Machine Learning – ITCS 6156

Homework – 3

Introduction:

The dataset given has 64 input features and 10 different classes. The dataset is constructed by first dividing the 32 x 32 image into 4 x 4 of 8 x 8 cells in each of 16 grids. And then they computed 1 x 4 vector from each 8 x 8 grid there by getting 16 times 4 (64) features. When observed each record in dataset, I had found a pattern in it. For a record if we map each non-zero element it gives the shape of the digit it belongs to.

Design:

I got the network architecture with 1 hidden layer with 20 nodes and having 64 nodes in input layer and 10 outputs in output layer. The factors I considered were number of hidden layers, number of nodes in each hidden layer, learning rates at each level (other than input level), and the difference between training error and testing error. For the activation function, I used the sigmoidal function.

Implementation:

I implemented the Neural Network using MATLAB platform. In the training phase, we first specify number of levels and nodes in each level, so that a model object is built based on this information storing the weights associated with each node in the previous level corresponding to each node in the current level. After weights' initialization, I considered different step size (learning rate) at each level other than input level. Now, I make a call to build the model, where I run the forward pass and backward pass on each training sample and update weights accordingly. The above forward and backward pass is done on each record of training sample. The number pf times this passes will be either controlled by the iteration limit or if the cost is increasing from the previous iteration or if error at that pass is very much less than the 0.001 in my case.

Regarding performance analysis, I am considering the change in step size, change in number of nodes, change in number of levels. For selecting the better model, I considered the difference between the training error and testing error, such that the configuration of the model that derives the minimum error difference is chosen. For checking the model not to get overfitted or underfitted, the criteria of minimum error difference between training and testing is used. While in testing phase, we simply run the forward pass using the updated model.

Experimentation:

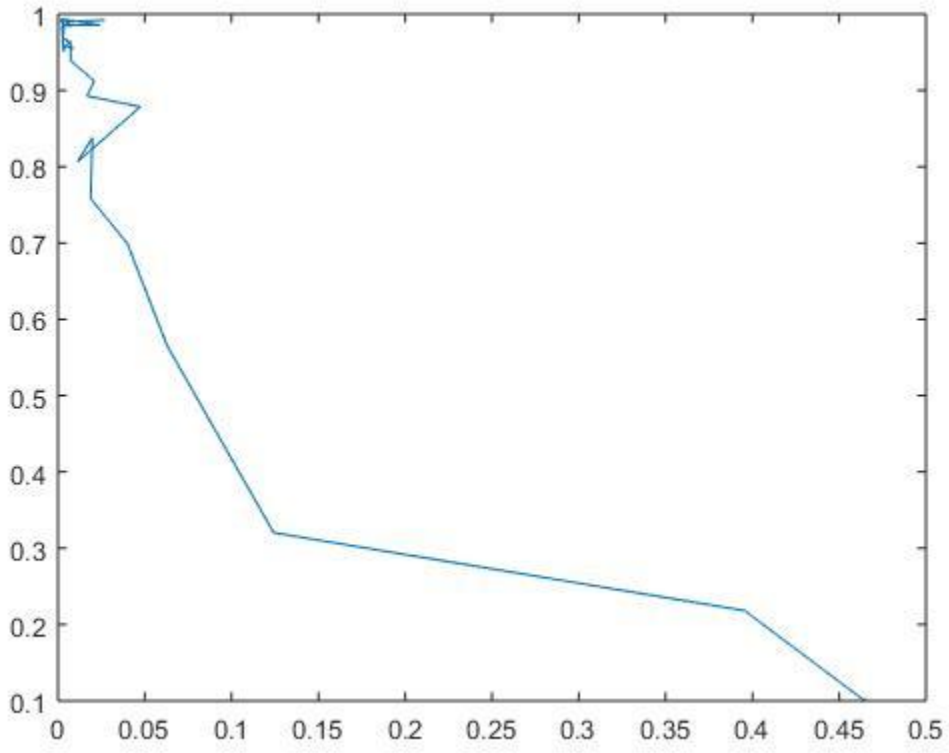
Network Config:

- The Iteration limit is fixed to 100.
- 25% of training data is used for training.
- 1 hidden layer with 20 nodes.
- Error threshold is 0.001.
- Learning rates: (0.1, 0.1).

Initially when rand() (0 – 1) function is used to produce the initial values for weights, the accuracy was too low ~9-10% on testing data. When traced, the problem is in the hidden layer, where all node activations are set to 1 and therefore for the next subsequent iterations the weights between input and hidden layer are not updated. To avoid this problem, I took the 0.001 * times the rand() (0 – 0.001) to produce initial values for weights. After this modification, the accuracy jumped to ~80% on testing data.

Network Config:

- The Iteration limit is fixed to 100 in all cases.
- 25% of training data is used for training.
- 1 hidden layer with 20 nodes.
- Error threshold is 0.001.
- Learning rates: (0.1, 0.1).



The above is the graph plotted for error value (x-axis) to the accuracy (y-axis). As shown in the graph when error is <0.1 the accuracy is oscillating this is due to the learning rate parameter. To avoid this and get clean curve the following changes have been made.

Network Config:

- The Iteration limit is fixed to 100 in all cases.
- 25% of training data is used for training.
- 1 hidden layer with 20 nodes.

- Error threshold is 0.001.
- Learning rates: 1. (0.1, 0.09) and 2. (0.1,0.11).

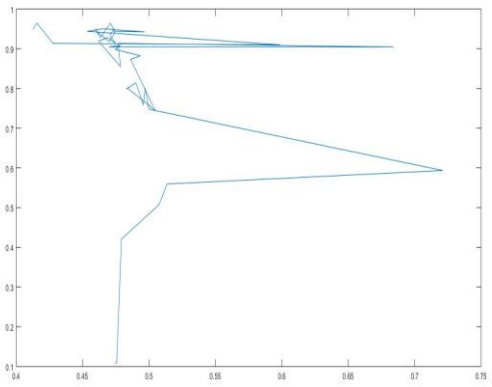


Fig. 1

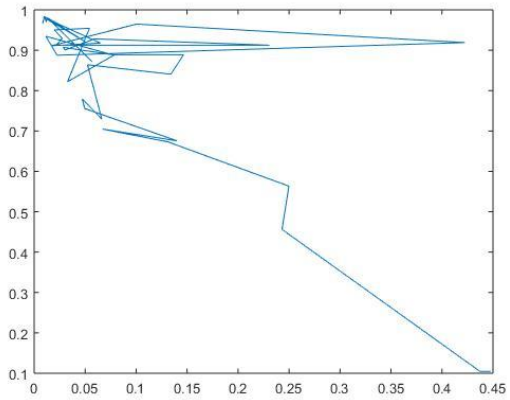


Fig. 2

As seen in the above graphs, either 0.09 or 0.11 for the second learning rate doesn't gives us the clean curve. That means the second learning rate's correct value is 0.1. And thus I started to change the first learning rate.

Network Config:

- The Iteration limit is fixed to 100 in all cases.
- 25% of training data is used for training.
- 1 hidden layer with 20 nodes.
- Error threshold is 0.001.
- Learning rates: 1. (0.01, 0.1), 2. (0.009,0.11), 3. (0.008,0.1), 4. (0.007,0.1) and 5. (0.005,0.1).

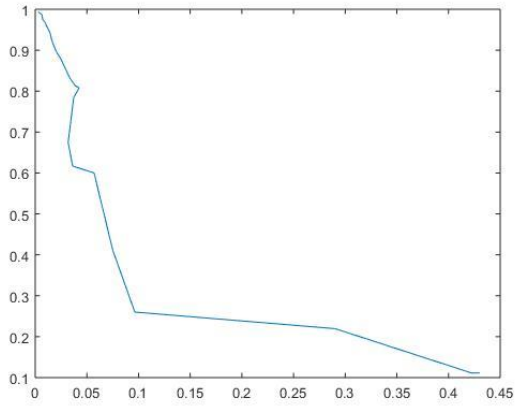


Fig. 1

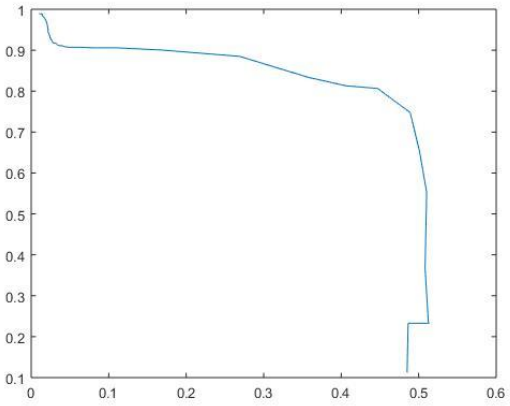


Fig. 2

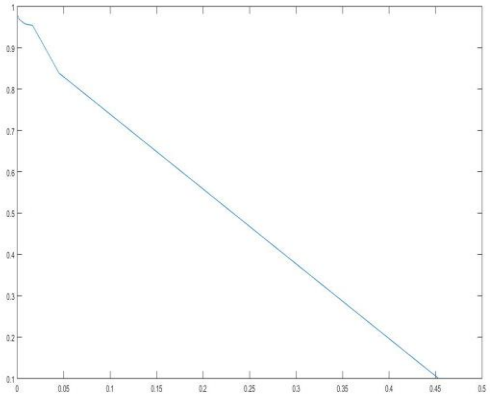


Fig. 3

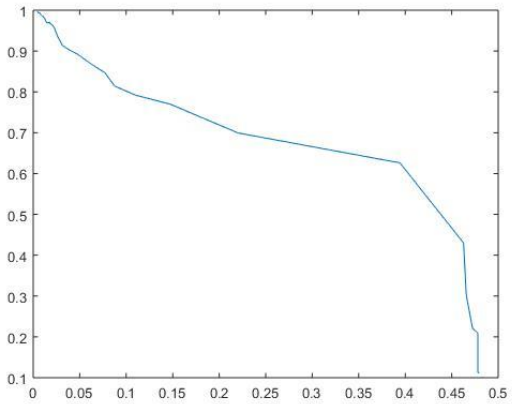


Fig. 4

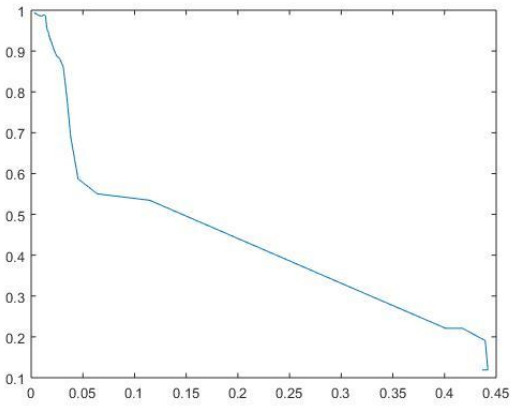


Fig. 5

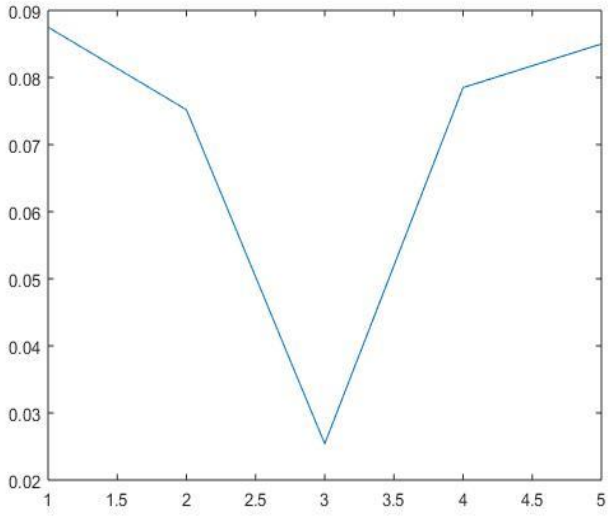
In all the above graphs, the fluctuations in the curve are too minimum and to choose a particular value for parameter setting such that it should not overfit the model. For that I computed the difference of the training error and testing error of all the 5 different settings. The error values are as follows:

training_errors = [0.0065, 0.0105, 0.0225, 0.0039, 0.0052]

testing_errors = [0.0940, 0.0857, 0.0479, 0.0824, 0.0902]

Using these values, we compute the difference in errors, which results to:

diff_errors = [0.0875, 0.0752, 0.0254, 0.0785, 0.0850] which gives the plot as follows:



It is clear from plot that the minimum error is at 3rd setting that is (0.008, 0.1) as learning rates. Let us try another model with the learning parameter as (0.008, 0.1) and varying levels and nodes in it.

Network Config:

- The Iteration limit is fixed to 100 in all cases.
- 25% of training data is used for training.
- a. 1 hidden layer with 30 nodes and b. 2 hidden layers with 5 nodes each.
- Error threshold is 0.001.
- Learning rates: (0.008, 0.1) and (0.008, 0.1, 0.008).

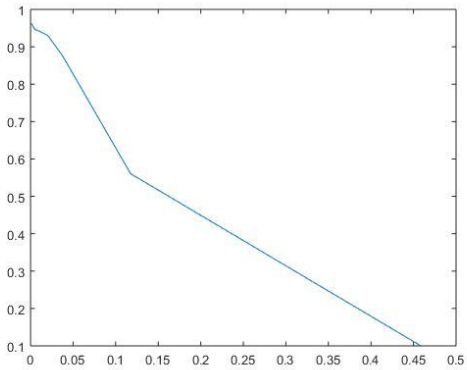


Fig. a

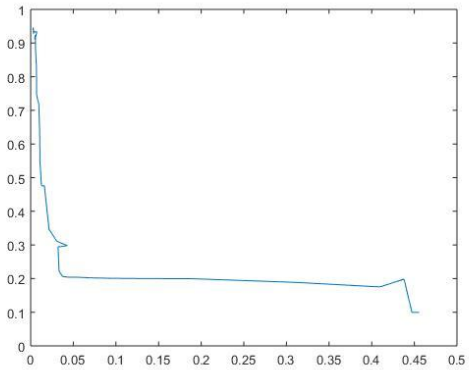


Fig. b

In the above scenarios, when I used model of 2 hidden layers the accuracy (training – 0.9411, testing – 0.9065) drops and even the difference between errors (0.0346) is also high. In scenario where we use 30 nodes in a single hidden layer then the accuracy is bit low and error (0.0415) is also high. That's why I choose the model with 1 hidden layer with 10 nodes in them and having learning parameters as (0.008, 0.1).

Conclusion:

In my analysis, I feel the parameters of the learning rate in combination to the random weights used for initialization has major impact on the model performance. The next immediate factor is the number of layers in combination with the number of nodes in those layers. The best performance I achieved was 95.21% on testing and 97.75% on training datasets.