Classify suspected FNA Cells using Logistic Regression

Shivaprakash Rajshekar Hiremath
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
hiremath@buffalo.edu

Abstract

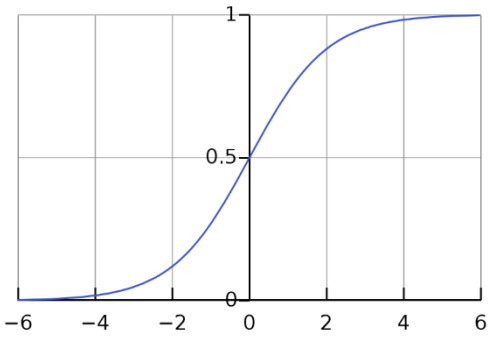
In this project, we study and build a logistic regression model for two-class classification problem. We are using WDBC cancer dataset for this project and we create a model by data partitioning, training using gradient descent, we tune hyper-parameters so that we achieve better performance on validation set. After all this we fix the hyperparameters and test our model's performance on the testing set. This will tell us the effective power gained by the learning.

1 Introduction

Logistic regression is a supervised classification algorithm. In a classification problem, the target variable i.e., output(Y), can take only discrete values for given set of features (or inputs), X. The model builds a regression model to predict the probability that a given data entry belongs to the particular category or not.

In order to map predicted values to probabilities, we use the sigmoid function (shown below). The function maps any real value into another value between 0 and 1.

As it is a cancer data set and we have to identify if the person is benign or Malignant which can be declared as 0 and 1 respectively. So we can build a model using logistic regression for this dataset which predicts whether a person is benign or malignant by providing all the necessary inputs to the model. In this project we mainly build our logistic regression model from scratch using python and evaluate its performance using accuracy, precision and recall.



Sigmoid Function graph

2 Dataset:

Wisconsin Diagnostic Breast Cancer (WDBC) 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 _ne 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.

3 Pre – Processing:

❖ Data reading, dropping column and mapping:

- We read the data from the dataset using the pandas library by calling the below function:

  pandas.read_csv

- Drop the ID column using the pandas library by calling the below function

  pandas.DataFrame.drop

- Map B, M to 0 and 1 respectively using the NumPy library by calling below function:

  numpy.where

❖ Data Splitting:

We divide the data into three categories (i) Training data (ii) Validation data and (iii) Test data.

Purpose of splitting data into three different categories is to avoid overfitting which is paying attention to minor details/noise which are not necessary and only optimizes the training dataset accuracy.

Validation dataset is used to select hyperparameters like bias. We are also using validation dataset to tune parameters in order to better validation dataset accuracy.

We need a model that performs well on dataset that it has never seen i.e., test data, So the actual measure of our model's performance should be tested on a dataset that has nothing to do with learning called test dataset.

- Split the data using sklearn library by calling below function

sklearn.model_selection.train_test_split

❖ Normalization:

We normalize data in order to compare two variables that have different scales or too different value range. Only when we remove the scale from both variables can we have a valid comparison. Example: comparing one variable measured in miles with another measured in inches.

- Normalize the data using sklearn library by calling below function

sklearn.preprocessing.MinMaxScaler

4 Architecture

Overview:

Logistic regression takes input and returns an output of probability, a value between 0 and 1. It does this with the help of a function called a logistic function or most commonly known as sigmoid. This sigmoid function is responsible for predicting or classifying a given input. After that, we try to optimize logistic regression and there are two common ways to approach the optimization of the Logistic Regression, one is through loss minimizing with the use of gradient descent which will be explained below in detail. After that we test our performance of the model by fixing the hyperparameters and results obtained are provided in the results section.

Definition of the logistic function:

$$z = \Theta^T x$$

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

x in the above definition is the input features and theta the weights.

Weights (represented by theta in our notation) plays a vital part of Logistic Regression and other Machine Learning algorithms and we want to find the best values for them. To start we assign 0 values and we need a way to measure how well the algorithm performs using those random weights. That measure is computed using the loss function.

The loss function is defined as:

$$h = sigmoid(X\Theta)$$
$$loss(\Theta) = \frac{1}{m} \cdot [-y^T log(h) - (1-y)^T log(1-h)]$$

The goal is to minimize the loss by means of increasing or decreasing the weights, which is commonly called fitting. Which weights should be bigger and which should be smaller can be decided by a function called Gradient descent. The Gradient descent is just the derivative of the loss function with respect to its weights.

The derivation of Gradient descent function is given as below:

$$J = \frac{-1}{m} \cdot [\sum_{i=1}^{m} y_i \cdot \log h_i + (1-y_i) \cdot \log 1 - h_i]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot [\sum_{i=1}^{m} \frac{y_i}{h_i} \cdot h_i^2 \cdot x_n \cdot \frac{1-h_i}{h_i} + \frac{1-y_i}{1-h_i} \cdot -h_i^2 \cdot x_n \cdot \frac{1-h_i}{h_i}]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{-1}{m} \cdot [\sum_{i=1}^{m} x_n \cdot (1-h_i) \cdot y_i - x_n \cdot h_i \cdot (1-y_i)]$$

$$\frac{\partial J}{\partial \theta_n} = \frac{1}{m} \cdot x_i \cdot [\sum_{i=1}^{m} h_i - y_i]$$

Gradients

After the gradient is calculated the weights are updated by subtracting the derivative (gradient descent) times the learning rate, as defined below:

$$\Theta := \Theta - \alpha \cdot \frac{\delta loss(\Theta)}{\delta \Theta}$$

We then validate our data with the updated theta(weights) and bias and calculate the validation loss.

We also test our data with the updated theta(weights) and bias and measure the accuracy, precision and recall of the model on the output of that data.

Decision boundary

Our current prediction function returns a probability score between 0 and 1. In order to map this to a discrete class (1/0), we select a threshold value above which we will classify values into class 1 and below which we classify values into class 2. For example, if our threshold was 0.5 and our prediction function returned .7, we would classify this observation as positive. If our prediction was 0.2 we would classify the observation as negative.

5 Results:

The task to be performed is:

1) Calculate accuracy, precision and recall of the model
2) Plot graph showing below details:
   I. accuracy vs number of epochs
   II. loss vs number of epochs

1)

Confusion Matrix: It is nothing but a tabular representation of Actual vs Predicted values. This helps us to find the accuracy of the model and avoid over-fitting. This is how it looks like:

Actual	Predicted	
	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

	Learning Rate	Epoch	Accuracy	Precision	Recall
Trial 1	**0.01**	**1500**	**89.47%**	**100%**	**68.42%**
Trial 2	**0.03**	**1200**	**92.98%**	**100%**	**78.94%**
Trial 3	**0.1**	**1000**	**96.49%**	**100%**	**89.47%**

Using trial and error method we input different values for hyperparameter like epoch and learning rate and observe the loss and other parameters, to judge the best learning rate and epoch for the model.

From the above three trials we can observe that learning rate with 0.01 value has lower accuracy rate and from the graph below we can observe that it has a very big loss values so do not choose it.

Similarly, from the remaining two trials learning rate with 0.03 has a lower accuracy and has a higher loss than the other learning rate 0.1. So, it is one of the reasons we chose 0.1. The other reason is as below:

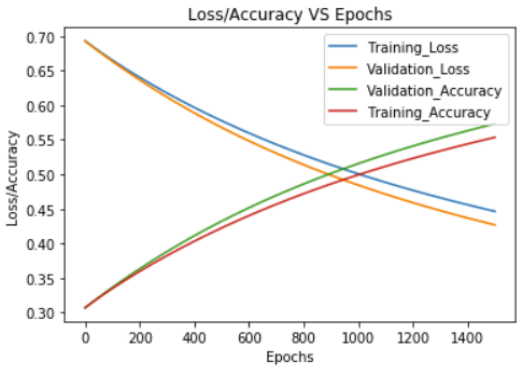
Learning rate of 0.1 also has the better recall rate than the other two. This is also one of the reasons because we do not want any affected patient to be classified as not affected without giving much heed to if the patient is being wrongfully diagnosed with cancer. This is because, the absence of cancer can be detected by further medical diseases but the presence of the disease cannot be detected in an already rejected candidate.

2) Graphs:

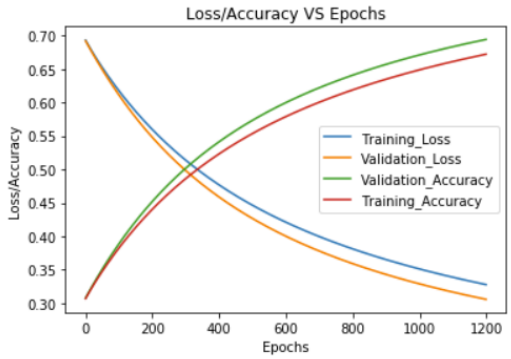
I accuracy vs number of epochs

II loss vs number of epochs

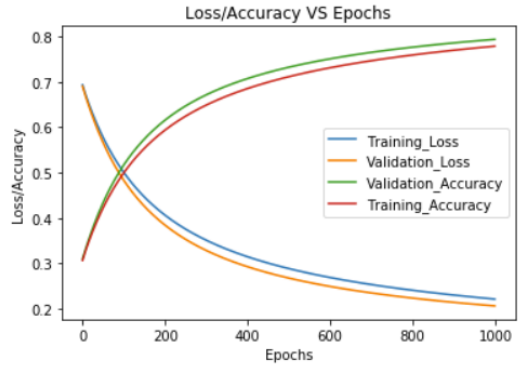
```
TP:  13.0  TN:  38.0  FP:  0.0  FN:  6.0
Learning Rate:  0.01
Accuracy:  0.8947368421052632
Precision:  1.0
Recall:  0.6842105263157895
```




```
TP:  15.0  TN:  38.0  FP:  0.0  FN:  4.0
Learning Rate:  0.03
Accuracy:  0.9298245614035088
Precision:  1.0
Recall:  0.7894736842105263
```




```
TP:  17.0  TN:  38.0  FP:  0.0  FN:  2.0
Learning Rate:  0.1
Accuracy:  0.9649122807017544
Precision:  1.0
Recall:  0.8947368421052632
```



6 Conclusion

For the given problem statement as it was a binary classification problem we used logistic regression and used sigmoid activation function, as the values should lie in between 0 and 1 and should classify into class 0 or class 1.

For the given data we normalized, split the data, trained and validated it. Later we used the obtained values of weights and bias from the trained data to perform the test to measure the accuracy, precision and recall. We also performed different trials for different hyperparameters like epoch and learning rate and chose the best fit for our model.

The optimal value that we chose from the above results are
learning rate = 0.1 and **epoch = 1000**.
For the above values the **Accuracy** is **96.49%, Precision** is **100% and Recall** is **89.47%.**

7 Acknowledgements

We are extremely grateful to Professor Sargur Srihari for elucidating all the necessary concepts related to Machine Learning. We also thank all the TAs for their efforts taken in clearing our doubts/questions.

8 References

[1] Pattern Recognition and Machine Learning (PRML) by Chris Bishop (Springer 2006)

[2] https://www.coursera.org/learn/machine-learning/home/week/3

[3] https://hackernoon.com/introduction-to-machine-learning-algorithms-logistic-regression-cbdd82d81a36

[4] https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

[5] https://pandas.pydata.org/pandas-docs/stable/

[6] https://scikit-learn.org/stable/modules/classes.html

[7] https://docs.scipy.org/doc

[8] http://www.win-vector.com/blog/2011/09/the-simpler-derivation-of-logistic-regression/

[9] https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html

[10] https://www.kaggle.com/jeppbautista/logistic-regression-from-scratch-python