# Machine Learning
# Programming Assignment-1

Implementation of Multilayer Perceptron Neural Network and evaluation of its performance in classifying Handwritten Digits

(2-4-2016)

Sivakiran Ayyagari
Rupesh Soni
Shubham Sharma
(Group-3)

Training Vs Testing error

Magnitude of the weights is directly proportional to the complexity. Higher the weight will give higher complexity. We can control the magnitudes of the weights by adding a penalty term.

The objective function with the regularization term is given by

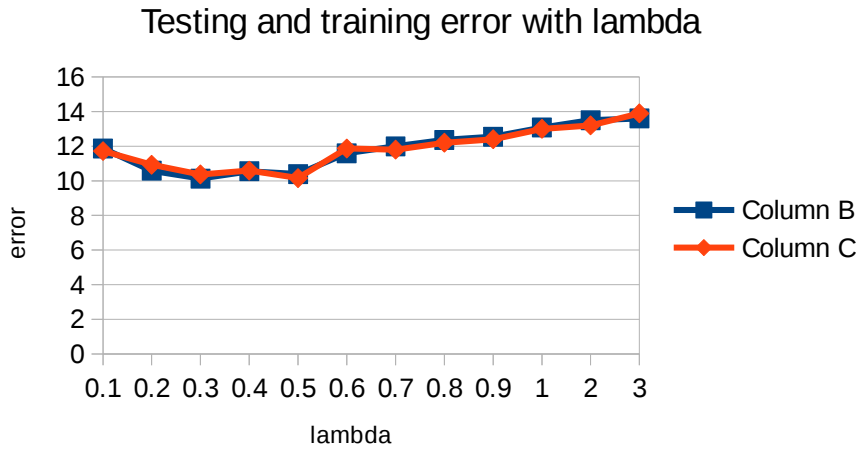
$$\widetilde{J}(W^{(1)}, W^{(2)}) = J(W^{(1)}, W^{(2)}) + \frac{\lambda}{2n}\left(\sum_{j=1}^{m}\sum_{i=1}^{d+1}(w_{ji}^{(1)})^2 + \sum_{l=1}^{k}\sum_{j=1}^{m+1}(w_{lj}^{(2)})^2\right)$$

Here the second term is the regularization term with lambda being the regularization coefficient. Changing the values of lambda will give respective accuracy. To get optimal value of lambda we ran the code for different lambdas by keeping the value of hidden nodes to be constant.

Number of hidden layers were kept to constant value of 8 and changed the values lambda. Below is the table consisting various accuracies and

| Lambda | Training accuracy | Testing Accuracy | Validation Accuracy |
|---|---|---|---|
| 0.1 | 88.132 | 88.28 | 88.77 |
| 0.2 | 89.416 | 89.07 | 88.64 |
| 0.3 | 89.87 | 89.63` | 89.19 |
| 0.4 | 89.448 | 89.41 | 89.03 |
| 0.5 | 89.61 | 89.84 | 89.04 |
| 0.6 | 88.406 | 88.13 | 87.35 |
| 0.7 | 88.01 | 88.2 | 87.1 |
| 0.8 | 87.64 | 87.8 | 86.8 |
| 0.9 | 87.45 | 87.6 | 86.9 |
| 1 | 86.92 | 87 | 86.5 |
| 2 | 86.5 | 86.8 | 86.8 |
| 3 | 87.1 | 86.1 | 85.8 |

Graph of plotted taking the above values for training and testing data



Here column B is the training data and column C is the testing data. As we see that with change in lambda error comes to minimum at lambda=0.5, and then after the error increases to 13.6 percent in both training and testing data.

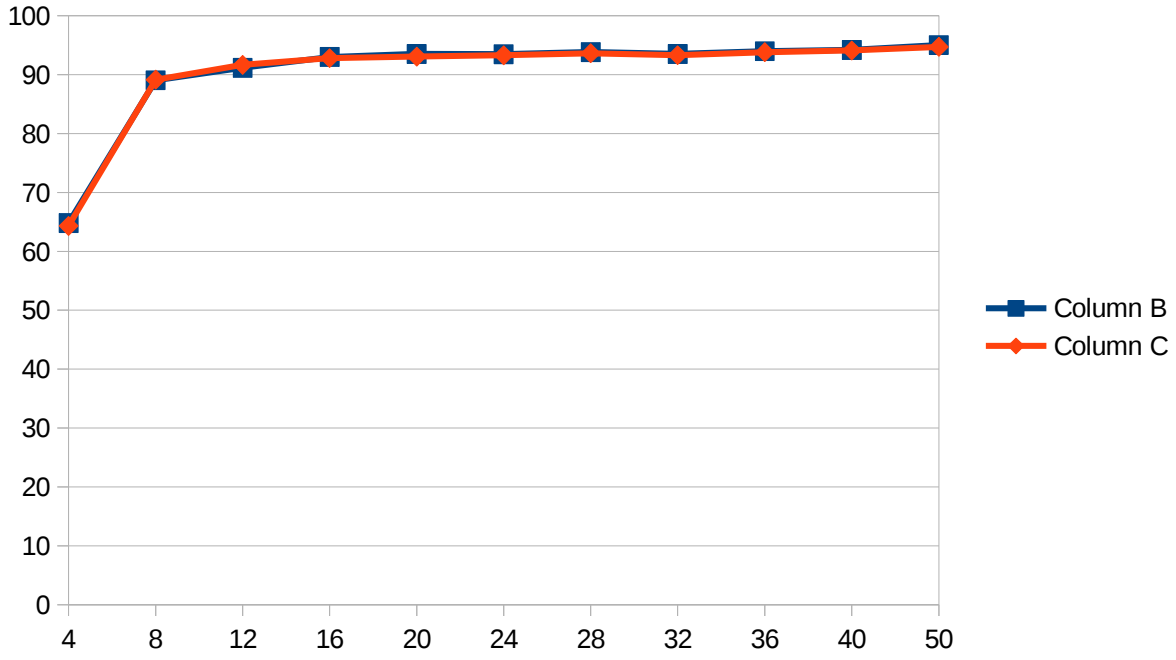
| Lambda | Training Data | Testing Data |
|---|---|---|
| 0.1 | 11.868 | 11.72 |
| 0.2 | 10.584 | 10.93 |
| 0.3 | 10.13 | 10.37 |
| 0.4 | 10.552 | 10.59 |
| 0.5 | 10.39 | 10.16 |
| 0.6 | 11.594 | 11.87 |
| 0.7 | 11.99 | 11.8 |
| 0.8 | 12.36 | 12.2 |
| 0.9 | 12.55 | 12.4 |
| 1 | 13.08 | 13 |
| 2 | 13.5 | 13.2 |
| 3 | 13.6 | 13.9 |

The above table gives the error of training and testing data.

After finding the optimal lambda we ran the code for getting optimal number of hidden nodes. Now we ran the code for various values of the hidden nodes. We kept lambda =0.5 , which is the optimal value of lambda.

| Hidden Nodes | Training Data | Testing Data |
|---|---|---|
| 4 | 64.8 | 64.3 |
| 8 | 89.042 | 89.14 |
| 12 | 91.152 | 91.65 |
| 16 | 93.004 | 92.8 |
| 20 | 93.5 | 93.06 |
| 24 | 93.46 | 93.28 |
| 28 | 93.83 | 93.62 |
| 32 | 93.52 | 93.3 |
| 36 | 93.98 | 93.81 |
| 40 | 94.196 | 94.1 |
| 50 | 95.02 | 94.71 |

The above table shows the increase in accuracy with the increase in the number of hidden nodes. If we see with 4 nodes we have accuracy of 64.3% and as we keep on increasing the hidden nodes at 28 nodes we see that the accuracy is 93.62%. Though with the increase in nodes we get 94.7% for 50 nodes but then it becomes computationally very expensive. So we take an optimal value for number of hidden layers which is 28 nodes.




Above it the graph of the accuracies of training and testing data with change in number of hidden nodes. We can see that accuracy increase with increase in the number of hidden nodes but at the same time it becomes computationally more expensive for greater number of hidden nodes.
So we found optimal value of lambda and hidden nodes.