PROGRAMMING ASSIGNMENT #2

Perceptron, Gradient descent and Newton's Algorithm for Logistic Regression

Amrita Sundari V

1. Introduction

The programming assignment 2 aims at implementing different algorithms like Perceptron, Gradient Descent and Newton's method for Logistic Regression.

2. Getting familiarized with the data set

The Data set consist of two folders consisting the training and testing data. The training set is a text file consisting of 200 data points with 74 features in each data point, the first column being the label of the each point. Similarly, the testing set is also a text file with 76 data points..

I have implemented the codes LoadingDataset.py to vectorize the dataset and store it in a numpy array.

3. Implementation of Perceptron algorithm for Logistic Regression.

3.1 Perceptron algorithm

The perceptron algorithm is implemented for three sets of data sets namely

- The raw data
- L-1 Normalised data

- L-2 Normalised data

The number of iterations , the perceptron algorithm takes to converge has been drastically reduced in the case of the L-2 normalised data. Since, the data is a linearly separable data the error rate of both training and test samples converges to 0 after a certain number of iterations.

The plots of the training error rates with respect to the number of iterations is as shown in fig.1. As we can see the error rate of L2 normalised data converges quickly at around the 200th iteration.

Fig.2 is the plot of the testing error rates with respect to the number of iterations. L2 normalised data performs the best even in the testing data.

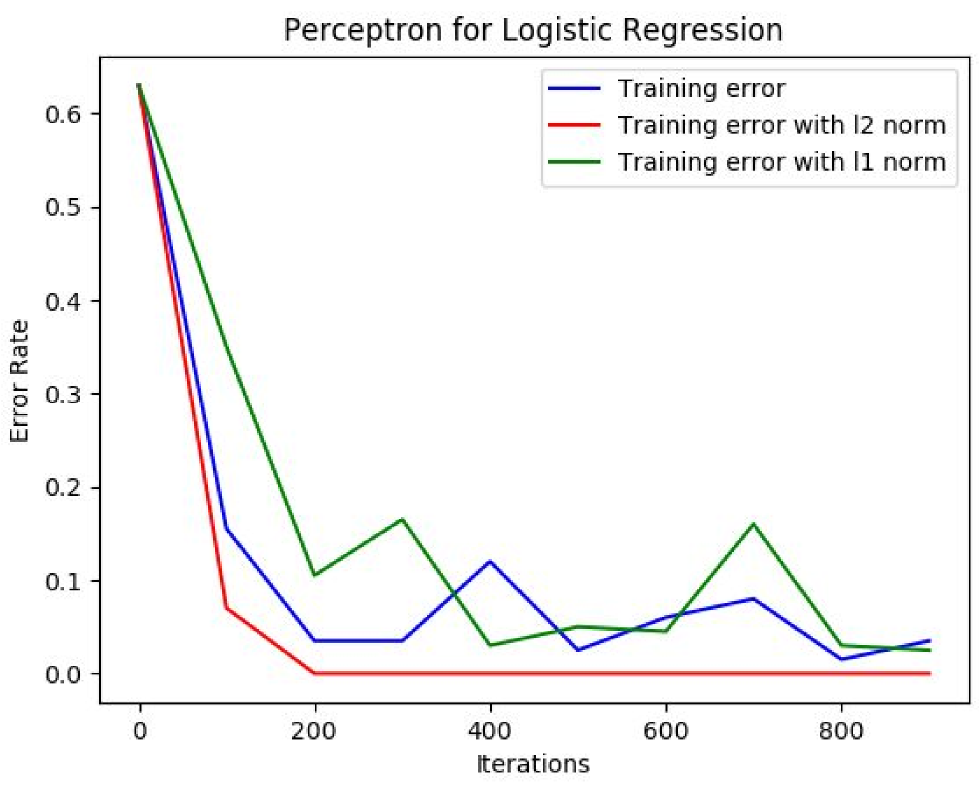

Figure 1

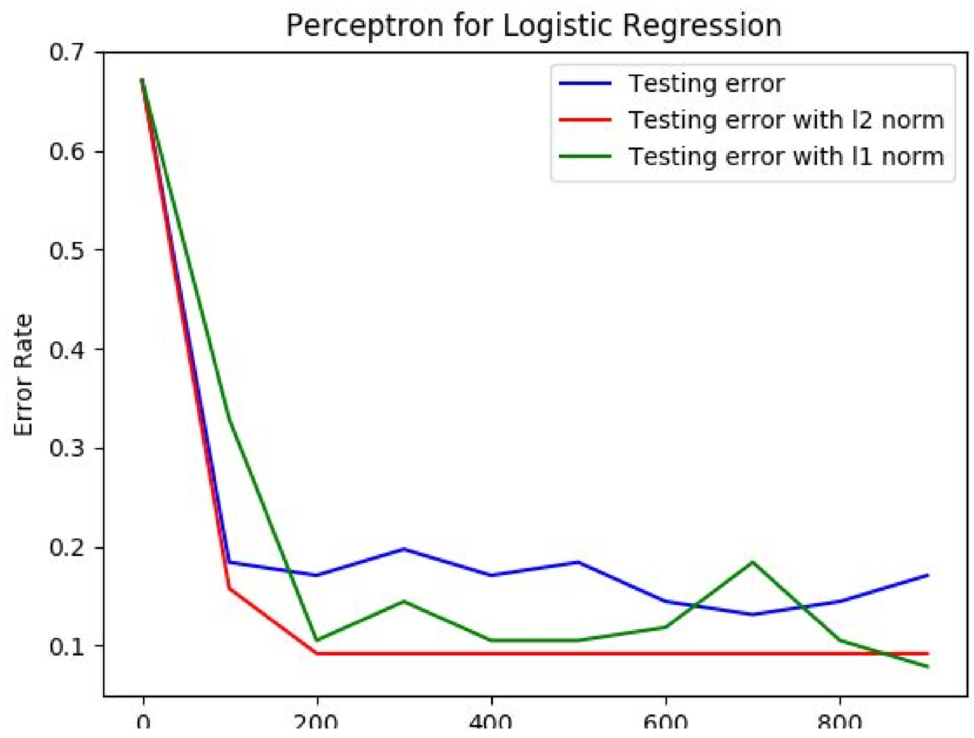

Figure 2

3.2 Averaged Perceptron algorithm

The averaged perceptron algorithm takes the weighted average of the parameters of all iterations. So in this case, the extra job is to store the parameters and the weights after each iteration.

Therefore training stage code remains the same but in the prediction stage, we consider all the parameters we have stored and take a weighted sum of them and use that resulting sum parameter as the model parameter and predict the values for the data point.

The weights of the parameters are decided according to number of misclassifications that particular parameter makes. Therefore more the misclassification, lesser the weight for the parameters. This will ensure that we take the best model parameters into account for the final classification problem. Therefore it is better than the previous vanilla perceptron and will converge faster than that.

Figure 3 and Figure 4 shows the training and testing error rate with respect to number of iterations. As we can see, the error rate of l2 normalised data converges faster than the vanilla perceptron.

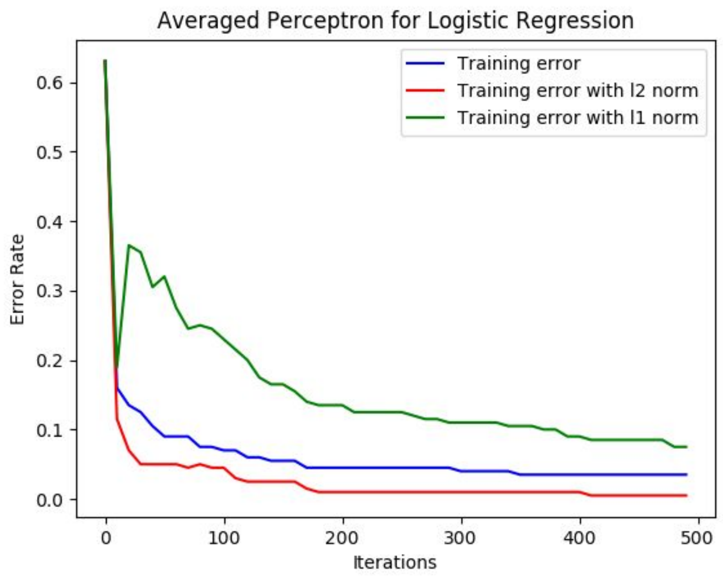

Figure 3

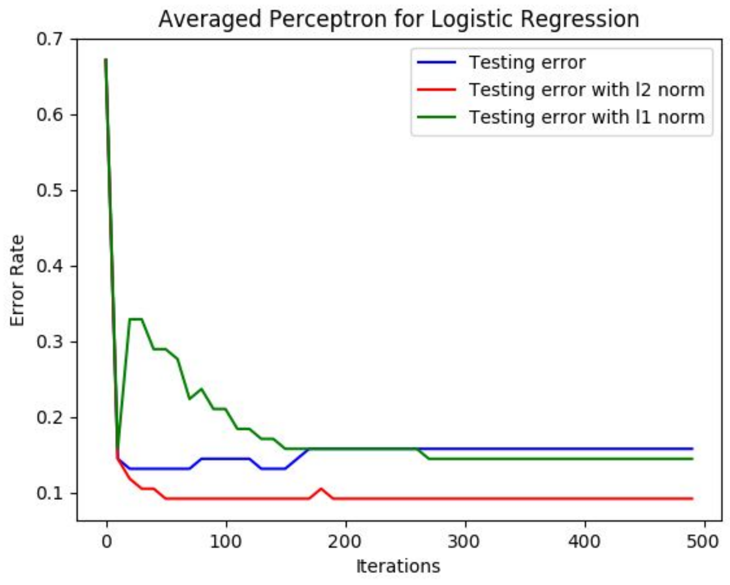

Figure 4

3.3 Difference between Vanilla perceptron and the Averaged perceptron.

We can see that the averaged perceptron is better than the vanilla perceptron. To make the vanilla perceptron to work better, we have to find a way to generalise it .The problem with the vanilla perceptron is that it takes into account the later points than the earlier points. To show this let us consider a perceptron model that runs for 5000 iterations. And let us consider that after 100 iterations, the model has learnt a good classifier, which classifies the dataset perfectly till 999th iteration and it makes an error in the 1000th iteration and the parameter gets updated. For all we know, the classifier at the 1000th iteration might not perform well generally but the final model we get is that classifier.

In order to rectify this, we average all the models by taking a weighted sum with weights that is directly proportional to how good the model performed in classification. This lets us to keep the good model into account too. Therefore the averaged model performs better.

4. Implementation of Gradient Ascent algorithm for Logistic Regression

4.1 Computation of the gradient and the plots of the error rates

Implementation of gradient ascent algorithm is done on the same data set but the labels have been changed from {-1,1} to {0,1} for our convenience. The gradient of the log likelihood function with respect to the model parameter **β** is as given below

$$\square (L(\boldsymbol{\beta})) = X^T(Y - \text{predictions})$$

Where X - data , Y- labels and predictions refers to sigmoidal output matrix of the $\boldsymbol{\beta}^T X$. Since our objective is to maximise the log likelihood function,we go for gradient ascent algorithm i.e we add the gradient term every iteration. The update rule of the parameters for every iterations **T** is given by

$$\boldsymbol{\beta}_{T+1} = \boldsymbol{\beta}_T + \boldsymbol{\eta}\square(L(\boldsymbol{\beta}))$$

Then we plot the training error rates and testing error rates for all the three cases of the dataset namely the raw, L1 normalised and L2 normalised.

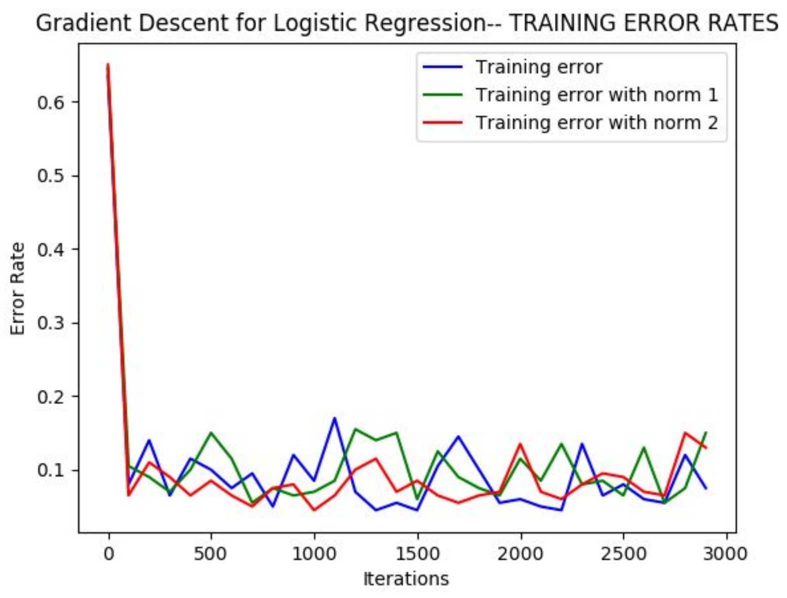

Figure 5

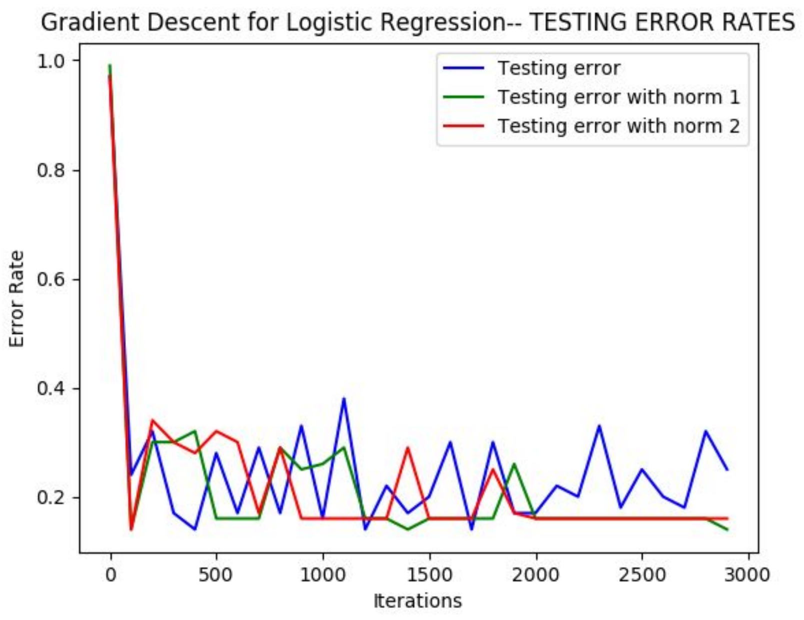

Figure 6

4.2 Perceptron algorithm Vs Gradient Ascent

As we can see from the plots,error rates in the perceptron algorithm converges to 0 while the rates in the gradient ascent doesn’t.

Since the data is linearly separable, perceptron algorithm performed amazingly well and also, unlike gradient descent , perceptron algorithm does not get stuck on the local maxima. It either converges at the global maxima or doesn’t converge at all.

Therefore in this data set, the Gradient descent might have got stuck in the local maxima and the error rates converged to a constant value around 0.1. But the perceptron algorithm found the global maxima and converged to 0.

5. Implementation of Newton’s method for locally weighted logistic regression

The locally weighted logistic regression can be seen as a modified version of KNN algorithm. The weights for each data point depends on whether it is close the point of query or not. Therefore if we wish to calculate the error rate of training and testing samples , we have to first compute the weights of the data points with respect to each of the data of the datapoint and use this specific weights to build models for each of the data. Therefore for the computation of training error rate , we will have 200 models each with different model parameters. Similarly for the computation of testing error rate we will have to model 76 different model parameters.

The computation of the weights is using the formula

$$w^i = \exp (- (|x - x^i|_{L2})^2 / 2\boldsymbol{\tau}^2)$$

W^i is the weight of the ith data point x^i with respect to the query point x. Tau = { 0.01, 0.05, 0.1, 0.5, 1.0, 5.0 } and we find the error rates with respect to different values of Tau.

5.1 Computation of gradient and hessian matrix

The gradient of the log likelihood function is given by

$$\nabla_{\beta} L(\boldsymbol{\beta}) = \sum_{i=1}^{N} (w^i \{ y^i (1- f_{\beta}(x^i)) x^{(i)\,T} - (1 - y^i) f_{\beta}(x^i) \})- 2\lambda\boldsymbol{\beta}$$

$$= \sum_{i=1}^{N} w^i \; x^{(i)\,T} \{ y^i - f_{\beta}(x^i) \} - 2\lambda\boldsymbol{\beta}^T$$

The Hessian matrix is given by differentiating the gradient with respect to **β** again.

$$\nabla_{\beta} \; (\nabla_{\beta} L(\boldsymbol{\beta})) = \sum_{i=1}^{N} (w^i \; x^{(i)} x^{(i)\,T} \{ 1 - f_{\beta}(x^i) \}) - 2\lambda$$

Where the lambda term is multiplied with identity matrix of (34x34) dimension. Therefore the hessian is a matrix of dimension 34x 34.

5.2 Implementation of Newton's method

The weights of each data point in the training and testing data set have been calculated with the given formula and have been stored in matrix.

The newton's method is also an iterative method whose update rule is given by

$$\boldsymbol{\beta}_{T+1} = \boldsymbol{\beta}_T + \boldsymbol{\eta} * (\nabla_{\beta} \; (\nabla_{\beta} L(\boldsymbol{\beta})))^{-1} * \; (\nabla_{\beta} L(\boldsymbol{\beta}))$$

The figure 7 shows the variation of training error rates with respect to Tau values. As we can see there is not much variation in the training error rates.

The codes for the prediction of the dataset, by learning the model are weights.py and have been uploaded in the github repository.

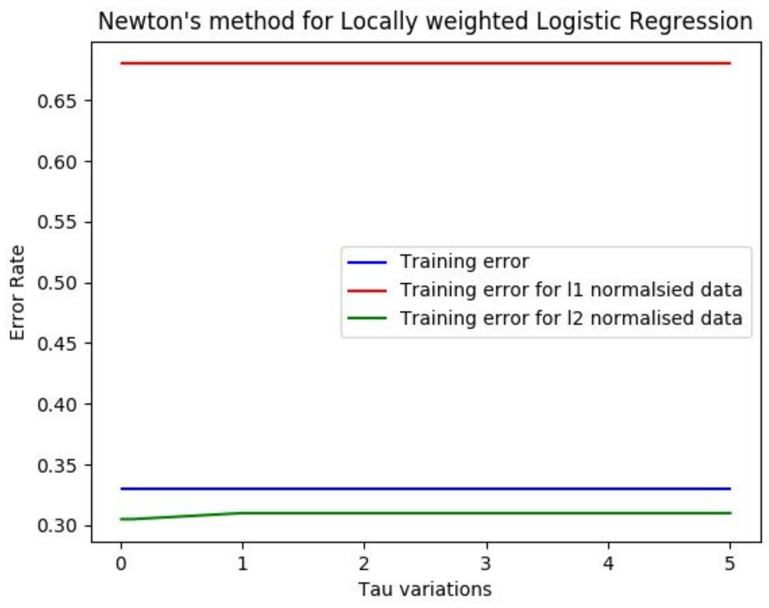

Figure 7

6. Conclusion

Thus we have discussed and successfully implemented three different algorithms namely perceptron, gradient ascent and newton's method for learning the parameters of the logistic regression model and also have plotted the different error rates plot for each algorithm.