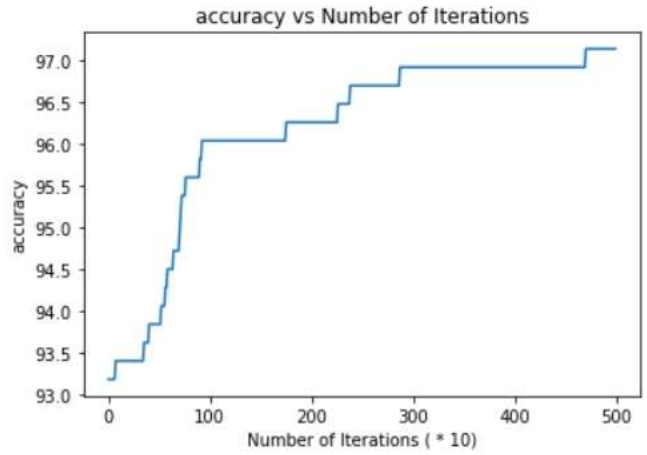
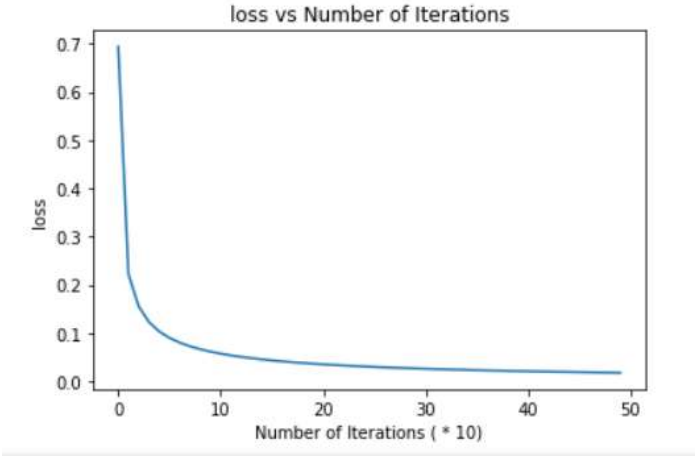


Figure 9: Training loss vs number of iterations for learning rate = 0.01



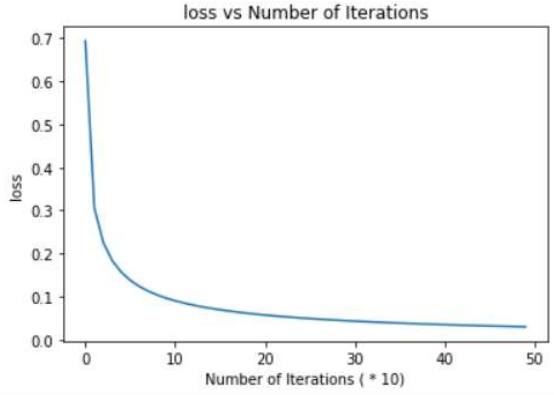
158

159 Figure 10: Training accuracy vs number of iterations for learning rate = 0.01



160

161 Figure 11: validation loss vs number of iterations for learning rate = 0.1 Cost = 0.017663



162

163 Figure 11: validation loss vs number of iterations for learning rate = 0.05 Cost = 0.030309

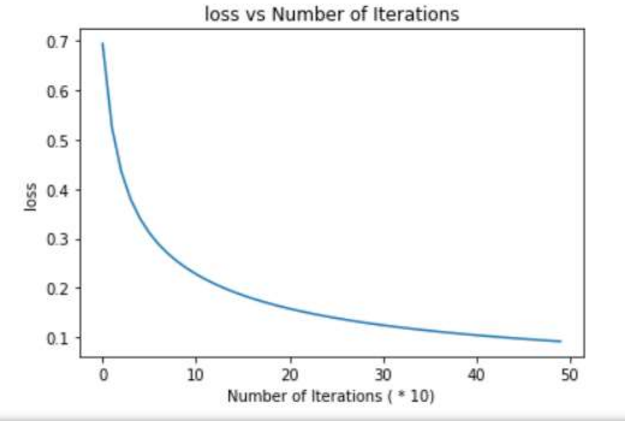


Figure 11: validation loss vs number of iterations for learning rate = 0.01 Cost = 0.092226

I initially trained the model for 500 epochs and learning rate of 0.01. Later I validated the model using the validation data and validating it for a learning rate of 0.01 for which I obtained a cost of 0.01. I used the validation dataset on the model by using the leanrning rate = 0.05 and obtained the cost = 0.03 I finally validated my model with a learning rate of 0.1 for which I obtained a cost of 0.017

6 Conclusion

I successfully trained the model using the training set validated it and finally tested it using the test data set. I was able to obtain an accuracy of 96.49122807017544 %, recall of 95.23809523809523 %, precision of 100.0 % and F-Measure of 97.5609756097561 %. This model will classify the type of tumor as benign or malignant with 96% accuracy.

References

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

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

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

[4]Derivatives for logistic regression-step by step http://ronny.rest/blog/post_2017_08_12_logistic_regression_derivative/

[5] Sklearn.Standard Scaler https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

[6] Confusion Matrix in Machine Learning https://www.geeksforgeeks.org/confusion-matrix-machine-learning/

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

[8] Gradient Descent https://ml-cheatsheet.readthedocs.io/en/latest/gradient_descent.html

[9] sklearn.metrics.confusion_matrix https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html