Project: 2 Classification of Handwritten Numerals

CSE 574: Machine Learning

SUBMITTED BY:

Ankit Jain

5009 7432

Project: 2 Classification of Handwritten Numerals

Objective:

To implement and evaluate classification algorithms and compare their performance with a publicly available package.

Data Source:

GSC features extracted from CEDAR/UB for Handwritten Numerals.

Models Used:

1. *Logistic Regression:*

Soft-Max Function

$$p(\mathcal{C}_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

Cross – Entropy Error Function

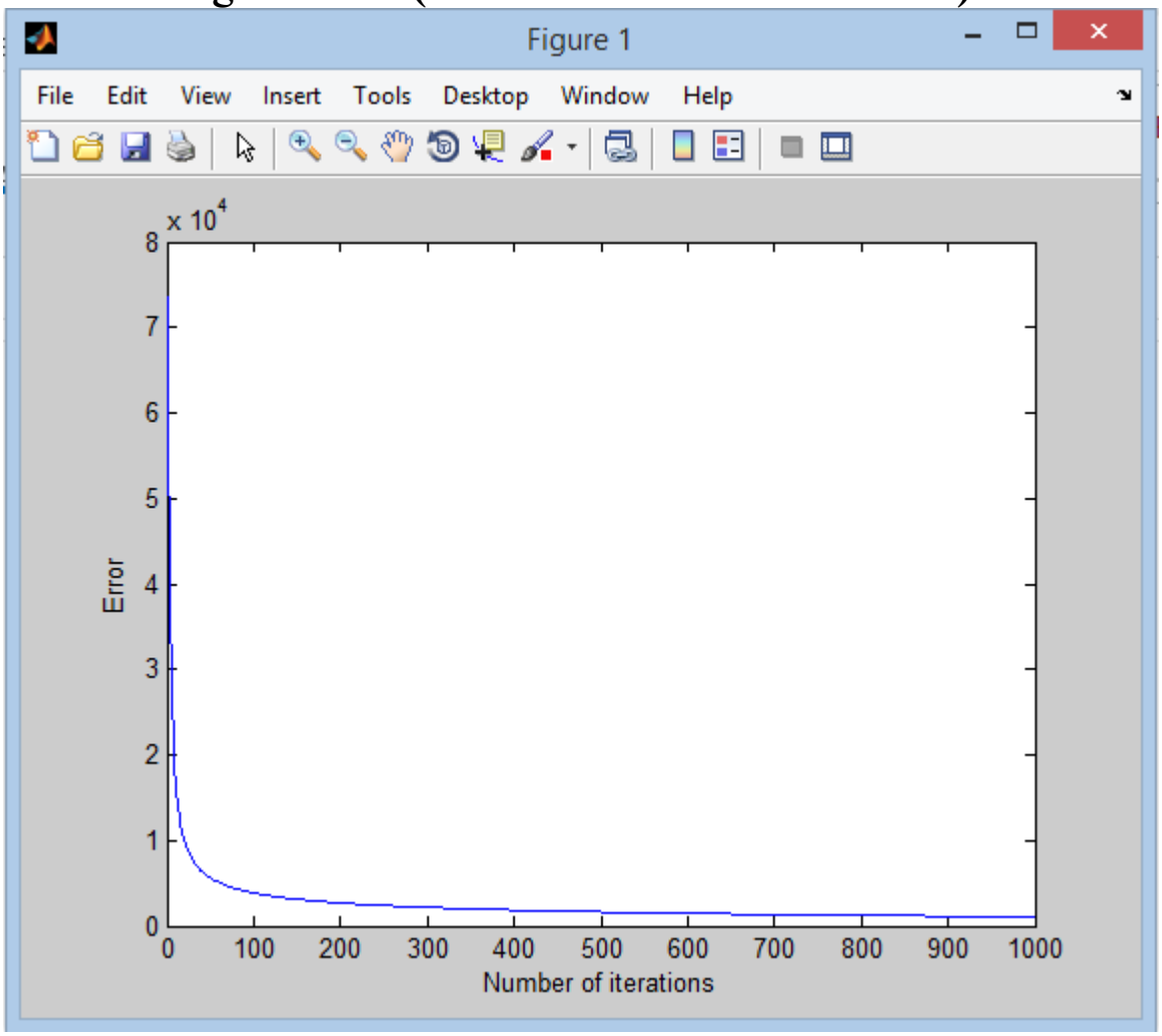
$$E(\mathrm{w}_1, \ldots, \mathrm{w}_K) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

Logistic regression is a probabilistic, linear classifier. It is parameterized by a weight matrix W and a bias vector b. Classification is done by projecting an input vector onto a set of hyper planes, each of which corresponds to a class. The distance from the input to a hyper plane reflects the probability that the input is a member of the corresponding class.

Implementation Idea:

The misclassification rate I achieved for logistic regression is ***3.2%***. I used the softmax function to implement the logistic regression, this function gives normalized output and thus probabilistic treatment to the classifier output. I trained the logistic regression model using Gradient descent approach i.e. *W_new=W_old - step_size*gradient of error.*

Error Convergence Plot (Error vs Number of iterations):



Analysis:

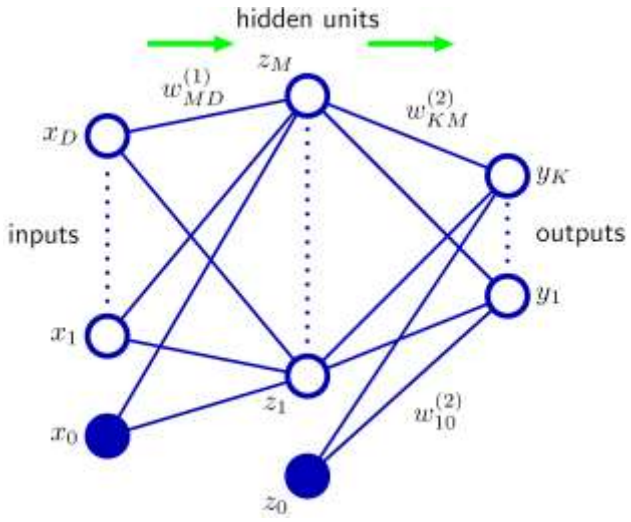
Error drops with growing number of iterations until it starts increasing.

Parameters Used:

Parameter	Initial Value	Step Size	Final Value
eta	0.00003	0.75*eta	-
No. of iterations	1	1	1000

Misclassification (Error) Rate: 3.26%
Accuracy: 96.74%
Training Time: 10 min (approx.)

2. *Neural Network Model:*



Soft-Max Function (Hidden Layer)

$$p(\mathcal{C}_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

Linear (Output Layer)

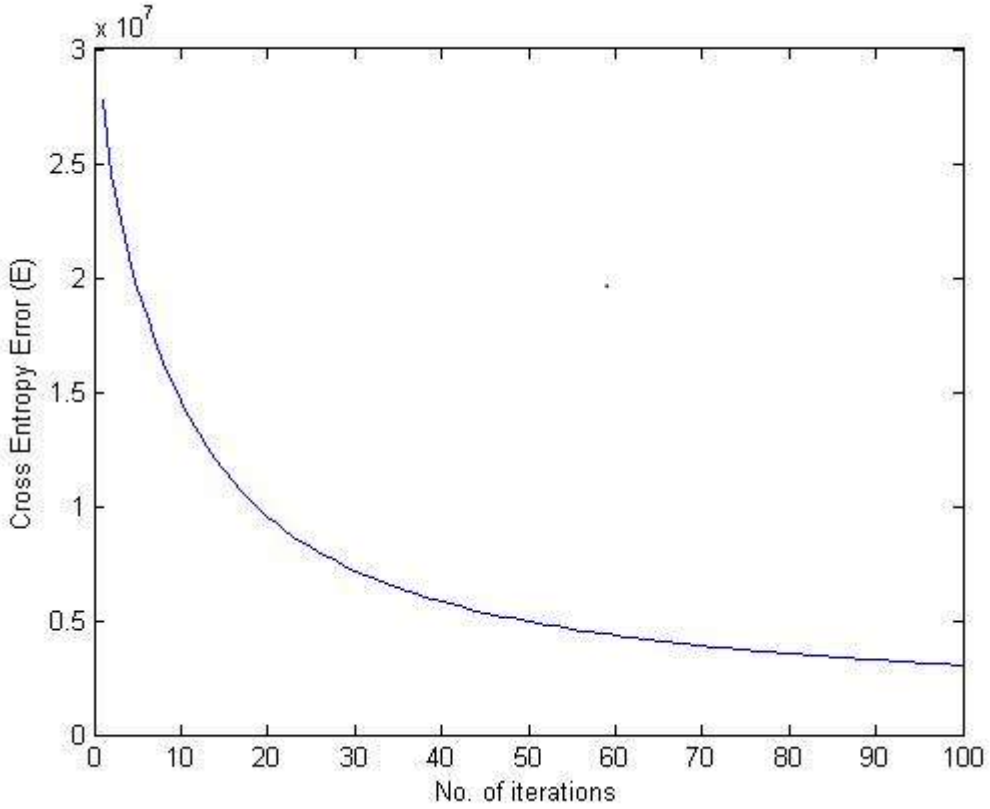
$$y_k = a_k$$

Neural networks are typically organized in layers. Layers are made up of a number of interconnected 'nodes' which contain an 'activation function'. Patterns are presented to the network via the 'input layer', which communicates to one or more 'hidden layers' where the actual processing is done via a system of weighted 'connections'. The hidden layers then link to an 'output layer' where the answer is output.

Implementation Idea:

The misclassification rate I achieved for logistic regression is ***5.8%***. I used the tanh function as the activation function of the hidden layer and softmax function as the activation function for the output layer. I used the error back propagation algorithm to communicate the error to the hidden layers and gradient descent algorithm to reduce to find W's which have very low value of error.

Error Convergence Plot (Error vs Number of iterations):



Analysis:

Error drops with growing number of iterations until it starts increasing.

Parameters Used:

Parameter	Initial Value	Step Size	Final Value
eta	0.00003	0.75*eta	-
No. of iterations	1	1	100

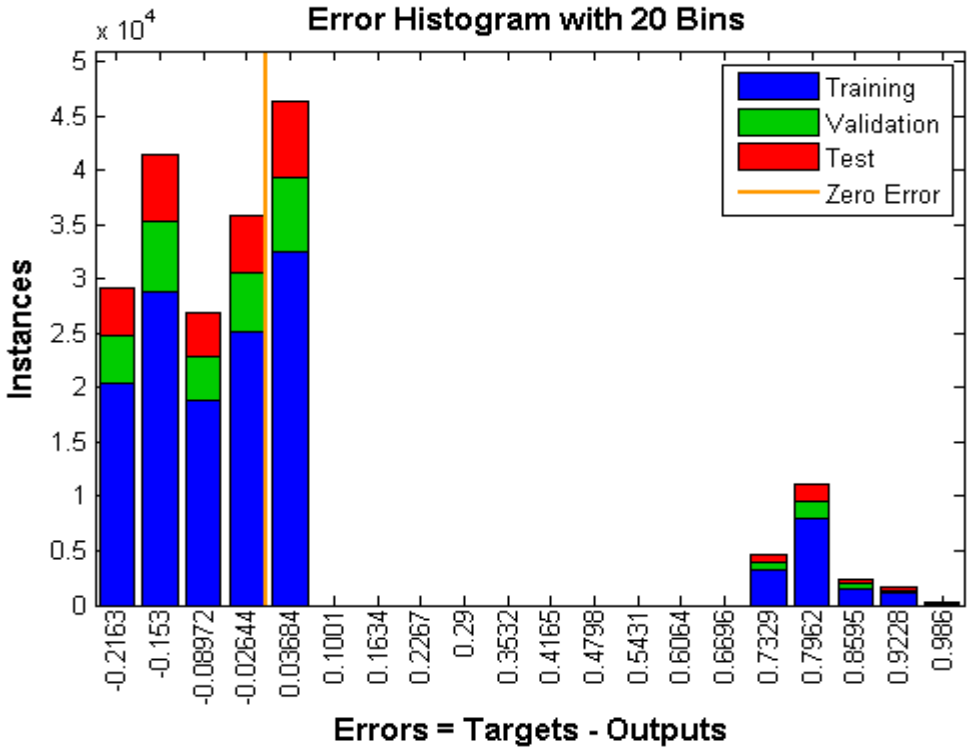
Hidden neurons (M): 20
Training Time: 30 min (approx.)

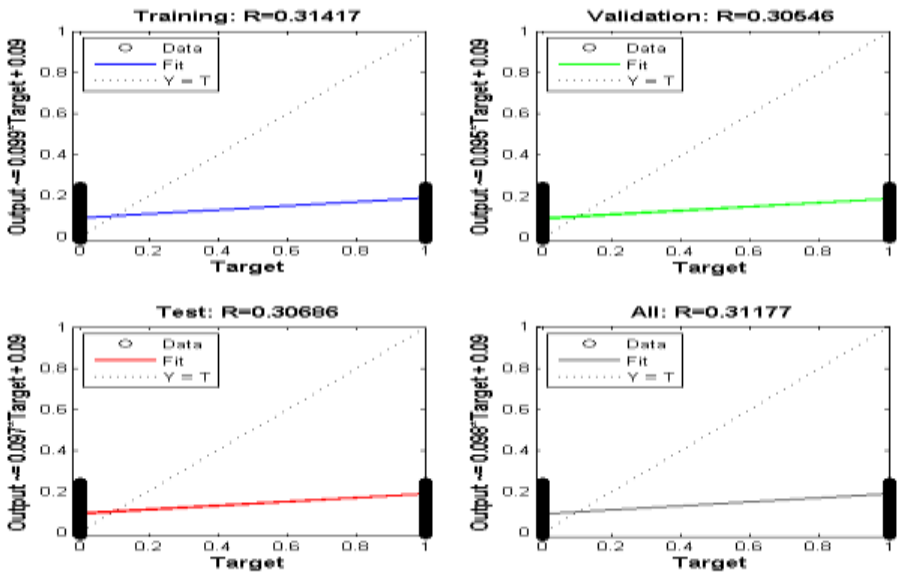
Misclassification (Error) Rate: 5.88%
Accuracy: 94.12%

3. Neural Network Package (nftool package):

Implementing project using packages: Neural Network toolbox of Matlab.

Hidden Neurons: 10





Comparison of models:

Parameter	Logistic Regression	Neural Network	Neural Network Package
Training time	10 mins (approx)	30 mins (approx)	60 mins(approx)
Error Rate	3.26 %	5.88%	9%
Accuracy	96.74%	94.12%	91%