CSE 574 – PROJECT 2 REPORT

Classification of Handwritten Numerals

Name: Animesh Kumar
UBID: 50134753
Email: animeshk@buffalo.edu

INTRODUCTION

Given a set of handwritten numerals corresponding to GSC feature set, train classification models on training data using Logistic Regression and Neural Network to predict class labels for given inputs and predict class labels (digit) for new unseen inputs from the test data. Also, evaluate a publicly available standard classification package and compare the performance with the various classification algorithms implemented.

Datasets:

1) Training Data: Training set consists of 19987 samples corresponding to digits from 0 to 9 where each sample is represented by 512 bit GSC corresponding feature vector. Each digit has 2000 samples available for training. Our classification algorithms will learn on the training set to predict the classes for given input.
2) Test Data: Test Data consists of 1500 samples corresponding to digits from 0 to 9 as in Training data set. Our learnt classification algorithms will take the samples from test data and try to predict the class labels for given inputs. Accuracy of the algorithm will be determined by the number of classes correctly classified out of the total number of classes classified.

APPROACH

We worked on two approaches for implementing the classification algorithms:-
1) Logistic Regression
2) Neural Network

The results obtained from the above approaches were compared with result from a standard classification package for Neural Network.

LOGISTIC REGRESSION

We use Softmax as activation function to classify as 1-of-K coding scheme. Softmax function calculates posterior probabilities of K-classes between 0 and 1. Class having maximum posterior probability will be assigned as the classified class for the given input sample.

We first initialize the weight vector to random vector between 0 and 1. This initial vector is randomized with very small values between 0.005 and 0.01 to avoid any possible occurrence of out of memory exception.

Once we have the initial weight matrix, it is multiplied with the Input Matrix to devise the Activation Function. This activation function is then used to calculate the Softmax function by:-

$$Y(k,:) = \frac{\exp(Activation\_Matrix(k,:))}{\sum \exp(Activation_{\_Matrix(j,:)})}$$

Using the Softmax function, we try to calculate ∇E_W by:-

$$\nabla E_W = Input_{Matrix} * (Y - Target)$$

We find out the classes correctly classified and try to increase the number by modifying the weight vectors.

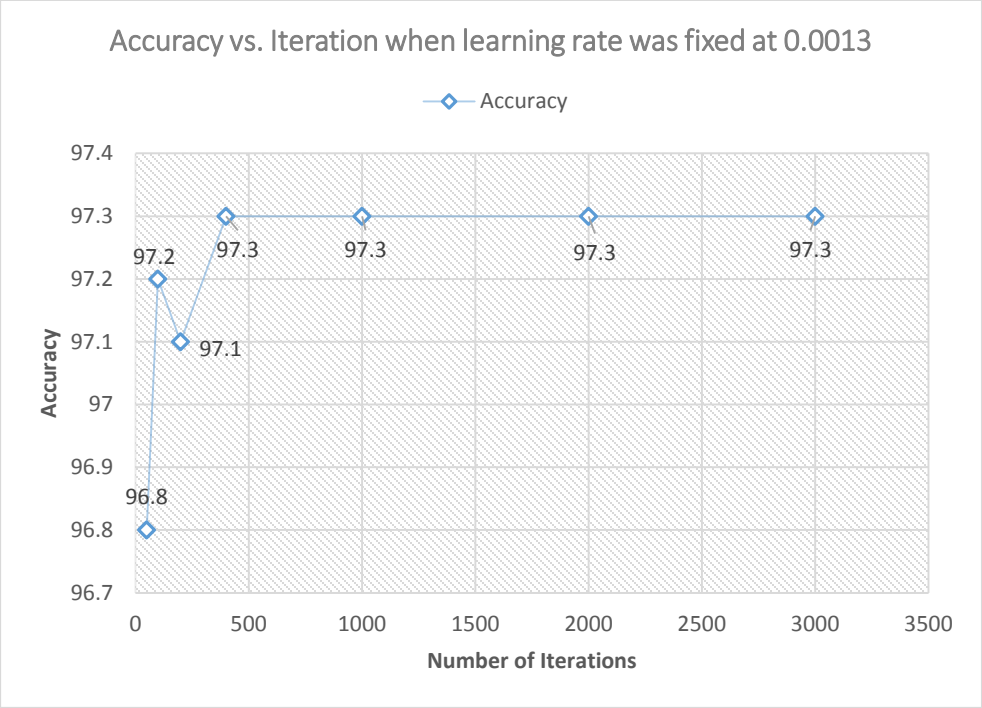
$$W^{new} = W^{old} - learning_{rate} * \nabla E_W$$

These weight vectors are used to recalculate the Activation function and then the above steps are repeated till we get a suitable Accuracy.

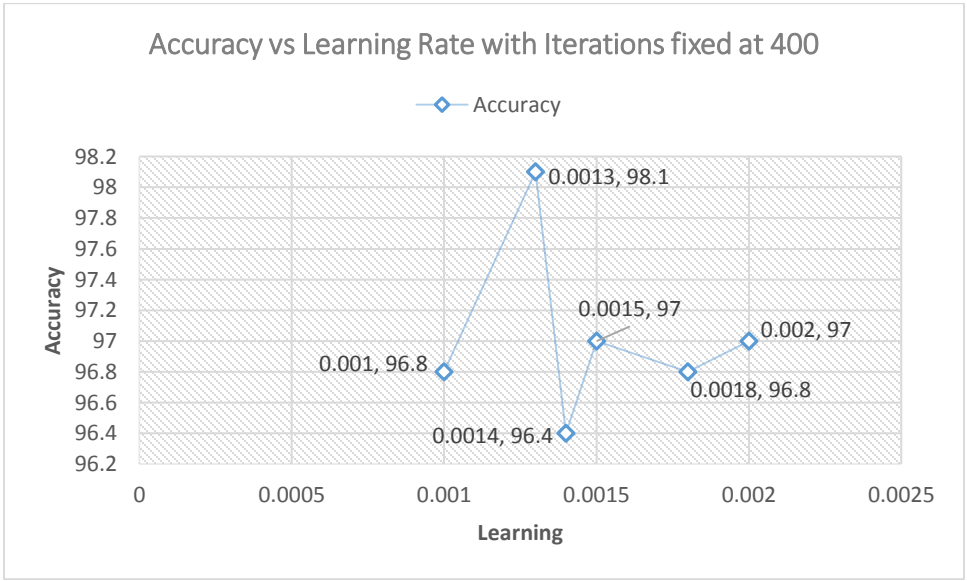
$$Accuracy = \frac{Number\ of\ samples\ correctly\ classified}{Total\ number\ of\ samples}$$

We try to tune the parameters by varying the number of iterations to be done with a fixed learning rate and then once the number of iterations is fixed, we tune the learning rate to give the best accuracy possible.

With a fixed learning rate of 0.0013, we receive the best accuracy at 100 iterations. We received an accuracy of 97.2% i.e., out of a total of 19978, 19919 samples were correctly identified. The error rate was 2.8%.



Once the number of iterations is fixed, we try to tune the learning rate to give the best results. We keep the learning rate between a very small intervals of 0.001 to 0.002 to avoid out of memory exceptions.



Final Accuracy obtained on the testing data set with learning rate at 0.0013 and 400 iterations is **97.8%** i.e., 1467 out 1500 samples were classified correctly. The error rate is 2.2%.

Neural Network

In neural network implementation we have created a network with only one hidden layer. The number of nodes (neurons) in this hidden layer is a parameter which is tuned to provide us with best accuracy. The activation function for hidden layer is calculated by the hyperbolic tangent function.

We first start with 100 neurons in the hidden layer and start the forward propagation by choosing random weight vectors for $W^{(1)}$ (for hidden layer) and $W^{(2)}$ (for output layer). We calculate the output of hidden layer as follows:-

$$Z_{Matrix} = \tanh(Weight\_ji_{Matrix} * \ Input_{Matrix}{}^{T})$$

The output from hidden layer is then passed to the output layer:-

$$Y_{Matrix} = Weight_{kj_{Matrix}} * Z_{Matrix}$$

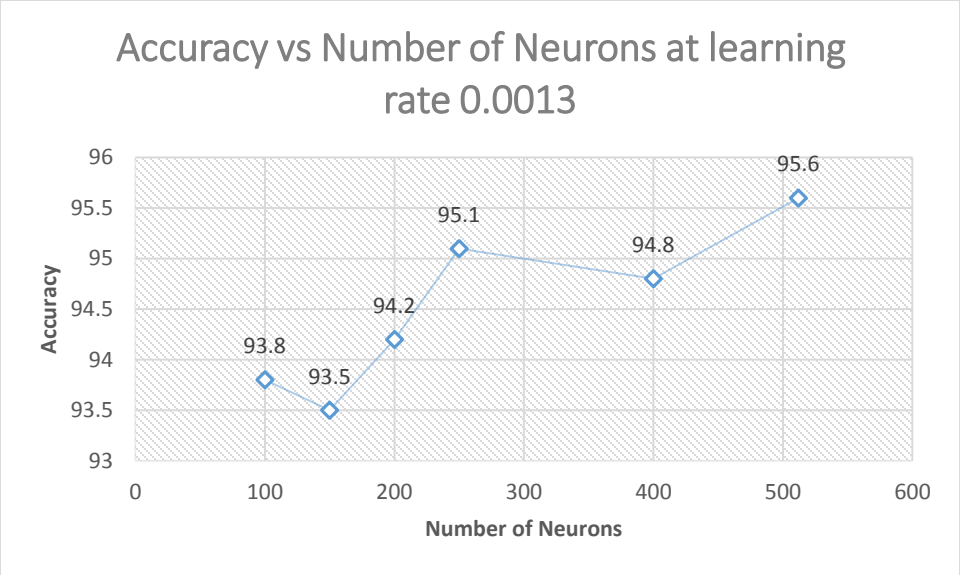
Once we have the output at the output layer, we start Back Propagation to adjust the weight parameters for the hidden layer as well as the output layer and hence improve the accuracy of the classification.

In back propagation, we first calculate δ_k which states the adjustment to be done in the weights of the output layer and then adjustments required in the hidden layer.

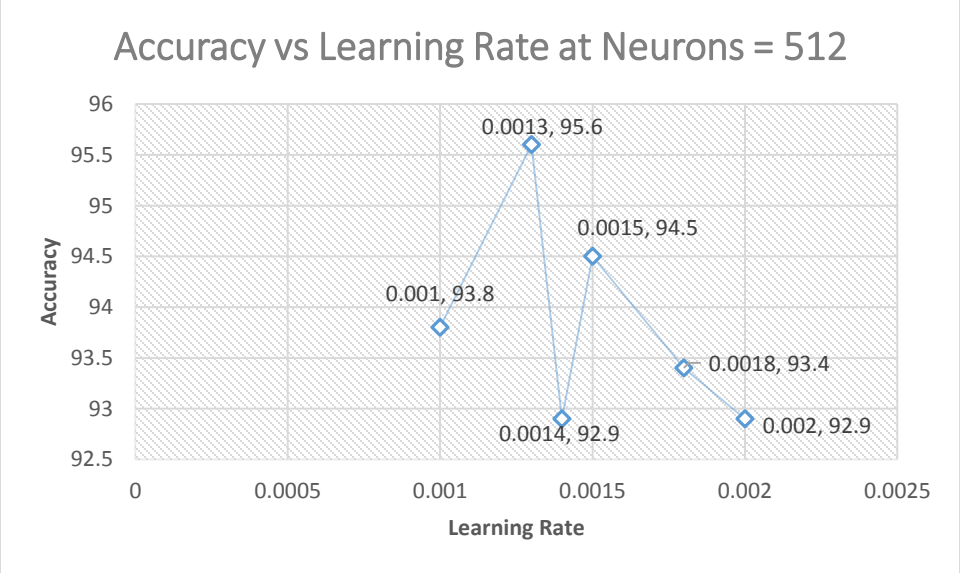
$$\delta_k = Y - Target$$
$$\delta_j = (1 - z_j{}^2) .* \ Weight_{kj_{Matrix}}{}^{T} * \ \partial_k$$
$$\partial_k = \delta_k * Input$$
$$\partial_j = \delta_j \ * \ z_j{}^{T}$$
$$Weight_{ji_{Matrix}} = \ Weight_{ji_{Matrix}} - learning_{rate} * \partial_j$$
$$Weight_{kj_{Matrix}} = \ Weight_{kj_{Matrix}} - learning_{rate} * \partial_k$$

We have introduced a learning parameter to avoid any out of memory exceptions.

The above step is repeated for each sample such that algorithm learns to predict the classes based on the input given. Once accuracy is found for given neurons, we change the number of neurons and repeat the steps to check the accuracy. Fixing the learning rate at 0.0013, gives us the best accuracy at 512 neurons. We get an accuracy of 95.6% i.e., 1434 classes out of 1500 samples get classified correctly.

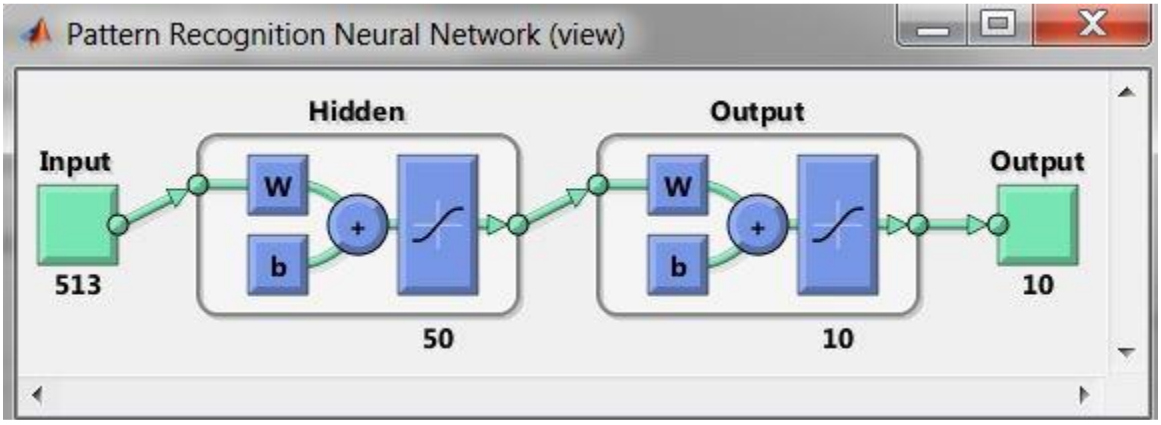


Once we have fixed the number of neurons, we try to tune the learning rate parameter using 512 neurons. We get the best accuracy at 0.0013 learning rate.



Hence, with a neural network of 1 hidden layer, 512 neurons and fixed learning rate of 0.0013, we get an accuracy of 95.6% i.e., 1434 samples were correctly classified out of a total of 1500 samples in test data. The error rate is 4.4%.

Evaluating using Standard Package: Neural Network



With the standard neural network package of 1 hidden layer and 50 hidden neurons, we obtained the following error:-
Training Error = 0.300343 %
Validation Error = 2.3023 %
Testing Error = 2.2022 %

Conclusion

Based on the implementations done for classifying algorithms, we can conclude the classification accuracy as follows:-

1. Using logistic regression: 97.8%
2. Neural Network Implementation: 95.6%
3. Using standard package: 99.8%

Amongst the implementation done by us, Logistic Regression showed better accuracy. It's also faster than the neural network implementation and more efficient. Though amongst all evaluations, Standard package gave very accurate results.