Naman Jain Vimal Kumar
1001670153

REPORT

The following report consists of solutions to Batch and Online Processing Techniques used to learn weights in Logistic Regression.

A **Python** program has been developed which performs logistic regression based on different methods and learning rates. The methods could be BATCH or ONLINE, and the learning rates could be a floating-point number. Initially, the program generates multivariant random class data based on given mean and standard deviation. The following can be changed under class definitions, that is definition of ***class1*** and ***class2***.

As we know that a bias term plays an important role in logistic regression, the generated data set is also appended with a **bias attribute** of 1. This helps us to classify classes away from the origin too, thus normalization is not a required.

Based on the user specification, the program further proceeds to particular logistic regression technique along with the **learning rate**. Once the training is completed, the program also does the tests based on the generated weights to classify the same into particular class. The Accuracy of the weights is generated and also, a **ROC curve** is plotted which accounts to True Positive Rate vs. False Positive Rate. Along with the curve, the **area under the curve** which represents accuracy is also generated.

With the help of ROC curve, we can study the pros and cons of different algorithms. This helps us to decide on the best algorithm that can be used for classification.

Notes:

- For both BATCH and ONLINE, the classification function used is sigmoid function.
- The Iteration limit for both is 10000
- There are three stopping criteria:
  - Cross Entropy
  - Gradient
  - Iteration Limit
- The size of testing data is half the size of training data

To run the program, we could use the following for our command line:
For BATCH Processing:

$$\$\, python\ main.py\ BATCH\ [learning - rate]\ [training - size]\ [roc]$$

For Online Processing:

$$\$\ python\ main.py\ ONLINE\ [learning - rate]\ [training - size]\ [roc]$$

Here:

learning-rate should be a number
training-size should be an integer
roc is written if an ROC curve is required

PROBLEM

Implement a perceptron for logistic regression. For your training data, generate 2000 training instances in two sets of random data points (1000 in each) from multi-variate normal distribution with μ1 = [1, 0], μ2 = [0, 1.5], Σ1 = [1, 0.75],[0.75, 1] , Σ2 = [1, 0.75],[0.75, 1] and label them 0 and 1. Generate testing data in the same manner but include 500 instances for each class, i.e., 1000 in total. Use sigmoid function for your activation function and cross entropy for your objective function. You will implement a logistic regression for the following questions. Initialize the starting weight as w = [1, 1, 1]. During training, stop your loop when the objective function (i.e., cross entropy) does not decrease any more (below certain threshold) or when the gradient is close to 0 or the iteration reaches 10000. Set your thresholds properly so that the iteration doesn’t reach 10000 for all the learning rate that you will be using.

Naman Jain Vimal Kumar
1001670153

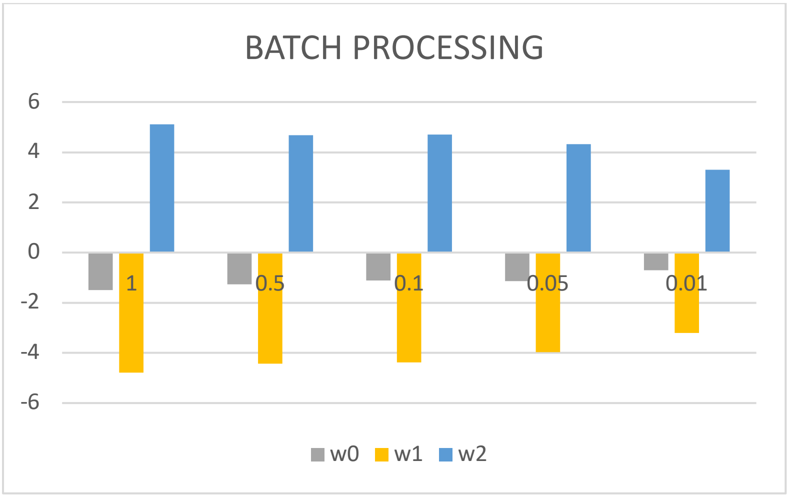
PROBLEM 1:

Perform batch training using gradient descent. Divide the derivative with the total number of training dataset as you go through iteration (it is very likely that you will get NaN if you don't do this.). Set your learning rate to be η = {1, 0.1, 0.01}. How many iterations did you go through the training dataset? What is the accuracy that you have? What are the edge weights that were learned?

Table 1 : Batch Processing for different learning rates

η		*1*	*0.5*	*0.1*	*0.05*	*0.01*
Accuracy (%)		95.1	95.2	96.7	96.5	96.4
Iterations		2181	3150	10000	10000	10000
Weights						
	w_0	- 1.491	- 1.261	- 1.116	- 1.132	- 0.717
	w_1	- 4.784	- 4.442	- 4.384	- 3.982	- 3.207
	w_2	5.119	4.676	4.695	4.313	3.299

OBSERVATIONS:



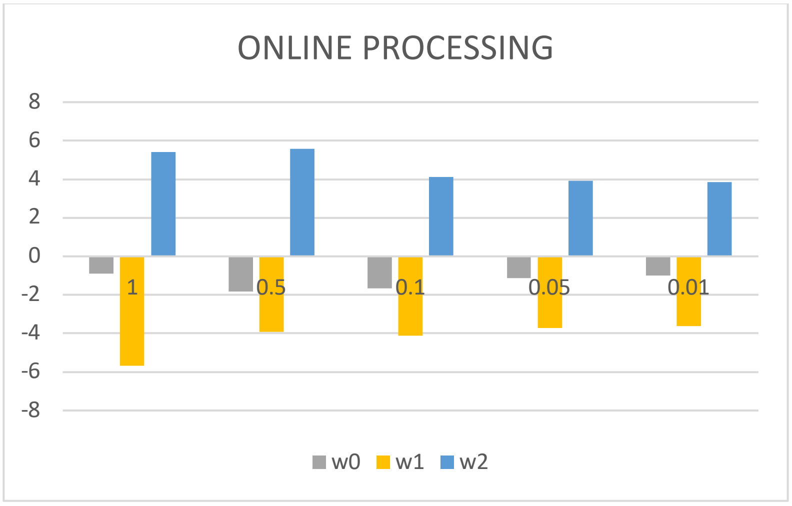
PROBLEM 2:

Perform online training using gradient descent. Set your learning rate to be η = {1, 0.1, 0.01}. Set your maximum number of iterations to 10000. How many iterations did you go through your training dataset? What is the accuracy that you have? What are the edge weights that were learned? Compare the learned parameters and accuracy to the ones that you got from batch training. Are they the same? Explain in your report.

Table 2 : Online Processing for different learning rates

η		*1*	*0.5*	*0.1*	*0.05*	*0.01*
Accuracy (%)		96.6	95.6	95.5	96.8	96.3
Iterations		1	1	2	2	9
Weights						
	w_0	- 0.909	- 1.824	- 1.646	- 1.137	- 0.987
	w_1	- 5.685	- 3.911	- 4.123	- 3.734	- 3.605
	w_2	5.411	5.591	4.130	3.904	3.864

OBSERVATIONS:



REPORT:

By seeing the above graphs, we can say that:

- The weight to the Bias attribute and first attribute is always negative.
- The weight of the second attribute is positive.
- A relationship between the weights is seen.
- Online Processing takes less iterations when compared to Batch Processing.
- We see faster convergence with higher learning rates.

PROBLEM 3:

Write your own code to draw ROC curve and computing area under the curve(AUC). Evaluate the performance of your LR classifier with batch training with η = {1, 0.1, 0.01}.

The ROC Curves are given below:

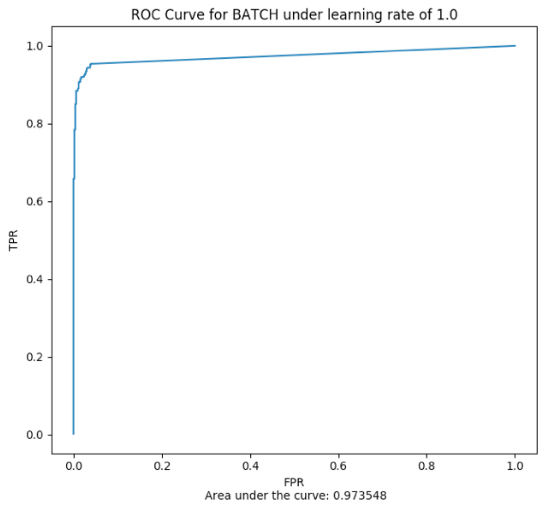


Figure 1 ROC Curve for Batch Processing with learning rate 1.0

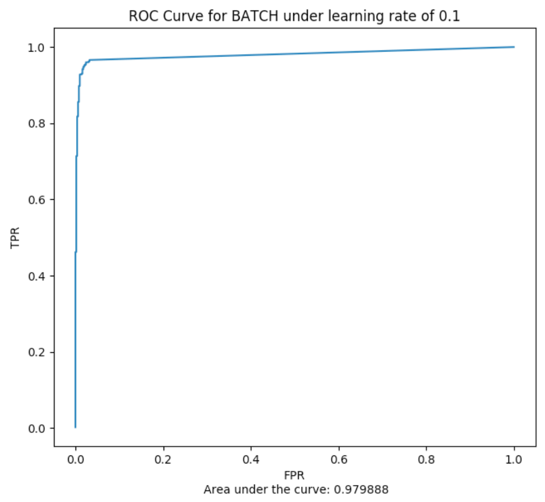


Figure 2 : ROC Curve for Batch Processing with learning rate 0.1

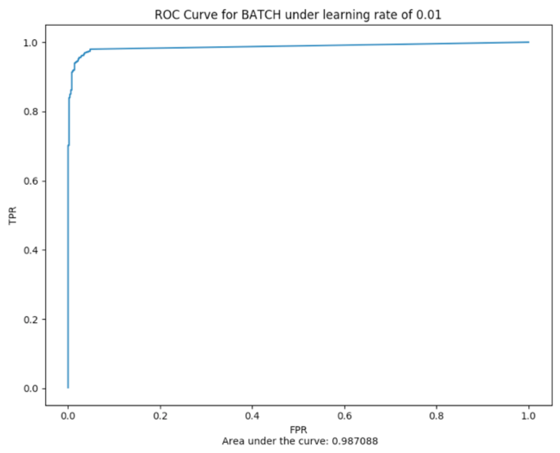


Figure 3 : ROC Curve for Batch Processing with learning rate 0.01

We know that that algorithms are classified best if they have greater value of Area under the curve. Here the Area under the curve (AUC) for the learning is equal to –

η	*1*	*0.1*	*0.01*
Accuracy	96.1	96.8	97.1
AUC	0.973548	0.978888	**0.987088**

We can see that the AUC for *η = 0.01* is greater than the other learning rates. Thus we can say that the best classifier among the three with *η = 0.01*. This can also be observed by their accuracy generated.

Note:

- To generate the area under the curve, trapezoidal rule has been used, which is the inbuilt function of numpy library.
- All the results shown in the report have their executed file in results.txt (Reference)