HANDWRITTEN DIGIT RECOGNITION USING CLASSIFICATION

By

Aravind Kumar Ramesh
CSE574-Introduction to Machine Learning
Dr. S N Srihari
STATE UNIVERSITY OF NEW YORK-
UNIVERSITY AT BUFFALO

OVERVIEW

The project implements a handwritten digit recognition using Logistic Regression and Neural Networks. The input dataset consists of GSC features extracted from ten digits (0-9). The task of classification attempts to implement Logistic Regression and Neural Networks and compare the performance of the same with publicly available package for neural networks.

LOGISTIC REGRESSION

We use a training data that contains the GSC features extracted from images of digits 0-9. The dataset contains 512 features for each image and hence these are used to construct a design matrix. We divide the dataset into training and testing data and their respective design matrix is constructed. The design matrix is constructed in such a way that the first column of the matrix is allocated to include the bias term (1 in this case). We choose a 1of K target matrix wherein for a data belonging to class C1 (digit 0), we assign t to be (1,0,0,0,0,0,0,0,0,0). The target vector for other classes are done similarly. Through logistic regression we compute the posterior probability. The formula for posterior probability can be given as

$$p(C_1|\phi) = y(\phi) = \sigma(\mathrm{w}^T\phi)$$

Where W= Weight matrix, and **Φ** =Design Matrix. We apply the logistic sigmoid function as the activation function to the input features. Our goal here is to obtain the optimum W for which the error in minimized.

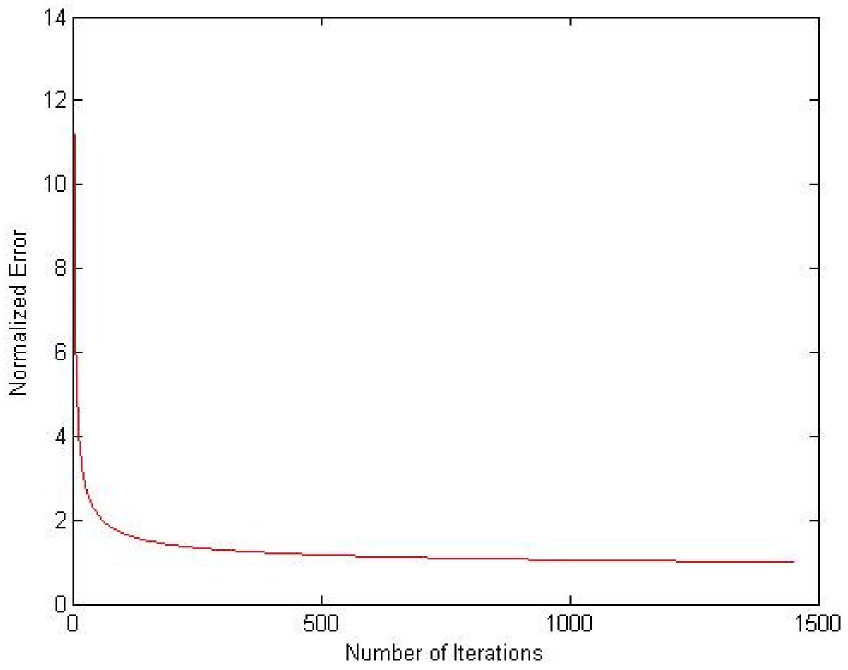
In order to minimize the error, we use the gradient descent approach wherein we choose a random number of iterations and by assigning a learning rate value, we descent down so as to reach a point of global minima.

The formula for Gradient descent is given as follows

$$\nabla E = t \times \log\left(\sigma(w^T\varphi)\right) + (1-t) \times \log(1-\sigma(w^T\varphi))$$

Where t=target vector. The term is also referred to as entropy error.

We now sum up the entropy error for every iterations. In order to make the plot easy, we proceed towards normalization where in this particular case we have chosen the error value to be normalized between a range of 1 and 14. The error values are then plotted with respect to the iteration numbers.



The above graph shows the convergence of the gradient descent. Hence at the end of total number of iterations, we obtain a weight matrix that gives the least error. With the weight matrix that we obtained by applying the gradient descent we proceed with the testing phase of logistic regression.

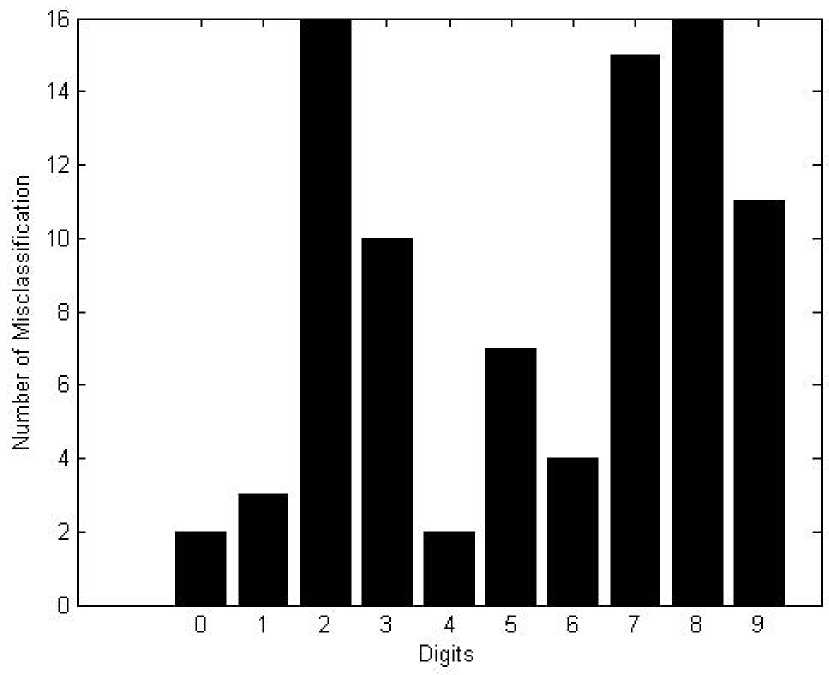
LOGISTIC REGRESSION TESTING

In order to carry out testing, we use the testing dataset and applying a non-linear transformation using a logistic sigmoid function as

$$y(\phi) = \sigma(\mathrm{w}^T\phi)$$

Where W is the weight matrix we obtained after the training. We now apply a simple decision logic wherein the y(**Φ**) is assigned the corresponding class label if the value of y(**Φ**) is greater than 0.5. Thus the result of this process is a matrix that is very similar to the target matrix containing the class label.

In order to find the error, we subtract the decision made with the target vector t. This gives us the number of misclassifications.



The following figure above shows the number of misclassifications for the respective digits from 0 to 9. Thus the total misclassification error for the system can then be simply found out by summing up the misclassification of digits from 0 to 9 and dividing it by the total number of testing data samples. The Error rate is **5.9333%**

NEURAL NETWORKS

For this particular approach we make use of the feed forward neural network with back propagation. We make use of a 2-layer neural network where the hidden layer contains 100 nodes (a rough heuristics) and the output layer contains 10 nodes corresponding to digits from 0 to 9. In order to begin, we import the input data set and assign random weight values for wh and wo where wh is the weight of the hidden layer and wo is the weight of the output layer.

For the hidden layer, we apply a non-linear transformation given by the following formula

The input to the hidden layer is

$$a_h = w_h \times X^T$$

The output to the hidden layer is

$$z_h = \sigma(a_h).$$

Input to the output layer

$$a_o = w_o \times z_h$$

Output of the output layer

$$z_o = \sigma(a_o).$$

With these parameters, we evaluate the back propagation and back propagate the error value from the output layer to the input layer.

$$\delta_o = (z_0 - t^T)$$

$$\delta_h = (w_o^T \times \delta_o)(\sigma(a_h))(1 - \sigma(a_h))$$

The error of the system is calculated by

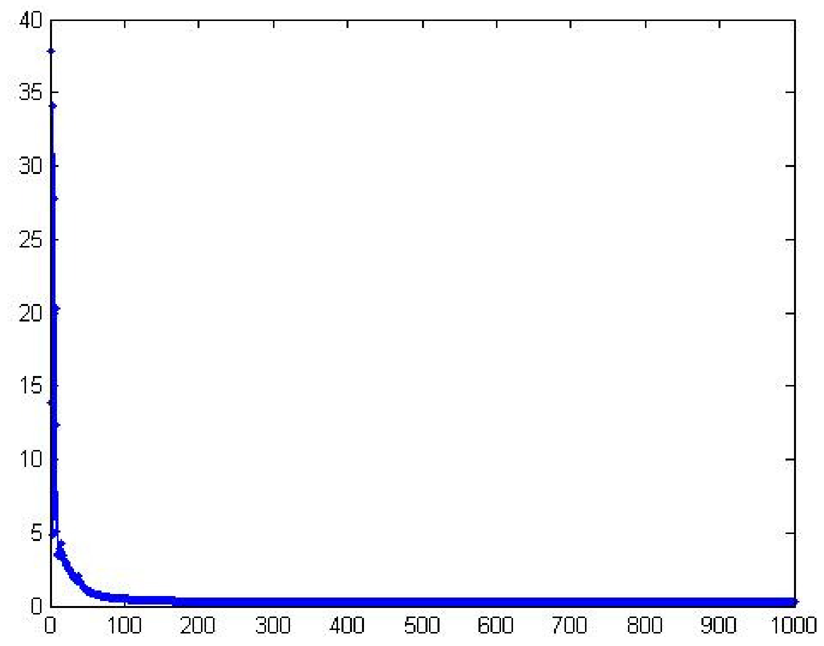
$$\nabla E = t \times \log\left(\sigma(w_o^T X)\right) + (1 - t) \times \log(1 - \sigma(w_o^T X))$$

Using these values we compute the error using the IRLS equation as follows

$$w_o^{new} = w_o^{old} - \varepsilon \times \Delta_o$$

$$w_h^{new} = w_h^{old} - \varepsilon \times \Delta_h$$

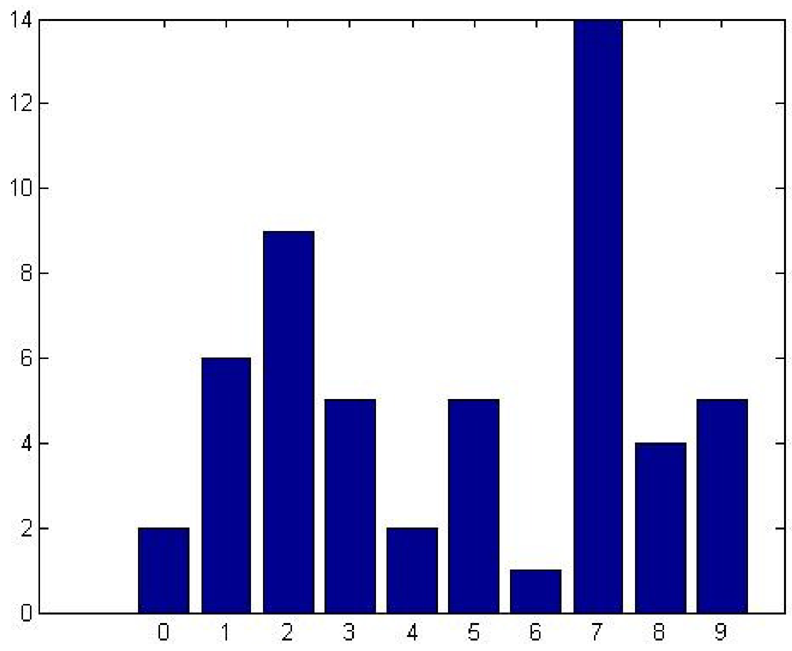
The IRLS equation is applied till the graph between error and iterations converges and this is guaranteed to give us the minimum error and optimum weight.



In the testing phase, we make use of the optimum weight matrix that we applied in the training phase, to compute the output Yn, as

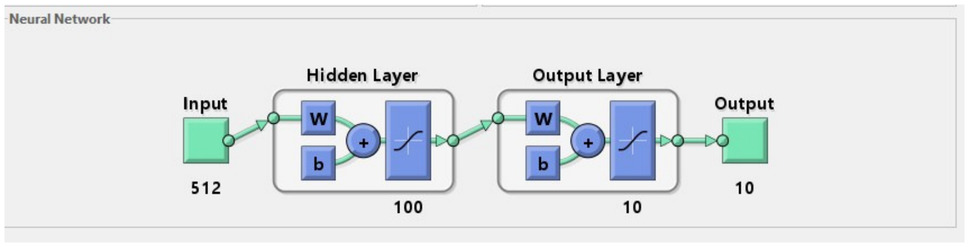
$$y(\phi) = \sigma(\mathrm{w}^T\phi)$$

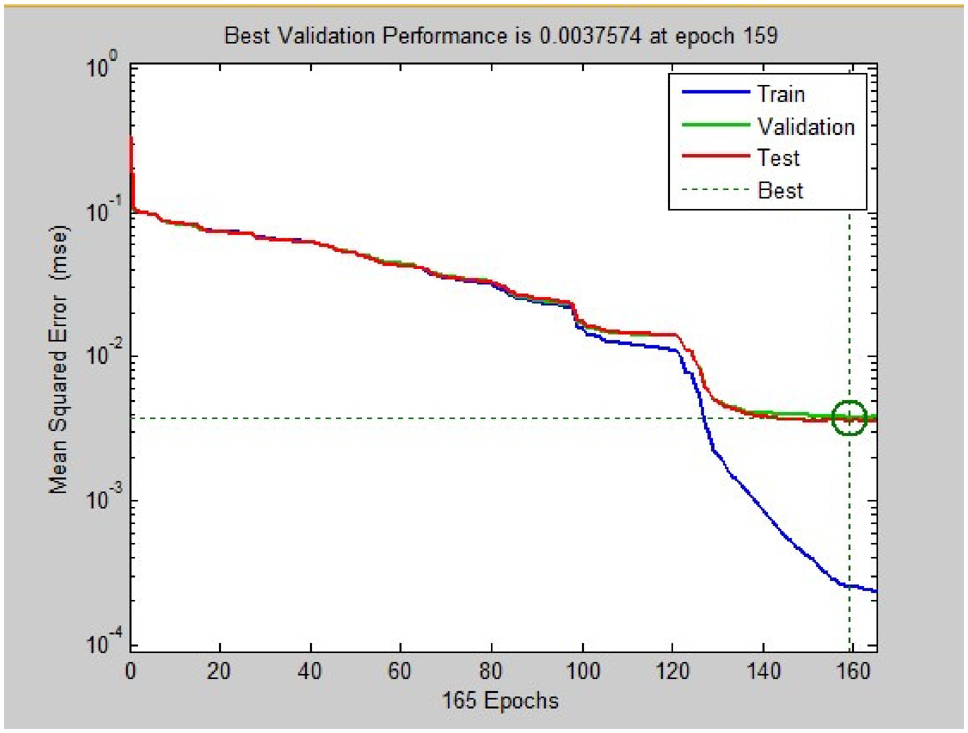
We now apply the same decision logic that we applied for logistic regression by assigning a class label corresponding to values that are greater than 0.5 in Yn. With this we calculate the misclassifications. The following graph illustrates the misclassifications for individual digits from 0 to 9. The overall error rate obtained is **3.55%**

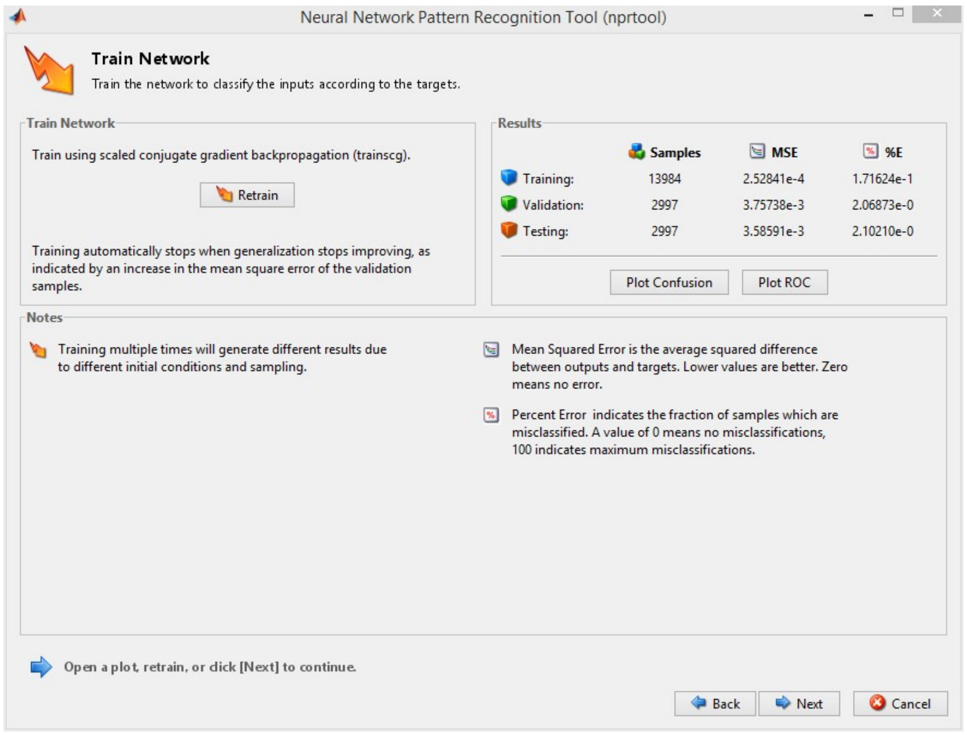


NEURAL NETWORK TOOLBOX- MATLAB

In order to compare the performance of our implementation with a standard toolbox, we have made use of the Neural Network Toolbox in Matlab.







Train Network

Train the network to classify the inputs according to the targets.

Train Network

Train using scaled conjugate gradient backpropagation (trainscg).

Training automatically stops when generalization stops improving, as indicated by an increase in the mean square error of the validation samples.

Results

	Samples	MSE	%E
Training:	13984	2.52841e-4	1.71624e-1
Validation:	2997	3.75738e-3	2.06873e-0
Testing:	2997	3.58591e-3	2.10210e-0

Notes

Training multiple times will generate different results due to different initial conditions and sampling.

Mean Squared Error is the average squared difference between outputs and targets. Lower values are better. Zero means no error.

Percent Error indicates the fraction of samples which are misclassified. A value of 0 means no misclassifications, 100 indicates maximum misclassifications.

Open a plot, retrain, or click [Next] to continue.

The input dataset for training and test were fed into the toolbox and the number of hidden layer was set at 100. The RMS Error was found to be follows

$E_{training}$: 0.17 %

$E_{validation}$: 2.06 %

$E_{testing}$: 2.102 %

RESULT AND CONCLUSION

With the results obtained we conclude that Neural Networks offers better error rate as compared to logistic regression and hence as a result is more accurate at recognizing hand written digits as compared to logistic regression. The error rates obtained are as follows.

$E_{LOGISTIC\ REGRESSION}$ = 5.933 %

E_{NN} = 3.55%

$E_{NN\text{-}TOOLBOX}$ = 2.102%