800936333
lgopalab@uncc.edu
Laxman Sai Karthik Mahidhar Gopalabhatla

Character Recognition Using Neural Networks

Introduction:

The current project is based on the digits dataset available in UCI Machine Learning Repository. The primary task of this assignment is to use the dataset to predict the digits using Artificial Neural Networks. This data set consists of preprocessed normalized bitmaps of handwritten digits from a preprinted form. 32x32 bitmaps are divided into non-overlapping blocks of 4x4 and the number of on pixels are counted in each block. This generates an input matrix of 8x8 where each element is an integer in the range 0..16. The task here is to design a Multi-Layer perceptron Neural Network using Back-Propagation to train and test the data.

Design:

The design of this program is inspired from an article "**Using neural nets to recognize handwritten digits**" by Michael Nielsen.

The entire computation of the neural networks in the code happens over the matrices. The input data is converted into matrices and the computations are performed on the matrices considering them as vectors. numPy package was extensively used for these computations. The neural network design was fairly simple and has been designed to have one input layer with 64 inputs, one hidden layer and one output layer with 10 outputs. The size of the Hidden layer has been considered one of the components for experimentation and would be determined in later stages after experimenting with different values. The primary factors that I have considered for experimentation are the following.

1. Epochs
2. Hidden layer Size
3. Learning factor

These factors are selected as they have huge impact on the way the classifier works and the right balance between these yields us the best results and it is our job to do that. Hence experiments were run by changing these parameters and the best combination of these parameters is selected.

Implementation:

The code is completely written in python. The implementation can be deeply explained as follows.
The entire Neural network is encapsulated in a class called NeuralNet. In the code, number of epochs to be run is given and for every epoch the **feed** function is executed which computes the sigmoid of the hidden layer and the output layer which are used to compute prediction and later this is used along with expected vector to back propagate. The errors of Hidden node and the output node are calculated using dot product and transpose functions which are in turn used to calculate the Delta functions for hidden and output layer. These delta values are used for modifying the existing weights. This entire process is done for specified number of epochs.

Now that the training is complete running the tests is this way. Once the output_vector is computed, this is used to predict the output. The index of the element with highest value is the digit to be predicted.

Here the number of correct predictions and overall sample length is taken to know with what accuracy the predictions were made. The same can be seen in the results as well.

The Cost (Loss or Objective) function is as follows

$$C(w,b) \equiv \frac{1}{2n}\sum_{x} \| \boldsymbol{y}(\boldsymbol{x}) - \boldsymbol{a} \|^2$$

w denotes the collection of all weights in the network,
b all the biases,
n is the total number of training inputs,
a is the vector of outputs from the network when x is input, and the sum is over all training inputs, x.

Experiments:

To optimize the networks generalization performance, the code was run with many combinations of epochs, learning factor and hidden layer sizes and the optimal values are selected based on that. The following were few observations made. Best Accuracy was yielded when Learning rate was 0.2, Hidden Layer Size was 40 and Epoch was 150. The network properly predicted 1430 digits out of 1500, which give an accuracy of 95.33%.

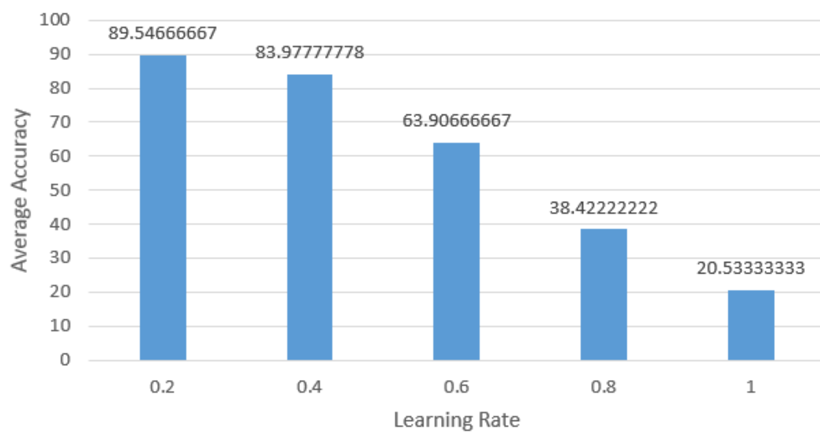
Learning rate	Hidden layer size	Epoch	Correct	Total	Accuracy
0.2	10	50	1059	1500	70.6
0.2	10	100	1314	1500	87.6
0.2	10	150	1249	1500	83.26667
0.2	20	50	1392	1500	92.8
0.2	20	100	1304	1500	86.93333
0.2	20	150	1372	1500	91.46667
0.2	30	50	1349	1500	89.93333
0.2	30	100	1358	1500	90.53333
0.2	30	150	1368	1500	91.2
0.2	40	50	1395	1500	93
0.2	40	100	1388	1500	92.53333
0.2	**40**	**150**	**1430**	**1500**	**95.33333**
0.2	50	50	1388	1500	92.53333
0.2	50	100	1388	1500	92.53333
0.2	50	150	1394	1500	92.93333
0.4	10	50	1277	1500	85.13333
0.4	10	100	1326	1500	88.4
0.4	10	150	1322	1500	88.13333
0.4	20	50	1043	1500	69.53333
0.4	20	100	1229	1500	81.93333
0.4	20	150	1333	1500	88.86667
0.4	30	50	1245	1500	83
0.4	30	100	1280	1500	85.33333
0.4	30	150	1207	1500	80.46667
0.4	40	50	1319	1500	87.93333
0.4	40	100	1316	1500	87.73333
0.4	40	150	1313	1500	87.53333
0.4	50	50	1126	1500	75.06667
0.6	50	50	973	1500	64.86667
0.6	40	100	977	1500	65.13333
0.6	40	150	1195	1500	79.66667
0.8	10	50	276	1500	18.4
0.8	10	100	299	1500	19.93333
0.8	10	150	579	1500	38.6
0.8	20	50	977	1500	65.13333
0.8	20	100	670	1500	44.66667
0.8	20	150	284	1500	18.93333
0.8	30	50	537	1500	35.8
0.8	30	100	950	1500	63.33333
0.8	30	150	898	1500	59.86667
0.8	40	50	153	1500	10.2
0.8	40	100	815	1500	54.33333
0.8	40	150	824	1500	54.93333
0.8	50	50	674	1500	44.93333
0.8	50	100	412	1500	27.46667
0.8	50	150	297	1500	19.8
1	10	50	582	1500	38.8
1	10	100	151	1500	10.06667
1	10	150	151	1500	10.06667
1	20	50	276	1500	18.4
1	20	100	515	1500	34.33333
1	20	150	300	1500	20
1	30	50	428	1500	28.53333
1	30	100	248	1500	16.53333
1	30	150	294	1500	19.6
1	40	50	762	1500	50.8
1	40	100	295	1500	19.66667

800936333
lgopalab@uncc.edu

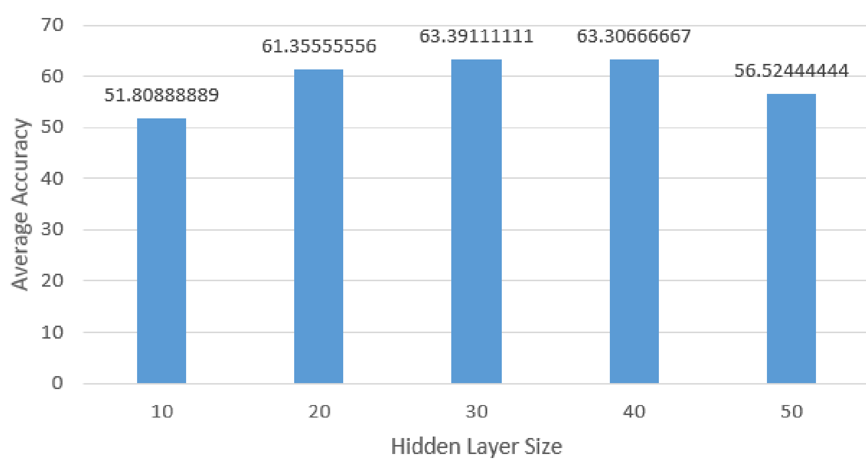
0.4	50	100	1222	1500	81.46667
0.4	50	150	1337	1500	89.13333
0.6	10	50	1033	1500	68.86667
0.6	10	100	283	1500	18.86667
0.6	10	150	756	1500	50.4
0.6	20	50	1157	1500	77.13333
0.6	20	100	822	1500	54.8
0.6	20	150	1131	1500	75.4
0.6	30	50	892	1500	59.46667
0.6	30	100	1023	1500	68.2
0.6	30	150	1186	1500	79.06667
0.6	40	50	909	1500	60.6

1	40	150	153	1500	10.2
1	50	50	156	1500	10.4
1	50	100	153	1500	10.2
1	50	150	156	1500	10.4

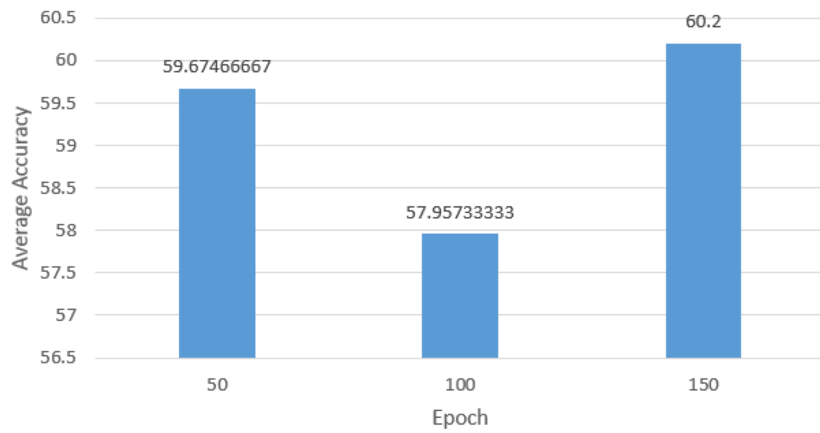
According to the above data collected, when a graph was plotted with Average accuracy vs Learning rate we have found that the train with lower learning rate yields better accuracy. This can be observed in the below graph.



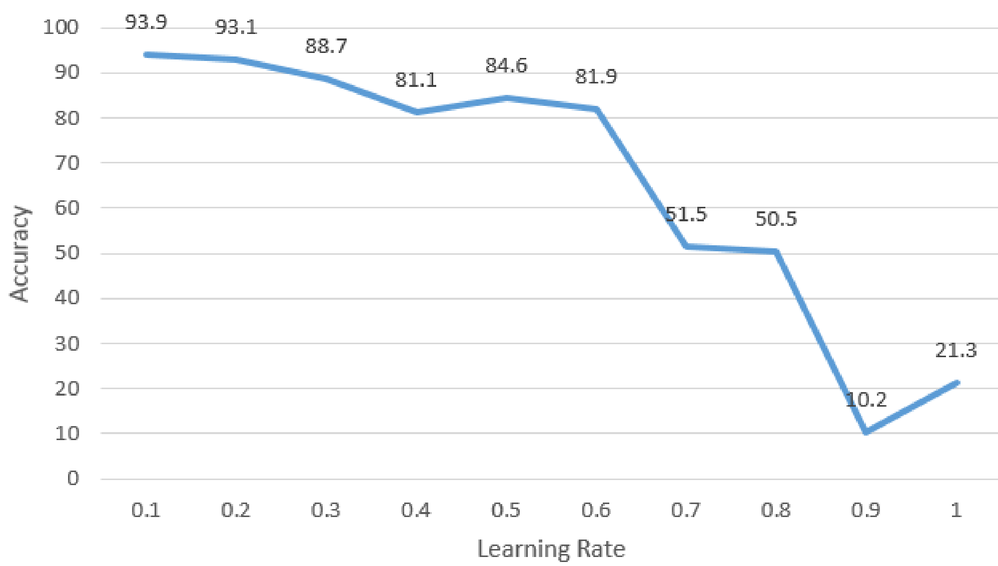
When a graph was plotted between Average Accuracy and Hidden layer size, it was found that the average accuracy was a bit more when the hidden layer size is 30. The same can be observed below.



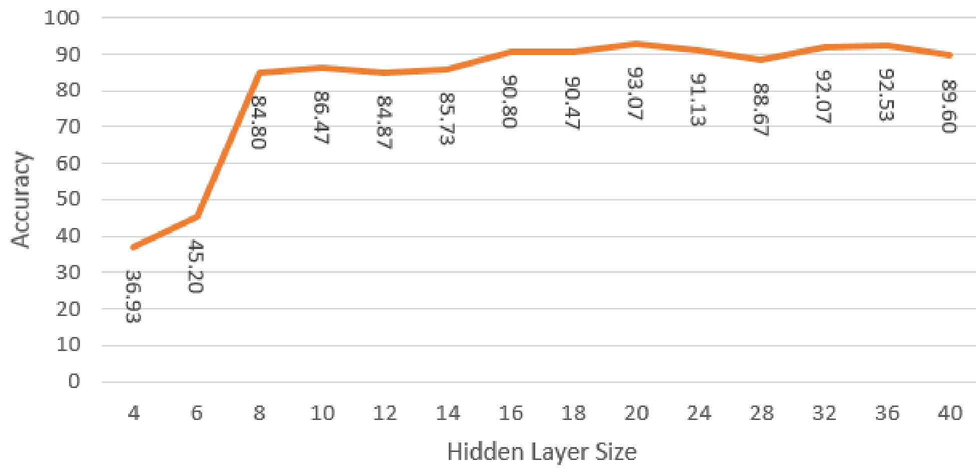
In the same way a graph between Epoch and Average Accuracy given us not much of a details as this may vary from different learning rate and hidden layer sizes.



To understand the relation between Learning rate and Accuracy, a test run was conducted with Epoch set to 50, Hidden Layer size set to 30 and the following results were yielded which clearly show that as the learning rate increases the accuracy decreases as the amount that has to be learning, grows.



To understand the relation between Hidden layer size and Accuracy, a test run was conducted with Epoch set to 50, Learning Rate set to 0.3 and the following results were yielded which clearly show that as the Accuracy gradually increases and after a certain point after the changes are not much significant.



We have also observed that the overall time for execution or say, for training, increases when the learning rate increases.

Therefore according to the observations made above the optimal solution for the current problem is a Learning Rate of 0.2, 40 Hidden layer size and a 150 Epochs. This has produced over 95% accuracy in predicting the correct digit.