NAME: RAKESH BAINGOLKAR
UB: 50097576
UBIT: rbaingol

MACHINE LEARNING - PROJECT 2
CLASSIFICATION - LOGISTIC REGRESSION AND NEURAL NETWORK

OBJECTIVE:

The objective of this project was to classify the handwritten digits into 10 digits from 0-9. For this the feature vectors were given which were divided into two sections. First section was the training feature vectors which included 2000 vectors for each digit so it came to around 20000 vectors for all the ten digits. Second section consisted of testing vectors which had 1500 vectors. Each vector had 512 features.

APPROCHES:

1. Multi-class logistic classification using stochastic gradient descent to find the weight matrix.
2. Neural-network approach in which we use sigmoidal functions as the hidden layer activation functions and softmax functions as the output layer activation function. Thus it is two layered. For finding the cross entropy I used Back Propagation.
3. To compare these above methods I used the standard package of neural network toolbox.

So, I trained the vectors using two methods and we will compare the performance later in this document.

LOGISTIC REGRESSION:

Multiclass Logistic Regression is used to train the network using probability i.e. it is a probabilistic classifier. In this logistic regression based on the graphs shown below the related parameters are chosen to minimize the error rate. In multiclass logistic regression there are basically two stopping criterion one being providing with a threshold on when to stop the training and the other being to compare the iteration count and the error rate which helps us to choose the iteration count on which the error value was minimum i.e. global minima.

I used the second method for finding the minimum error rate.

FORMULAE:

The likelihood in a multiclass classification is given by:

$$p(\mathbf{T}|\mathbf{w}_1,\ldots,\mathbf{w}_K) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(\mathcal{C}_k|\boldsymbol{\phi}_n)^{t_{nk}} = \prod_{n=1}^{N}\prod_{k=1}^{K} y_{nk}^{t_{nk}}$$

The error rate to be found is given by:

$$E(\mathbf{w}_1,\ldots,\mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1,\ldots,\mathbf{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_{nk}$$

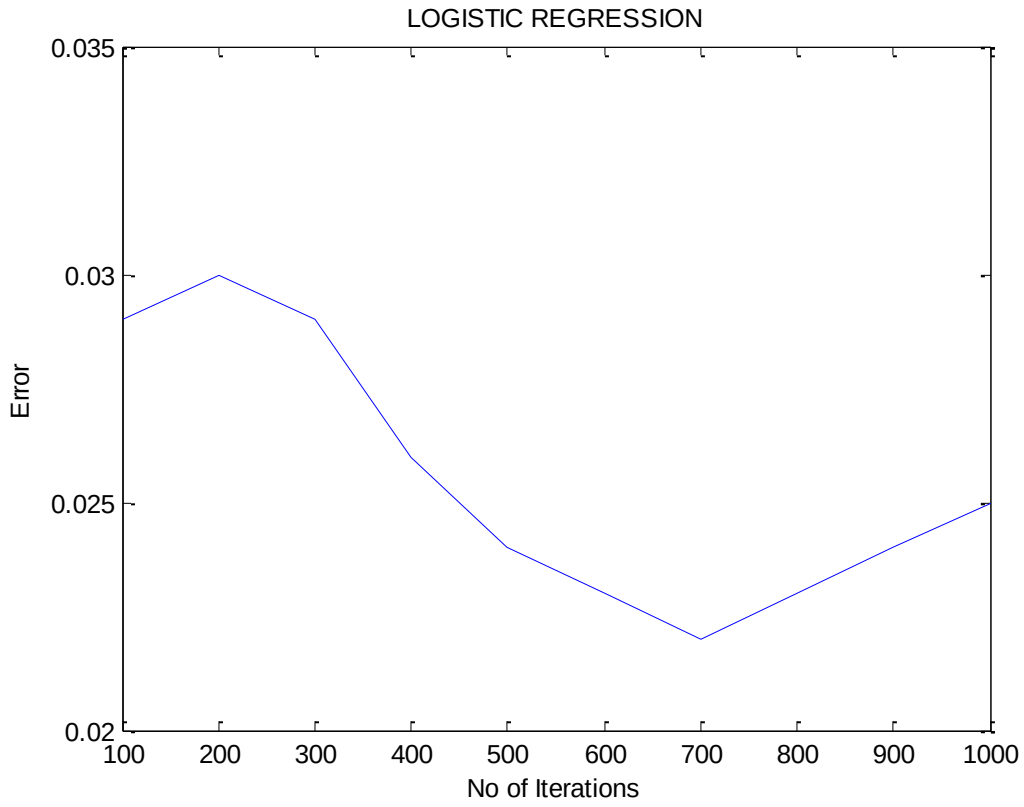
And the gradient of error is given by:

$$\nabla_{\mathbf{w}_j} E(\mathbf{w}_1,\ldots,\mathbf{w}_K) = \sum_{n=1}^{N} (y_{nj} - t_{nj})\,\boldsymbol{\phi}_n$$

Feature space being a linear function Phi is same as X in the above equation.

GRAPHS:

Error was noted down for various iterations



Here from the graph we can see that for the iterations 700 we get the minimum error which is 0.022. The error initially increases a bit and then it decreases till 700 iterations and then it again increases.

ERROR PERCENTAGES

Error: 0.022

Learning Rate: 0.003

Iterations: 700

Reciprocal Rank Percentage: 98.38%

NEURAL NETWORK:

In this method the data is trained using the gradient descent method. Here we take random samples from the training data and we obtain the derivatives using back propagation. We use sigmoidal functions as the hidden layer activation functions and softmax functions as the output layer activation function.

FORMULAE:

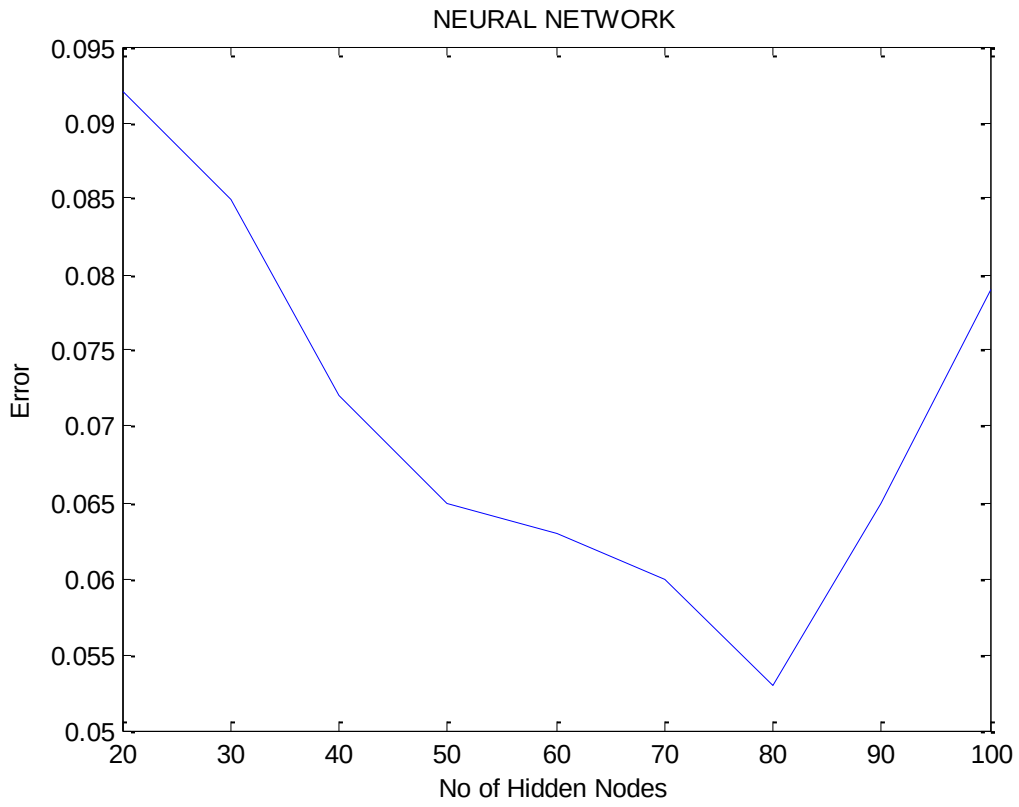
The Back Propagation formula is given by

$$\Delta w_{hl}^{(2)} = \eta \sum_{p} \left(targ_l - out_l^{(2)} \right) out_h^{(1)}$$

$$\Delta w_{hl}^{(1)} = \eta \sum_{p} \sum_{k} \left(targ_k - out_k^{(2)} \right) w_{lk}^{(2)} . out_l^{(1)} . \left(1 - out_l^{(1)} \right) . in_h$$

Since random input is taken for every trial we can say that the error that we get can be assumed to be approximate to the value that actually is when we consider the entire data set.

GRAPH:



Here from the graph we can say that as we increase the number of hidden nodes then the error decreases from about 0.09 to 0.05 at 80 hidden number of nodes and then it increases again for 90 and 100 hidden number of nodes.

ERROR PERCENTAGES

Error: 0.055

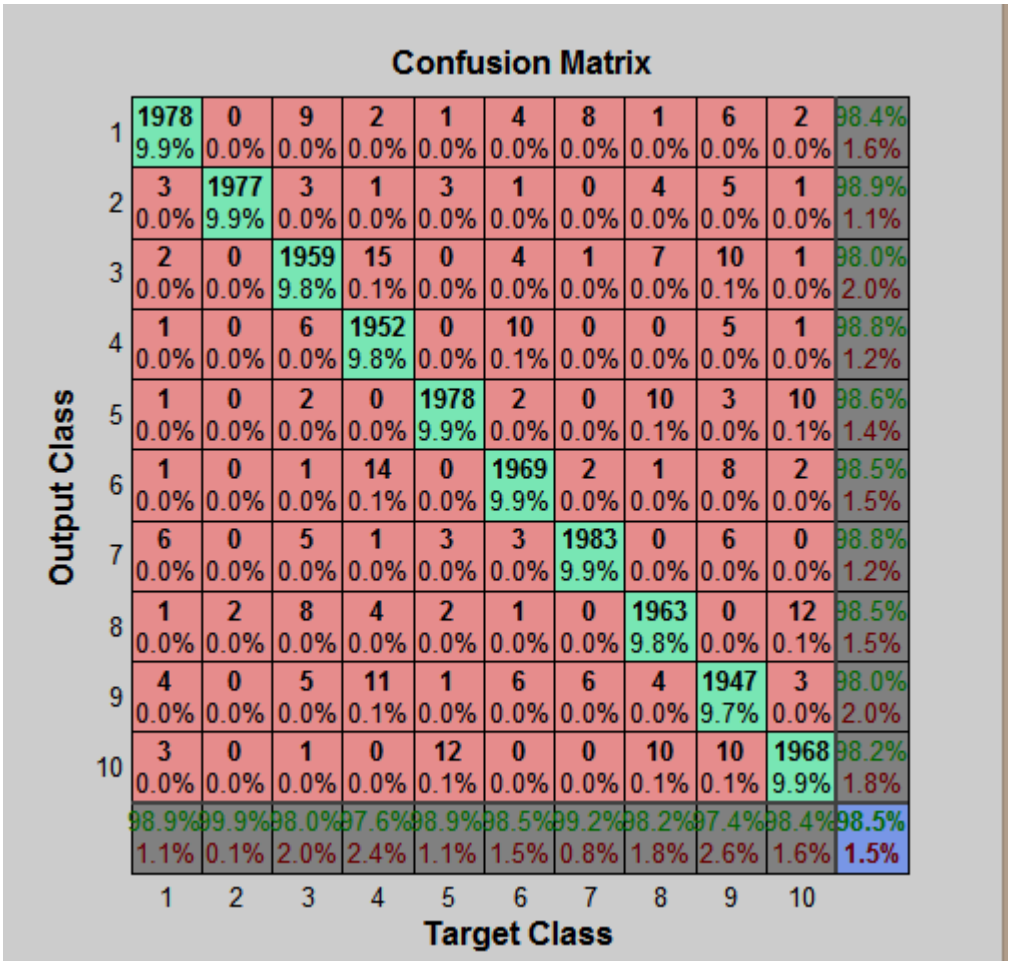
Learning Rate: 0.01

No of Hidden Nodes: 80

Reciprocal Rank Percentage: 97.65%

NEURAL NETWORK PACKAGE:

Neural network package is used in this method making the use of Pattern Recognition tool in the neural network package of matlab.



From the confusion matrix we can see that the error percentage for test data is 1.5% with the digit 8 having highest error and the digit 1 having least error. The amount of deviation is not a lot in the digits and thus we can say that the model is well trained.

ERROR PERCENTAGES

Testing Percentage Error: 1.5%

CONCLUSION: Best performance was given by NN package followed by Linear Regression followed by NN model.