Project 2:

Classification

Project Report
CSE 574: Machine Learning
Due 6th December, 2013

Ravi Karan Sharma
50097193

Introduction

In this project we perform classification on a set of images of handwritten numerals using the techniques of Logistic Regression and Neural Network Back Propagation. A standard available package is also used to compare results with the techniques implemented through code.

Linear Regression Classification Model

In this technique we use the training data provided to tune the model by the weights for each parameter which are 512 characteristics of the images extracted using feature extraction programs. Here we also add a bias of value 1 to each parameter, to give us a feature space of 513 dimensions in all. In Logistic Regression, the posterior probability of class C_1 can be written as a logistic sigmoid acting on a linear function of the feature vector φ so that

$$p(C_1|\varphi) = y(\varphi) = \sigma(w^T\varphi)$$

Here w are the weights that we will be tuning using the gradient step method. C_1 can be assumed to be the class of digit 0, C_2 of digit 1 and so on. Thus, in this implementation we begin with a random set of values for w, and calculate $y(\varphi)$. We then use the error function as the negative log probability to find the convergence of the error to a stable value.

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

Here t is the target vector, which is constructed using the 1 of k classification. Hence the target vector for a sample belonging to C_1 would be ⟨1,0,0,0,0,0,0,0,0,0⟩ and so on for all the classes. Then, we use the gradient step function to find new set of weights to be used in the next iteration of the above process as

$$s = \sigma(w^T\varphi) - t$$

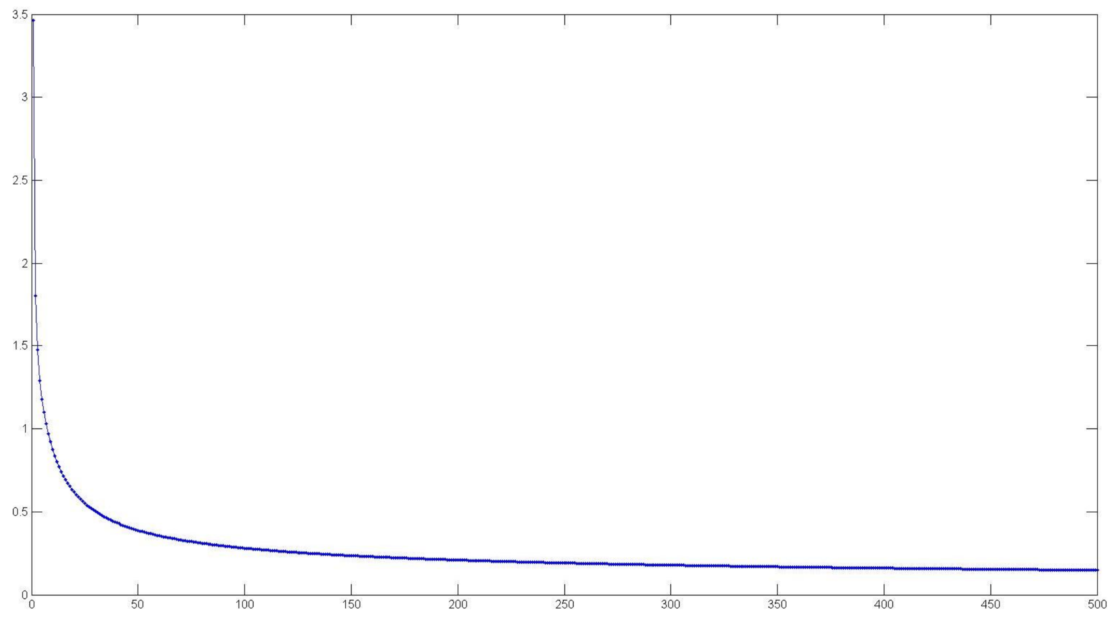
S is the gradient which measures the difference between the current value of $p(C_1|\varphi)$ and the target t.

$$g = \varphi \times s$$

$$w_{new} = w_{old} - \varepsilon \times g$$

Where ε is the gradient step size, also called the learning rate. ε determines the period with which the linear regression model changes the weight with the gradient.

We get the following plot between the error and the number of iterations, shown at 500 iterations:



The result shows that the error decreases steeply at first and then gradually tapers down to a near constant rate. The final matrix of weights obtained was used on the testing set and it mis-classified **143 samples out of 1500 giving an error of 9.53 %.**

Feed Forward Neural Network with Back Propagation

In this technique instead of using the samples to calculate $p(C_1|\varphi)$, we feed them to a 2 layer neural network having 1 hidden layer and 1 output layer. The hidden layer contains 50 nodes and the output layer contains 10 nodes, one for each class. The samples are drawn randomly from the training set. We follow the forward propagation with the back propagation algorithm on all the samples. This cycle is repeated a number of times.

For the hidden layer, we have the set of weights assumed to be w_h as random values initially and the for the output layer the set of weights w_o are similarly assumed to be random values. Then by the forward propagation algorithm, we find

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

and the output of the hidden layer

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

We similarly perform the same calculations for the output layer with

$$a_o = w_o \times z_h$$

and the output of the neural network as

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

We proceed with back propagation by calculating δ_o for the hidden layer as

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

We back-propagate the error from the output layer to the hidden layer similarly and find δ_h for the hidden layer

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

Using these values we find the gradient for the weights W_o and w_h of the hidden and output layers as:

$$\Delta_h = \delta_h \times X^T$$

$$\Delta_o = \delta_o \times z_h$$

Hence the updated values of the weights are:

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

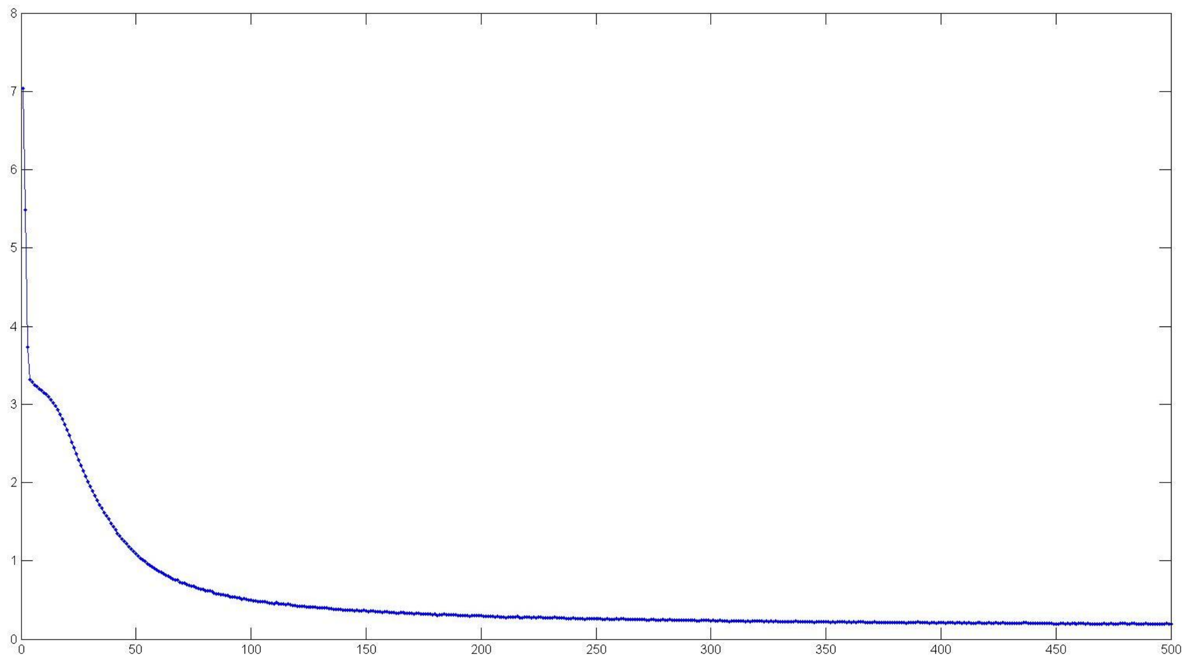
$$w_h^{new} = w_h^{old} - \varepsilon \times \Delta_h$$

Using the updated weights we operate on the next sample withdrawn randomly.

This process was repeated a number of times until the error calculated as below converges.

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

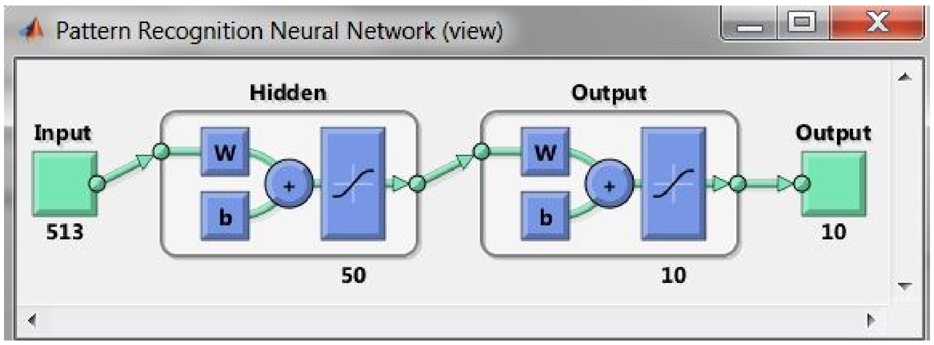
The graph for the above result of Error vs. the number of iterations of repeating the whole process 500 times, was found to show the following relation:



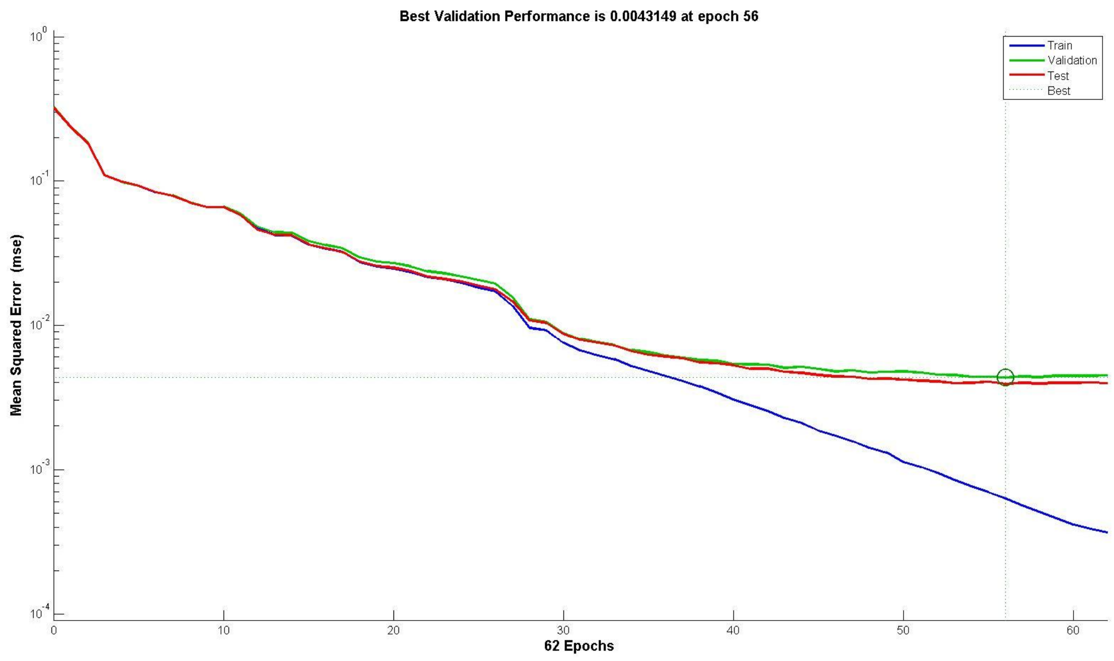
The value of the error was found to decrease rapidly in the beginning but then it keeps decreasing slowly and gradually as the number of iterations increase. We use the weights for the output and the hidden layer of the network found at the end of all iterations on the testing set and it was found that the network **misclassified 48 samples out of 1500 giving a percentage error rate of 3.2%.**

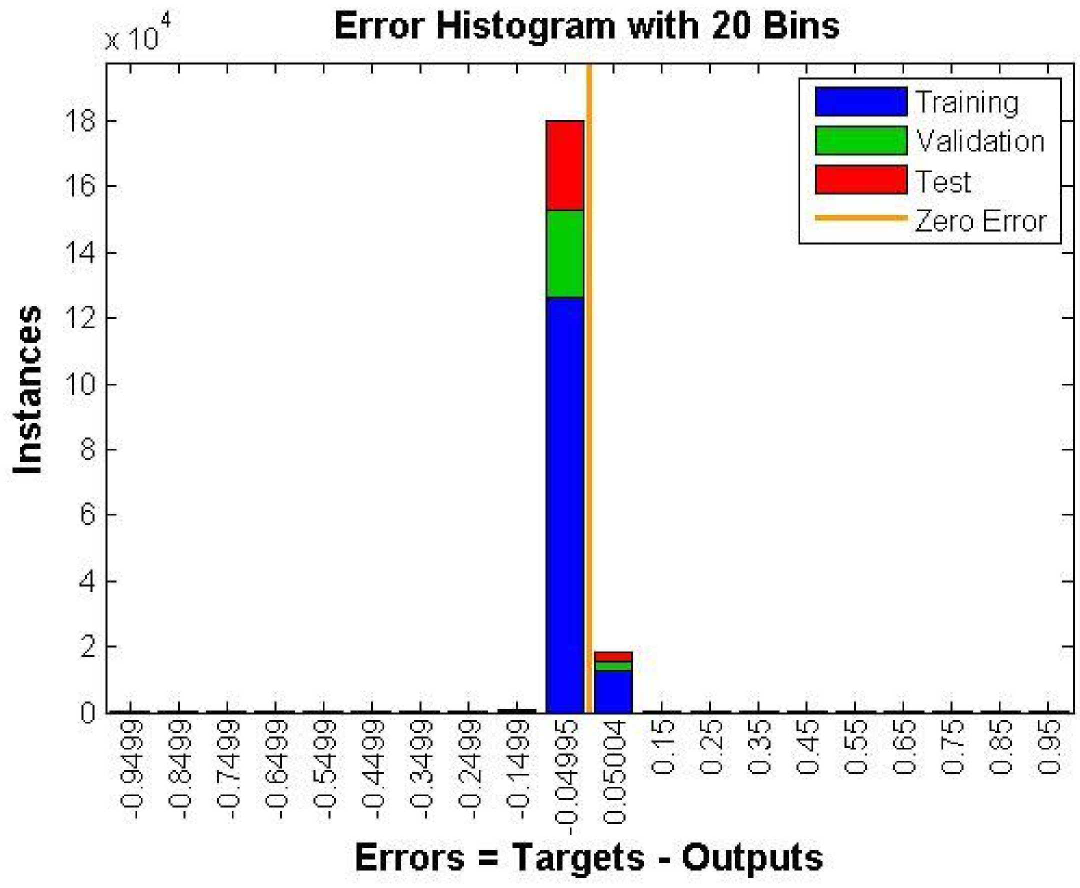
Standard Package: Neural Network package with MATLAB

The standard package available for neural networks with MATLAB were used and trained the network with the training set and calculated the error on the testing set given. A network of 50 hidden layer nodes was used similar to the network created through code before.



The network was trained on the training set and following results were obtained:





The following error rates were obtained:

$\%E_{TRAINING}$ = 0.300343 %

$\%E_{VALIDATION}$ = 2.3023 %

$\%E_{TESTING}$ = 2.2022 %

Conclusion

Amongst the methods implemented through code, Neural Networks with Back Propagation showed better accuracy in classification than Linear Regression. However the standard package used from the Neural Network Toolbox on MATLAB gives an even lower error rate than the code-implemented Neural Network with backpropagation.