CSE574 : Introduction to Machine Learning(Fall 2014)

**Project 2: Hand-written digit recognition**

**with Logistic Regression and Neural Networks**

Naveen Narayan

UB ID : nnarayan

UB Person # : 50134647

**Aim :** The aim of this project is to, given 512 features representing a hand-written digit, predict the corresponding digit. The hypotheses used to predict the output of a new sample is derived using Logistic Regression and Neural Networks.

**Implementation :**

**Logistic Regression :**

1. Loading data :

For each class, input data samples are loaded into a matrix. A corresponding target vector to represent the digit in the 1 of K ( K = 10 ) scheme is created.

The test data for all the 10 classes are loaded into a single matrix InputTrain ( 19,978 X 512 ).

Corresponding target vectors are also combined into a single matrix TargetTrain ( 19,978 X 10 ).

A bias unit of 1 is added to each row of InputTrain.

1. Gradient Descent :

We perform gradient descent for 1000 iterations.

In each iteration, for each input sample we calculate the target vector.

The hypotheses function for Logistic Regression is -



Parameters are derived using Error gradient -



We also calculate the Cross-entropy error for each iteration.



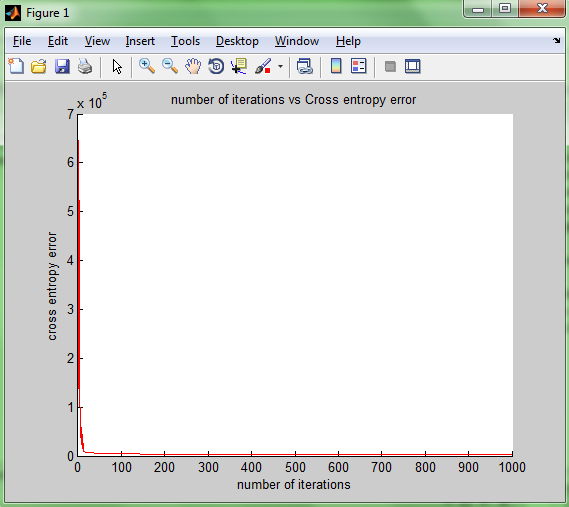


Fig 1 : Plot of number of iterations vs Cross-Entropy error

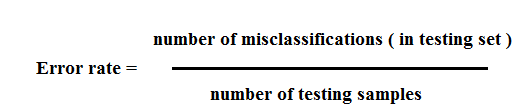
1. Testing :

Like training data, testing data is also loaded into a single matrix InputTest ( 1500 X 512 ).

Corresponding target vectors are created and combined into a single matrix TargetTest ( 1500 X 10 ).

We predict the classes for each testing sample using the hypotheses and the parameters derived from the training phase.

Error rate is calculated using -



Reciprocal rank is also calculated.

Number of misclassifications = 30

Number of testing samples = 1500

Error rate = 0.02 ( 2 % )

Reciprocal rank = 1.4831 X 103  ( 98.87 % )

1. Critical observations :
2. Initially, as the number of iterations increases, cross-entropy error decreases. The rate of decreases as the number of iterations increase.

=====================================================================

**Neural Networks :**

1. Loading data :

For each class, input data samples are loaded into a matrix. A corresponding target vector to represent the digit in the 1 of K ( K = 10 ) scheme is created.

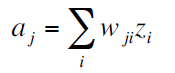
The test data for all the 10 classes are loaded into a single matrix InputTrain ( 19,978 X 512 ).

Corresponding target vectors are also combined into a single matrix TargetTrain ( 19,978 X 10 ).

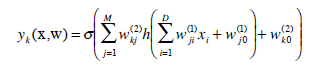
A bias unit of 1 is added to each row of InputTrain.

1. Forward propogation :

Randomly pick a row from the input data matrix. For this row, using the parameter matrix W1, compute the sigmoid value. This will be the input from the hidden layer to the output layer.







1. Back propogation :

To compute the error gradient, we start with the output layer and traverse backwards to the input layer.

For output layer –

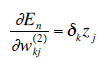


For hidden layers -



Calculate the error derivatives –





Update the Weight matrices –



Repea steps 2 and 3 for a few iterations.

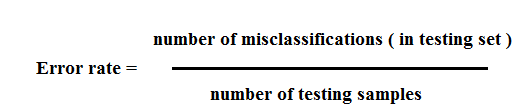
1. Tesing:

Like training data, testing data is also loaded into a single matrix InputTest ( 1500 X 512 ).

Corresponding target vectors are created and combined into a single matrix TargetTest ( 1500 X 10 ).

We predict the classes for each testing sample using the hypotheses and the parameters derived from the training phase.

Error rate is calculated using -



Reciprocal rank is also calculated.

Number of misclassifications = 36

Number of testing samples = 1500

Error rate = 0.024 ( 2.4 % )

Reciprocal rank = 98.6656 %

1. Critical observations : Training the Neural Networks takes considerably more time.

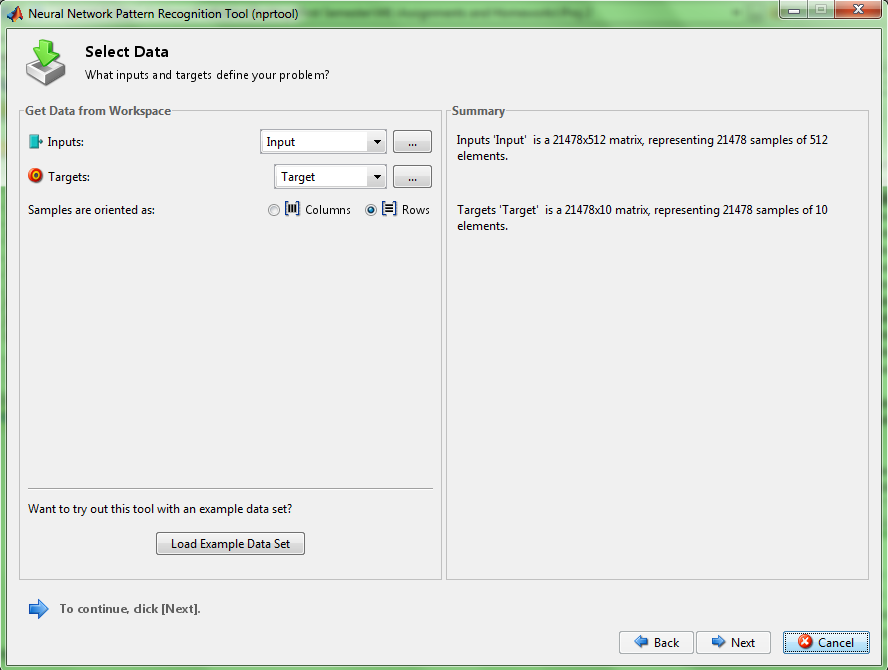
=====================================================================

**Performance on publicly available packages :**

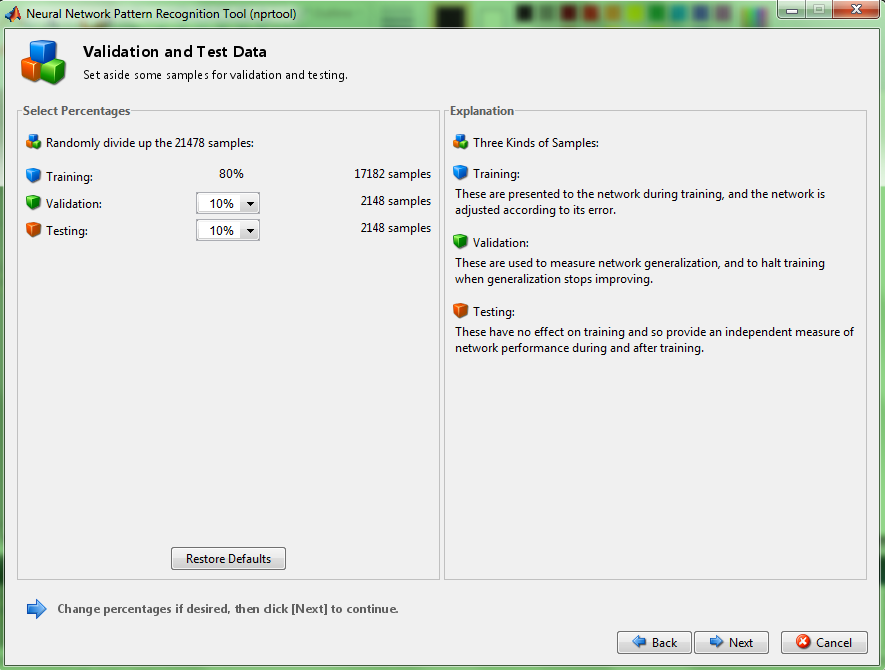
Matlab's neural network toolbox is used to train the data.

Combin the training and testing data into one single matrix.

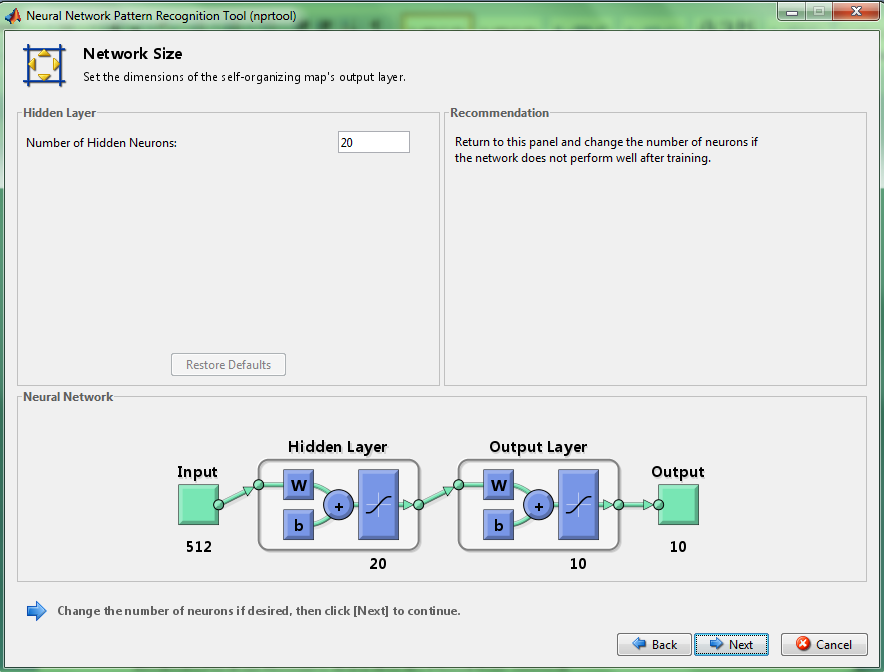
Select these matrices as input and target for the network model.

****

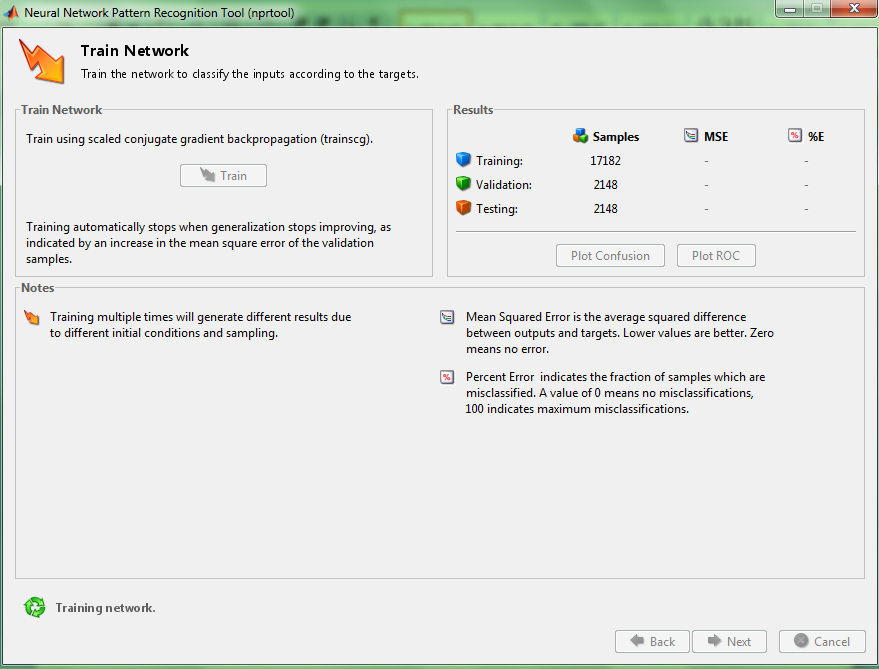
Partition the data as necessary -

****

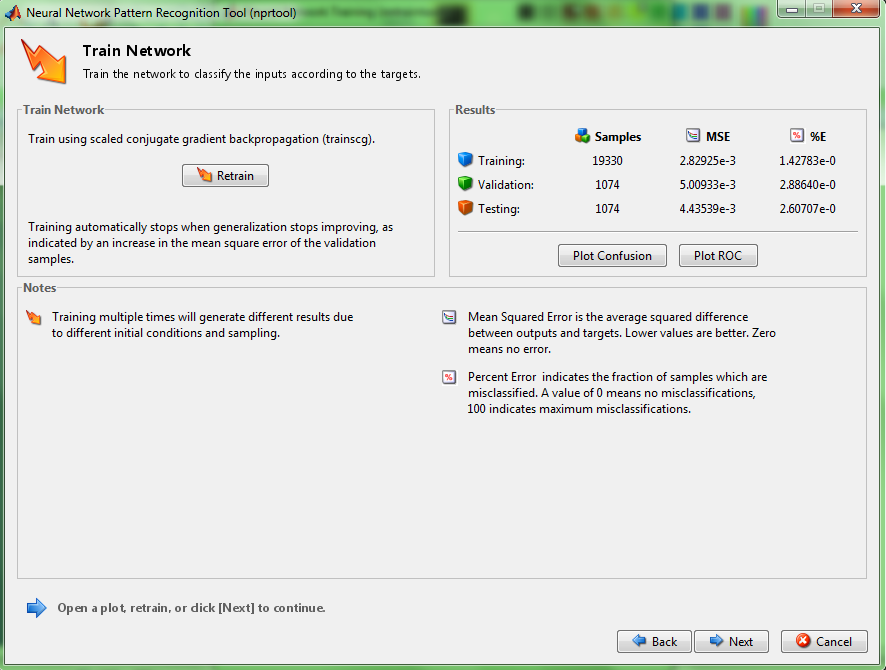
Select the number of units in the hidden layer -

****

Train the network -

****

The mean square error and the percentage error values for training, validating and testing data sets -



The percentage error using Logistic Regression is 2 %.

The percentage error using Neural Networks is 2.4 %.

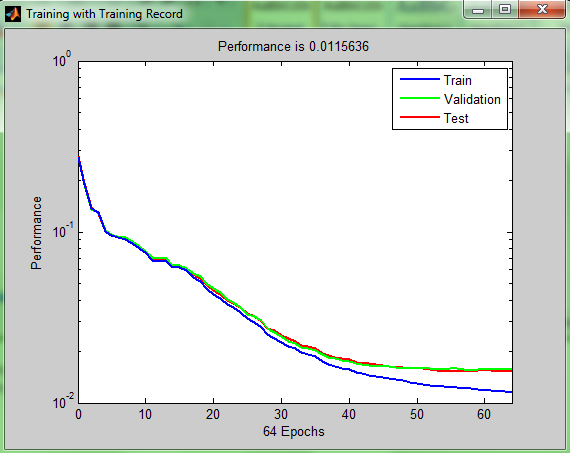


Fig : ROC plot

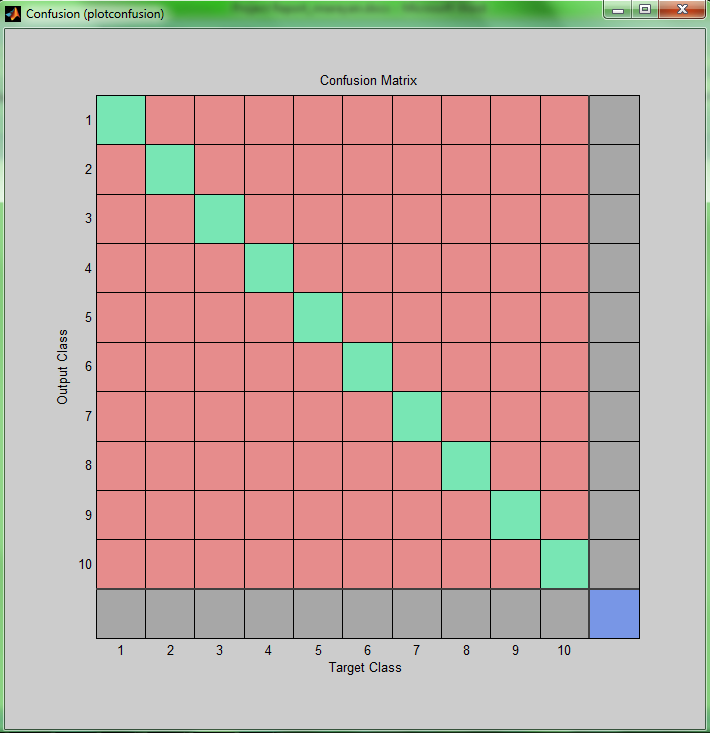


Fig : Confusion plot