CSE 574 Introduction to Machine Learning

Programming Assignment 1

Handwritten Digits Classification Using Neural Networks

Submitted by:

Debika Dutt: 50170009

Mahesh Venkataramaiah: 50170332

Vardhana Srinivas Kulkarni: 50169374

Group# 7

1. INTRODUCTION

A Multilayer Perceptron (MLP) is a feedforward artificial neural network model that maps sets of input data onto a set of appropriate outputs. An MLP consists of multiple layers of nodes in a directed graph, with each layer fully connected to the next one. Except for the input nodes, each node is a neuron (or processing element) with a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training the network. MLP is a modification of the standard linear perceptron and can distinguish data that are not linearly separable.

A neural network is put together by hooking together many of our simple "neurons," so that the output of a neuron can be the input of another.

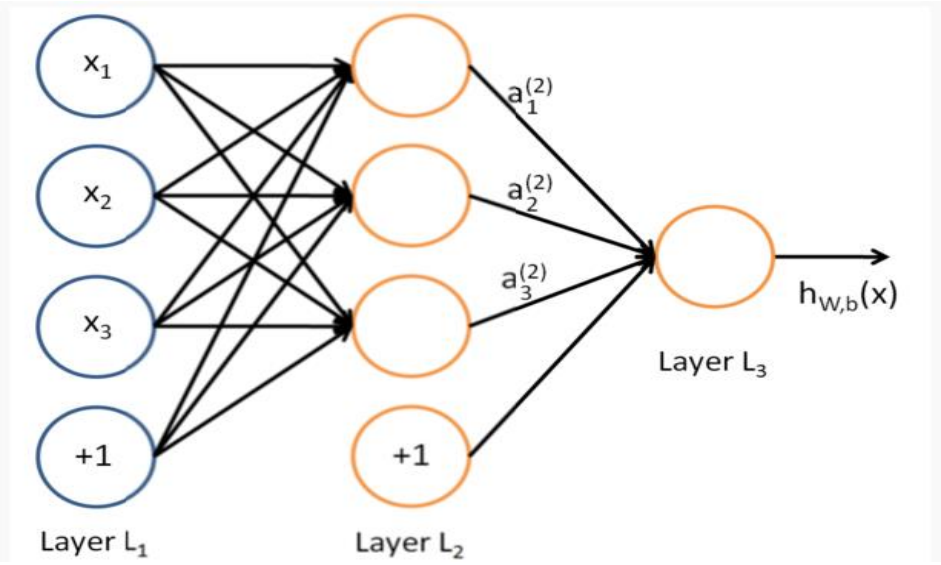

Fig 1. Example Neural Network

In this figure, the circles denote the inputs to the network. The circles labeled "+1" represents the bias unit. The leftmost layer of the network is the input layer, and the rightmost layer the output layer. The middle layer of nodes is the hidden layer which can be considered as the learned features extracted from the original data set. We also say that our example neural network has 3 input units (not counting the bias unit), 3 hidden units, and 1 output unit.

2. IMPLEMENTATION

In this assignment we implemented a Multilayer Perceptron Neural Network and evaluated its performance in classifying handwritten digits. We were able to setup a Machine Learning environment by using Python 2.7.1 and installing the Numpy, Scipy and Math packages in Python. We imported the original dataset "mnist_all.mat" file from MNIST which consists of 10 matrices for testing set and 10 for training set, corresponding to 10 digits. We have split the training data into training and validation data. The input layers consist of 784 nodes corresponding to the 28 x 28 pixel image of the hand written digits. In addition to it, we added a bias unit in the input layer, therefore the number of nodes in the input layer of the neural network now becomes 785. The weights for each of these nodes is randomly initialized to some small value and the weight of bias node is set to one. The number of hidden nodes is 50 and an additional bias node is added to the hidden layer making the total number of hidden nodes as 51. The weights from hidden layer to the output layer too is initialized to some small random value. The output layer consists of 10 nodes corresponding to the ten digits. The output of neural network is calculated by multiplying the input by the weight and result is passed through the sigmoid activation function. This completes the forward pass. Now the outputs of the 10 output nodes are compared to the actual labels of data and the error is calculated. The derivative of error function with respect to weight of each connection in neural network is calculated for all the samples. The gradient of error function is calculated for the samples. Now the average gradient of error function and total error for all the samples is further passed to minimize function. The minimize function tries to reduce the error and accordingly updates the weights.

a. Bias Node:

As discussed earlier, the output of neural network is calculated by multiplying the input by the weight and result is passed through the sigmoid activation function. By changing the weights changes the steepness of the sigmoid curve. Just changing the steepness of the sigmoid is not very helpful. We should be able to shift it to right or left and this is where bias node is useful. The bias value allows the shifting of activation function to right or left. The bias node is only used in the feed forward propagation and not during the back propagation. We do not calculate the error for the bias node and updating of weights corresponding to bias node is not done as we want the bias weights to be constant.

b. Regularization coefficient λ:

A common problem encountered in neural network is the overfitting problem. That is, the learning model is best fit with the training data but gives poor generalization when it is tested with the validation data. In order to overcome this, we make use of a regularization coefficient λ.

3. SELECTION OF HYPERPARAMETERS

The graph shown below indicates that with each iteration, the gradient value decreases. Also, with the constant number of hidden nodes = 50, we observe that decrease in the value of lambda, results in a very small change in the variation of gradient values. Lower values of lambda give better results. In fig 3, we observe that for constant value of hidden nodes, accuracy increases with decrease in lambda value.

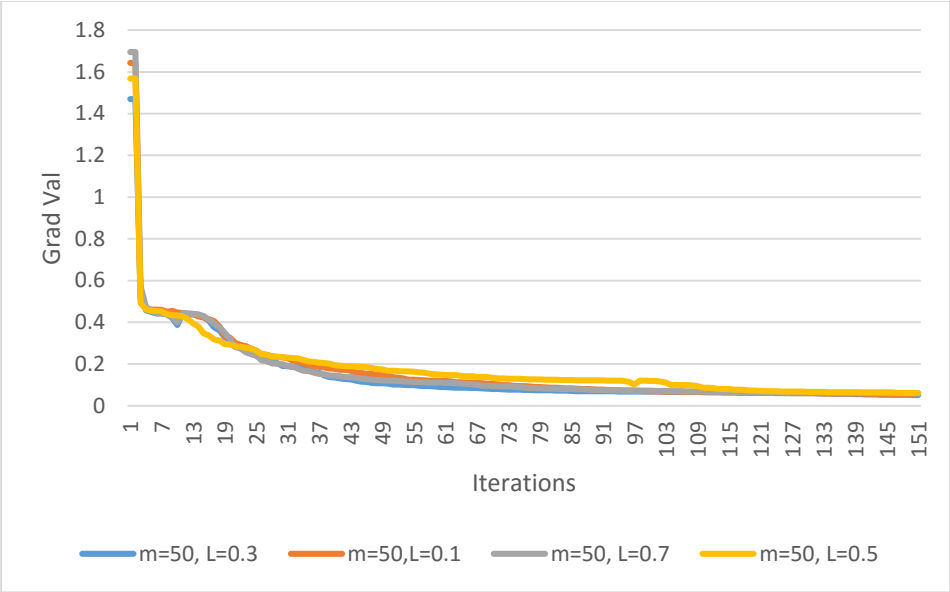

Fig 2. Graph of Gradient value vs Number of Iterations

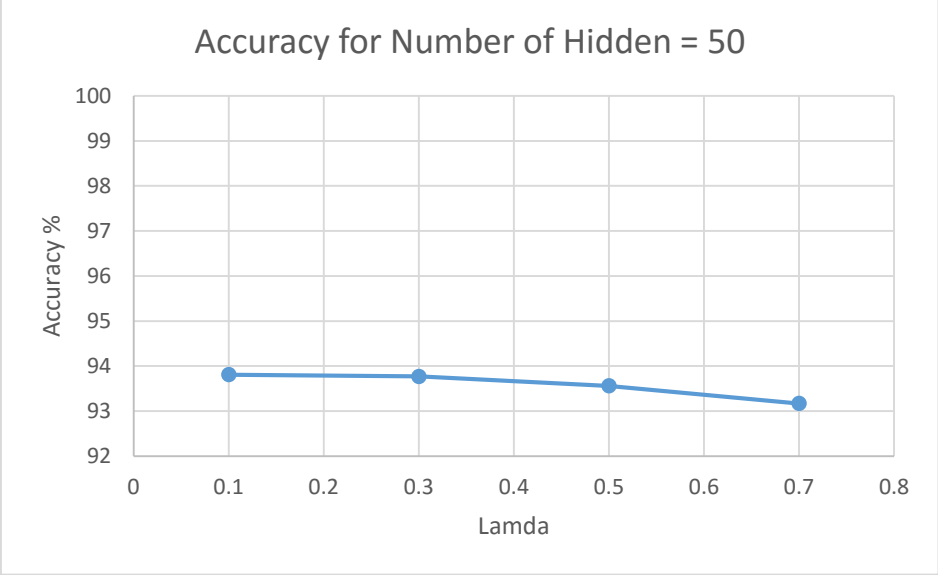

Fig 3. Graph of Accuracy vs lambda

The graph shown below indicates that with each iteration, the gradient value decreases. Also, with the constant value of lambda = 0.7, we observe that decrease in the number of hidden layers, it takes more number of iterations to reduce the error significantly. Lower values of lambda give better results. In fig 3, we observe that for constant value of hidden nodes, accuracy increases with decrease in lambda value. Also, as the number of Hidden nodes are increased, the accuracy of the neural network increases

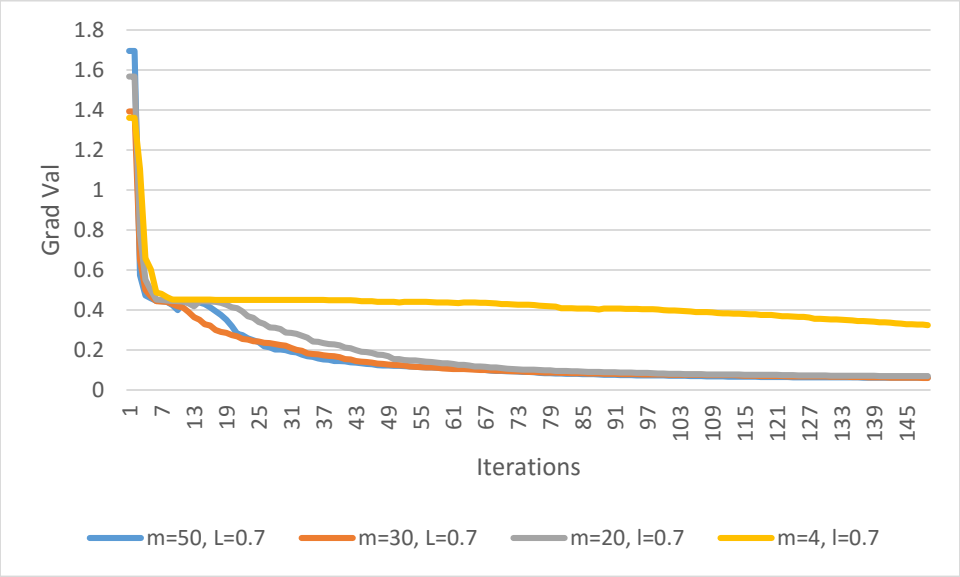

Fig 4. Graph of Gradient value vs number of iterations

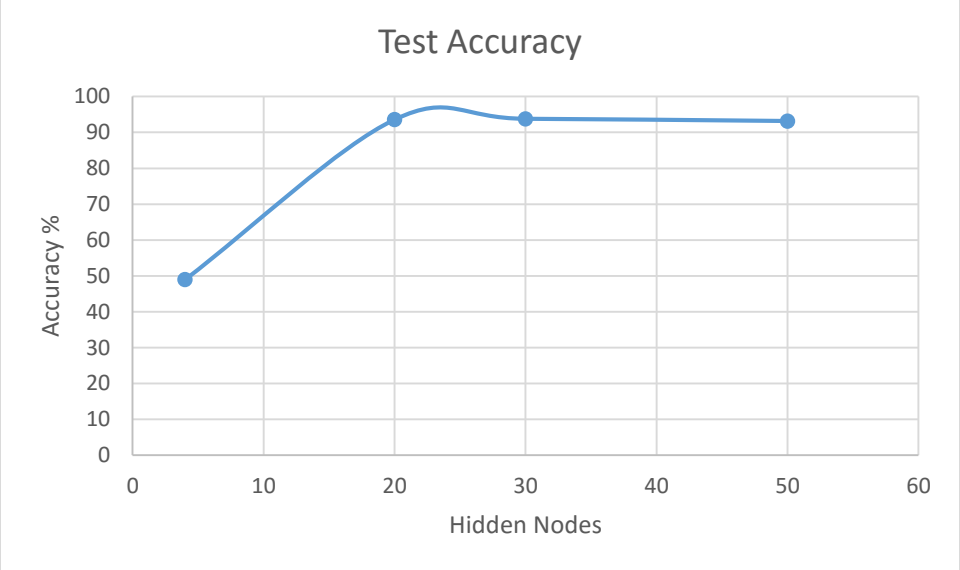

Fig 5. Graph of Accuracy vs number Hidden nodes

4. RESULTS

Accuracy\(m, L)	50, 0.5	50, 0.3	50, 0.1	50, 0.7	30, 0.7	20, 0.7	4,0.7
Train Accuracy	93.26	94.586	93.96	93.068	93.864	93.52	47.854
Validation Accuracy	94.28	95.12	93.74	93.98	94.58	94.42	54.77
Test Accuracy	93.56	94.48	93.81	93.17	93.79	93.57	48.98

Table 1. Accuracy vs. number of Hidden nodes & Lambda value

From the above table, we observe that the optimum results are obtained when hidden nodes are 50 and lambda value is 0.3.

5. CONCLUSION:

The Multilayer Perceptron Neural Network was successfully implemented to classify the handwritten digits. Upon testing the network to classify the hand written digits in the training set, validation set and the test set, the network gave a high accuracy. The regularization of the network is implemented using the regularization coefficient which avoids the over fitting problem.

6. REFERENCES

Book - McGrawHill_-_Machine Learning_-Tom Mitchell

Website - https://en.wikipedia.org/wiki/Multilayer_perceptron

http://ufldl.stanford.edu/tutorial/supervised/MultiLayerNeuralNetworks/