Performing Classification for a two-class problem using Logistic Regression

Upasana Ghosh
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
upasanag@buffalo.edu

Abstract:

The purpose of this project is to design a Logistic Regression Classifier Model using Gradient Descent to perform classification on a two class Classification Problem. We are provided with the Wisconsin Diagnostic Breast Cancer (WDBC) dataset which contains 569 instances with 32 attributes. The features used for the classification are pre-computed from images of a fine needle aspirate (FNA) of a breast mass. We would be classifying the suspected FNA cells to Benign (class 0) or Malignant (class1) using logistic regression. For this, I have preprocessed the original dataset by manipulating the label field value and converting it to 0 and 1 to represent Benign and Malignant respectively. All the redundant data has been removed and the dataset has then been divided into training, validation and testing dataset to train and evaluate the model. The graph that shows the cost and accuracy trends with epoch has been leveraged to verify if the tuning of the hyperparameters are correct. Finally, the accuracy, recall and precision on the testing data set has been calculated to validate if the model can predict the correct outputs for unknown datasets in the future.

Introduction:

Supervised Learning is a machine learning technique in which we associate a set of inputs with a set of targets in a given dataset. We broadly have two types of Supervised Learning problems, “Regression” and “Classification”. A Classification problem is a problem in which we divide the input data into discrete categories or separate classes. There are two main types of classification problems:

- Binary classification: where we divide our data set broadly into two discrete categories or classes.
- Multi-class classification: where we map the data in our data set into multiple discrete classes or categories

In order to solve or to perform classification on a classification problem, we use a classifier. A classifier is an algorithm that implements classification, especially in a concrete implementation. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category. Here we are presented with a Binary Classification problem and I have used Logistic Regression as my classifier to solve the given problem. Logistic Regression is a statistical method for analyzing a dataset in which there are one or more independent variables that determine an outcome. The outcome is measured with a dichotomous variable (in which

there are only two possible outcomes). Rather than modelling a response directly, logistic regression models use the probability that the response belongs to, to map the response to a particular category with the help of a Sigmoid Function. For this experiment, I have used Batch Gradient Descend to implement the Logistic Regression. Gradient Descent is an iterative optimization algorithm for finding the minimum of a function. The algorithm takes steps proportional to the negative gradient of the function at the current point. For this, we define a loss function and optimize the parameters (weights and bias in our case) of the network to obtain the minimum of the function. There are two ways of performing Gradient Descent i.e., Stochastic Gradient Descent and Batch Gradient Descent. The concept of Batch Gradient Descent is the same as that of Stochastic Gradient Descent except for the fact that we update the network parameters after all the training examples in a batch have been passed through the network instead of calculating the loss after every training sample in the training set.

To train and evaluate the Logistic Regressor Model that I'm designing here for the Wisconsin Diagnostic Breast Cancer (WDBC) dataset, I have divided the dataset into 'Training', 'Validation' and 'Testing' datasets, and have used the confusion matrix to calculate the accuracy, precision and recall for each epoch or iteration. I have also used the following graphs to evaluate the model further:

- Training Error and Validation Error VS Epoch
- Training Accuracy and Validation Accuracy VS Epoch
- Validation Accuracy VS Learning Rate

Dataset:

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

	Characteristics
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)

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

Preprocessing:

The original csv data file has been processed into a Pandas Data frame to extract the feature values and image IDs from the data. The label column in the data that gives the output 'M' or 'B' for a record has been replaced with '0' and '1' accordingly for the ease of computation. In order to scale the real valued inputs and compare the output with the actual output. The first column or the ID column in the data has been dropped to reduce data redundancy. The data frame has been partitioned into a training, validation and testing data set by randomly selecting 80% of the data for training data, 15% for validation and 5% for the testing data. The output for each of the datasets has been extracted out into separate lists and the data frame containing the input features has been normalized for the ease of computation, using the preprocessing library provided by Scikit-Learn.

Architecture:

The binary Classification Problem provided is solved using Logistic Regression. The Logistic Regression is implemented here using Batch Gradient Descent. The generalized model of a Logistic Regression Classifier can be shown as below:

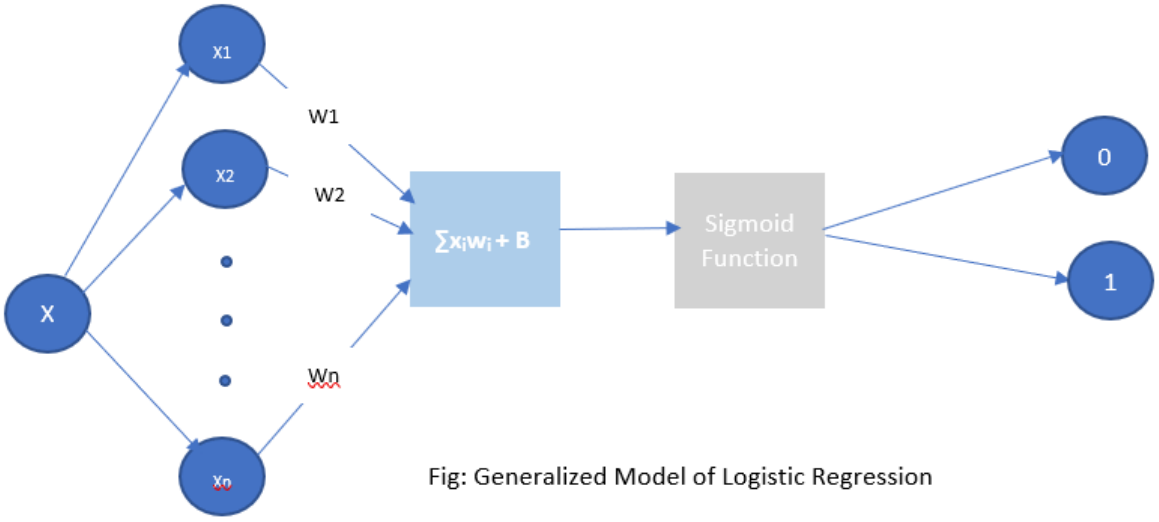


Fig: Generalized Model of Logistic Regression

We use the Sigmoid function to scale the real valued predicted outputs and compare it with the actual output of 0 and 1 in the provided dataset.

The prediction function (after passing the predicted output through the Sigmoid function) returns a probability between 0 and 1 for each data x in X. In order to map this to a discrete class we select a threshold value, above which we will classify the data into class 1 and below which we classify the data into class 2.

$$p \geq 0.5, class=1$$

$$p < 0.5, class=0$$

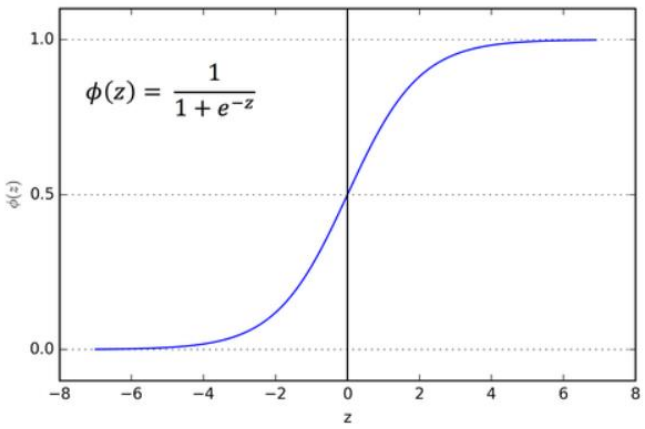


Fig: Sigmoid Function

To implement the Batch Gradient Descent, we have divided the testing data set into multiple batches (batch size = 46) and for each epoch, we calculate the Z = ($W^T X$) + b is for each batch in X (569x30 dimensions) and pass it through the Sigmoid Function:

$$\sigma(Z) = \frac{1}{1+e^{-z}},$$

Here Z is the genesis function, W (30x1 dimensions) is the weight associated with the input dataset and b is the bias. We need to initialize the vector W with either some random values or as a zero vector. For this experiment, I have initialized W with random values and have arbitrarily chosen a value for the bias. By adjusting the learning rate, we tune these hyperparameters. We change the value of the weight W and the bias b for each data in the batch and use the final value from one epoch in the next epoch. While calculating the Gradient Descent and tuning the values of W and b, we need to make sure that the value given by the Cross-Entropy function (Loss function) which denotes the delta between the predicted output and the actual output for the validation data decreases.

Considering the total epoch as 1000 for this experiment, for each epoch we calculate:

$Z = W^T X + b$

$p = \sigma(Z)$

$$L = \frac{-(ylogp + (1-y)\log(1-p)}{m}$$

Where L is the Cross-Entropy Function (i.e., Loss Function)

$$= \frac{-(ylog(\sigma(Z)) + (1-y)\log(1-\sigma(Z)}{m}$$

Taking the partial derivative of L,

$$\frac{\partial L}{\partial \theta} = \frac{\partial}{\partial \theta}\left(-\frac{1}{m}\left((y log\big(\sigma(Z)\big) + (1-y)\, log\big(1-\sigma(Z)\big)\right)\right)$$

$$= -\frac{1}{m}\left(\left(\frac{y\,\sigma(Z)\big(1-\sigma(Z)\big)}{\sigma(Z)}\right)\mathrm{X1} + \left(\frac{(1-y)(\sigma(Z)\big(1-\sigma(Z)\big)}{1-\sigma(Z)}\right)\mathrm{X1}\right)$$

$$= -\frac{1}{m}\left(y\big(1-\ \sigma(Z)\big)\mathrm{X}_1\right.$$

$$= \eta\ \left(-\frac{1}{m}\left(y\big(1-\ \sigma(Z)\big)\mathrm{X}_1\right)\right.$$

We use the above derived form of the Cross-Entropy Function L to tune our parameters W and b for the different Learning Rates.

In this experiment, I'm using a confusion matrix to evaluate the model. The Confusion Matrix gives us the count of the True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) in the predicted output. We can define the TP, TN, FP and FN as follows:

		PREDICTED	
		Positive	Negative
ACTUAL	Positive	TP	FN
	Negative	FP	TN

Using these values, we calculate the Accuracy, Precision and Recall for the training, validation and testing data sets with the help of the following formulae:

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

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

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

The trend in the accuracy, precision and recall values show us whether we are tuning the hyperparameters correctly and finally help us judge if our model is trained enough to predict the correct output for an unknown dataset in the future.

Results:

- The final value of W^T and the bias B for which I have evaluated my Logistic Regressor Model (by calculating the accuracy in predicting the output for the Testing dataset) are as follows:

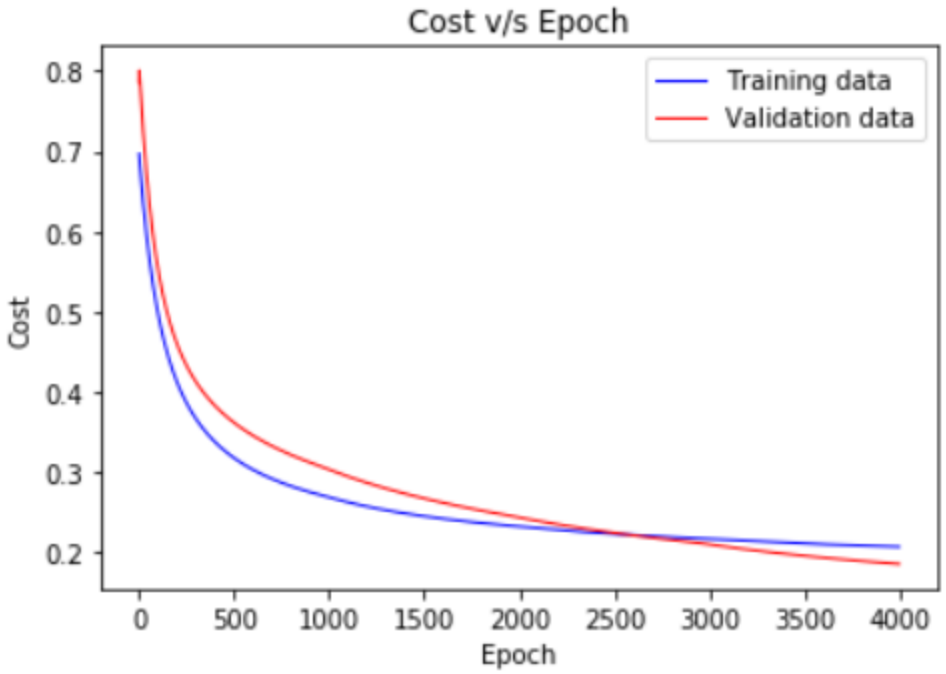
```
W-Transpose:
[[ -8.13525337  -8.56537356 -38.70153118 -22.31367156  -1.12360184
    1.28155251   0.49962732  -1.03553144  -2.42513417  -1.37590583
   -0.54761202  -1.21532989  -0.55471029  15.83134213   0.16574996
   -1.66589432   0.19839161   0.69079106   1.00045775  -0.30512079
   -6.3865632  -12.10599028 -38.54894742  25.62282146   0.56584369
   -0.17415014   2.15189436   0.94330151  -1.35173683  -0.29007775]]

Bias: 0.9187092339361977
```

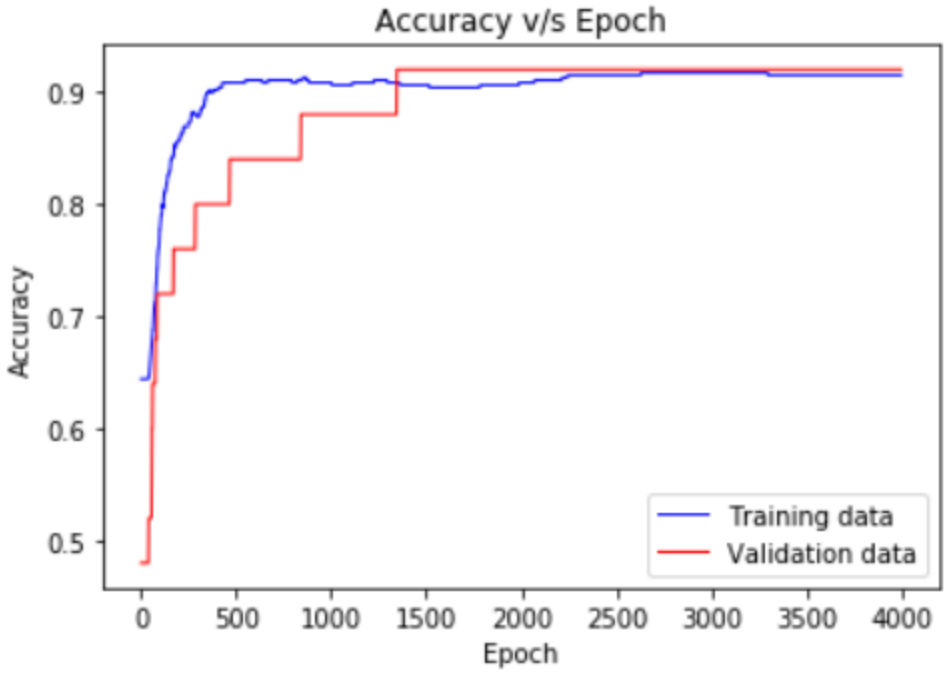
- The accuracy, precision and recall calculated on the Testing dataset:

```
ACCURACY CALCULATED ON TEST DATA: 0.9302325581395349
PRECISION CALCULATED ON TEST DATA: 0.96875
RECALL CALCULATED ON TEST DATA: 0.8611111111111112
```

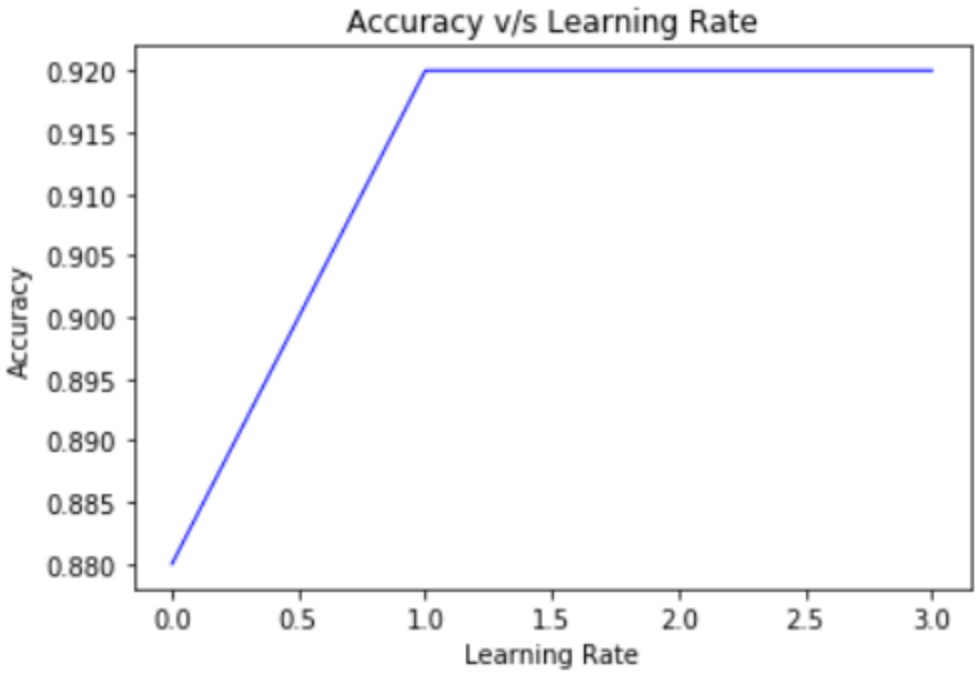
- The cost v/s Epoch graph for the training and validation data:



- The Accuracy v/s Epoch graph for the validation data:



- The Cost v/s Epoch graph for the validation data:



Conclusion:

This experiment to design a Logistic Regression Classifier for a two class Classification problem provides us with an insight about how we can use Logistic Regression and Gradient Descent for modelling classifiers to solve classification problems involving real world data sets. From the graphs in our result, we can conclude the following:

- The Accuracy v/s Epoch graph shows that with the increase in the number of epochs, the difference between the predicted output and the actual output decreases and after a certain iteration the predicted output becomes almost same. Hence, it can be said that if our model is well trained, it is capable of predicting the correct output for unknown data sets.

- The cost v/s epoch graph shows that as we tune the hyperparameters correctly with the increasing iterations, the cost becomes lower and the cost of training data set starts resembling the cost of the validation data set which means that the loss reduces as we tune our hyperparameters correctly.

- The accuracy calculated over the testing data set shows how accurately our model can predict the output for a set of unknown data set in the future.

Acknowledgement:

1. An Introduction to Statistical Learning by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani
2. https://medium.com/@divakar_239/stochastic-vs-batch-gradient-descent-8820568eada1
3. https://www.statisticssolutions.com/what-is-logistic-regression/
4. https://towardsdatascience.com/a-study-of-classification-problems-using-logistic-regression-and-an-insight-to-the-admissions-ec69ddf93f36
5. https://www.cs.cmu.edu/~kdeng/thesis/logistic.pdf
6. https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623