CSE574: Introduction to Machine Learning Project 2: Classification of Handwritten Numerals using Logistic Regression and Neural Networks

By-

Aniket Deshpande (ad78)

50133684

Introduction:

The objective of the project was to recognize handwritten digits and predict the digit for some test data. This is a classical problem of Classification in Machine Learning. To accomplish this we have used two methods namely:

1. Multiclass Logistic Regression

The following formula was used:

MULTI CLASS LOGISTIC REGRESSION:

$$\mathrm{p}(C_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}.$$

where the 'activations' a_k are given by $a_k = w_k^T \phi$

2. Neural Networks

We have used the following formula:

$$y_k(x, w) = f\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)})\right)$$

Where f (.) is softmax function and h (.) we have taken as tanh function

For input data, I have used the provided GSC features extracted from each of the image.

Logistic Regression Approach:

Multiclass logistic is implemented using softmax function.

$$\mathrm{p}(C_k|\phi) = y_k\phi = \frac{\exp(a_k)}{\sum_j \exp(a_j)} \text{ where the 'activations' } a_k \text{ are given by } a_k = w_k^T\phi.$$

Here we will train the model for classification by minimizing the error. For error function we use the cross – entropy error function which is given as:

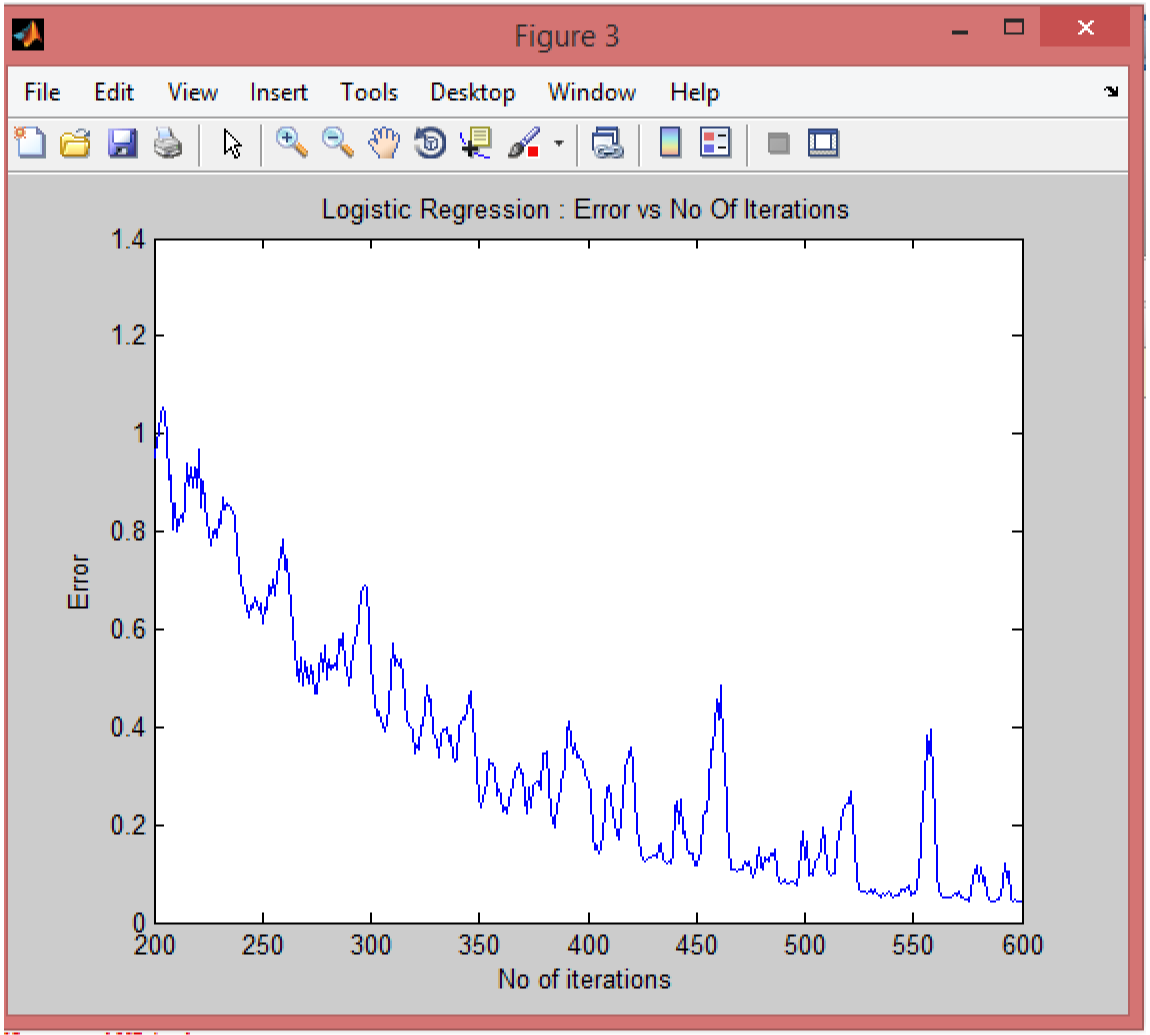
$$E(w_1 \dots w_k) = -\ln p(T|w_1 \dots w_k) = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

To accomplish this, we use gradient descent algorithm whose formula is given by:

$$w^{\tau+1} = w^{\tau} - \eta \nabla E_n$$

Following steps were taken to implement the above steps: [runfile_lr.m]

1. Initially, an input matrix X was created by reading the provided .txt files through matlab and appending all of them into a dingle variable X. Then bias parameter is incorporated in this variable. Thus the dimensions of X, in our case, become [(D + 1) x N] where D are the number of features in the provided .txt files and N is the total data for training. A Target matrix T was also created using 1-of-k scheme for our 10 classes. The dimensions of T are [K x N]. [load_train_data.m and load_test_data.m]
2. The weight vector is then initialized to all ones with dimensions [(D + 1) x K]. After this the parameters of “eta” and “no. of iterations” are set. We initially try for 800 iterations and eta value as 0.2. Following graph is generated for Error vs No of Iteration:



[train_lr.m]

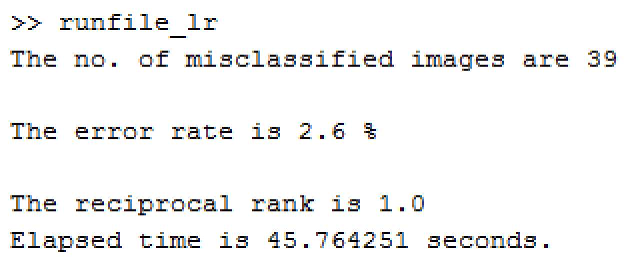
3. Then Y ie output vector was calculated using the above softmax function. The dimensions of Y would be [K x N]. To prevent underflow, a logsumexp() function was written. Then error gradient was calculated using following formula:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

   Then, the above mentioned gradient descent formula was used to update the weight vector [train_lr.m]

4. After 600 iterations, the labels were predicted using generated weight vector [test_lr.m]

The output after running the runfile_lr is as follows:



```
>> runfile_lr
The no. of misclassified images are 39

The error rate is 2.6 %

The reciprocal rank is 1.0
Elapsed time is 45.764251 seconds.
```

The predicted labels for the net test data are then saved in classes_lr.txt.

Neural Networks Approach:

The net Neural Network output function can be written as follows:

$$y_k(x, w) = f\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_{k0}^{(2)}\right)$$

We implement Neural Network with 1 Hidden Layer.

For the first Layer i.e. Input layer (h), we use the tanh function

For the output layer(f), softmax function is used as is the norm in multiclass regression.

The formula for hidden layer is as follows:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

$$z_j = h(a_j)$$

$$\text{i.e. } z_j = \sigma(a_j)$$

The quantities aj are called activations. Here h(.) is the tanh function we use on the activations to get values of the hidden units zj .

For the output layer, following formula is used

$$a_k = \sum_{i=1}^{D} w_{kj}^{(2)} z_j + w_{k0}^{(2)}$$

where k = 1 . . . , K and K is the number of outputs, i.e. 10 in our case.

The cross entropy error function used by us is:

$$E(w_1 \ldots w_k) = -\ln p(T|w_1 \ldots w_k) = -\sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \ln y_{nk}$$

Now to calculate error gradient at hidden layer and output layer, we use error backpropagation. The following formulas are used:

$$\delta_k = y_k - t_k$$

$$\delta_j = z_j(1 - z_j)\sum_{k=1}^{K} w_{kj}\delta_k$$

$$\frac{\partial E_n}{\partial w_{ji}^{(1)}} = \delta_j x_i$$

$$\frac{\partial E_n}{\partial w_{kj}^{(2)}} = \delta_k z_j$$

The error gradients are required to calculate the weight matrix at each layer according to following formula of gradient descent:

$$w^{(\tau+1)} = w^{(\tau)} - \eta \nabla E(w^{(\tau)})$$

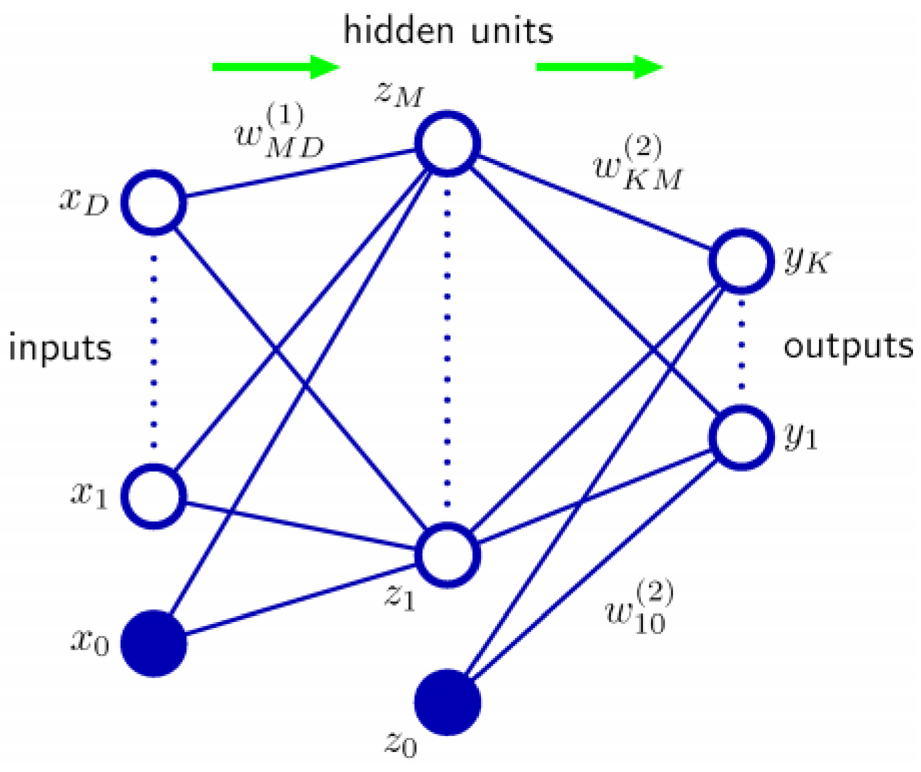
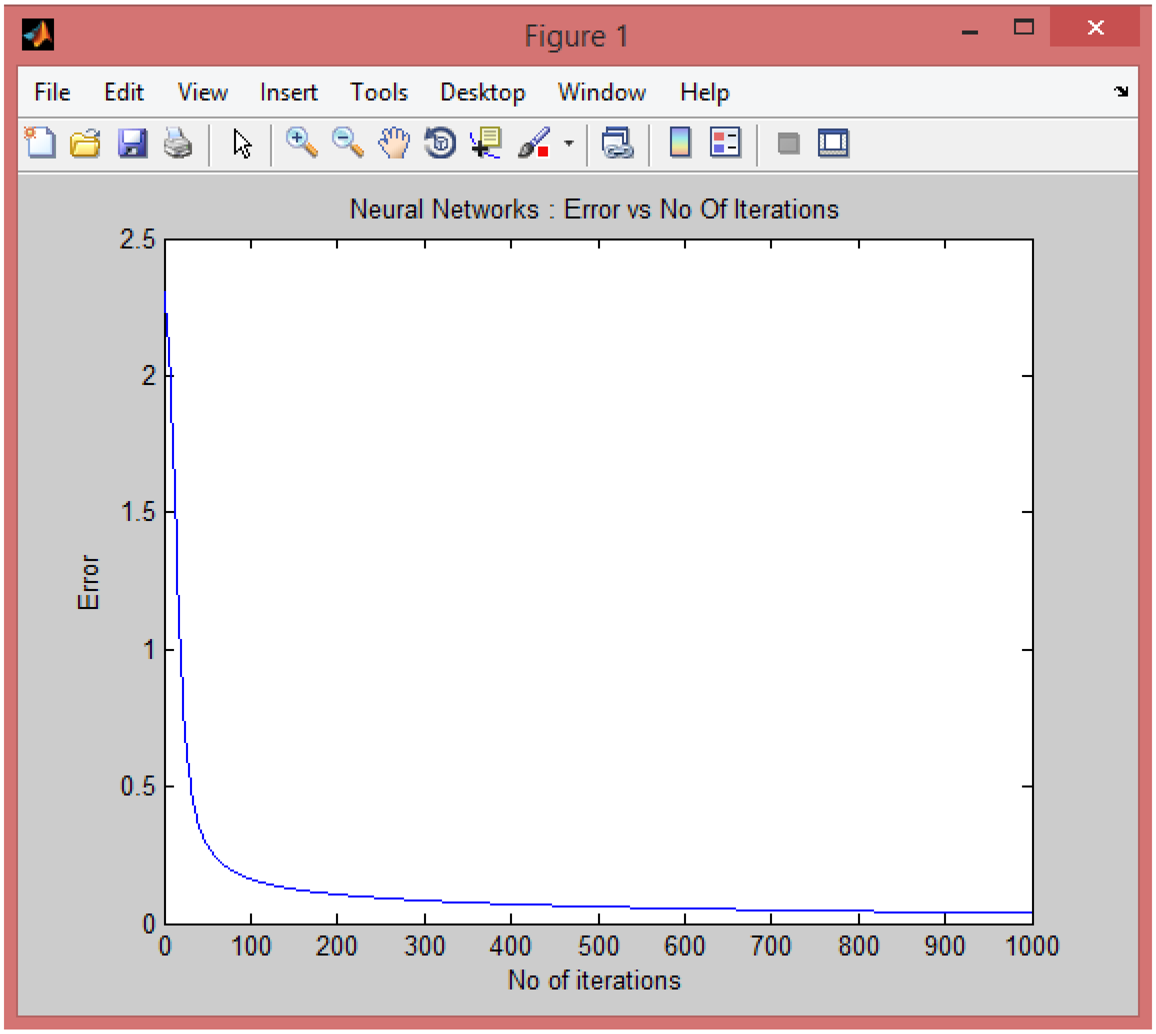


Figure 10: Neural Network

The model shown above is the standard neural network model that I have implemented in my code.

The steps taken to implement the above model are as follows: [runfile_nn.m]

1. Initially, an input matrix X was created by reading the provided .txt files through matlab and appending all of them into a dingle variable X. Then bias parameter is incorporated in this variable. Thus the dimensions of X, in our case, become [(D + 1) x N] where D are the number of features in the provided .txt files and N is the total data for training. A Target matrix T was also created using 1-of-k scheme for our 10 classes. The dimensions of T are [K x N]. [load_train_data.m and load_test_data.m]
2. The values for parameters M and learning rate are decided. In our case M is taken as 20 and learning rate as 0.00015. The weight matrices are then initialized to random values using rand() function. The dimensions of the weight matrices are: For hidden layer weight matric ie W1, the dimensions are [(D + 1) x M] and for output layer weight matrix ie W2, the dimensions are [M x K]. The activations, output, delta and updated weight matrices are then calculated using above mentioned formulas. This is done for 1000 iterations. The graph for error vs no of iterations as follows :[train_nn.m]



3. After these iterations the labels were predicted with the generated weight matrices.

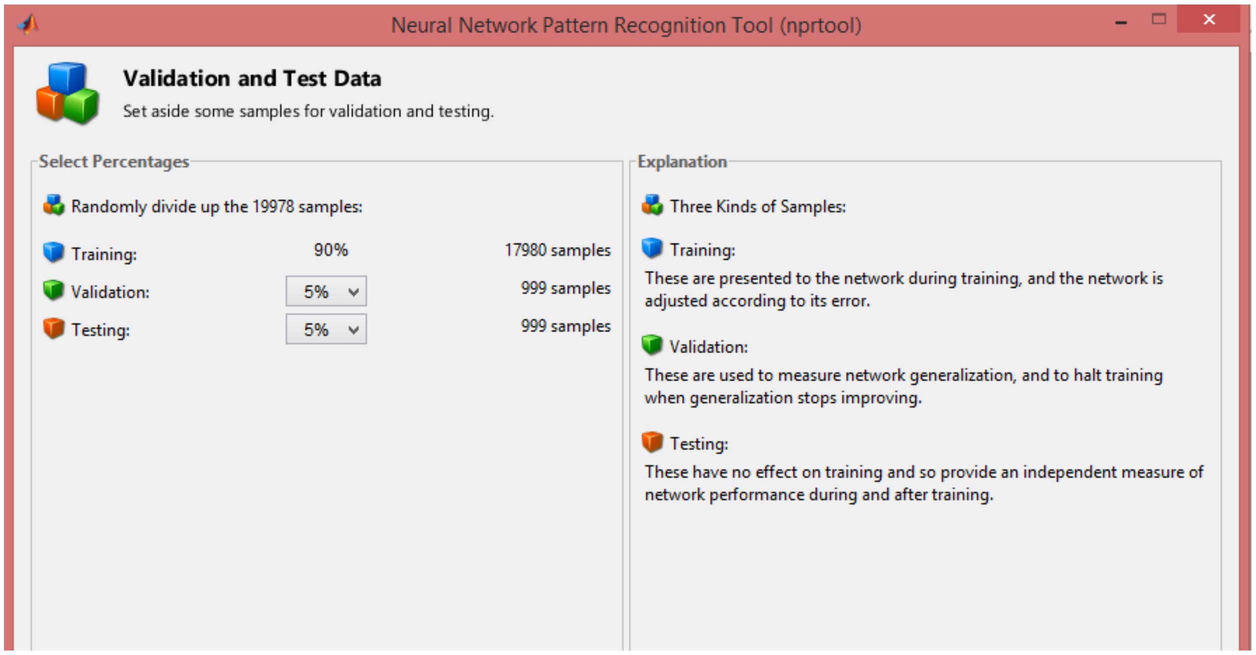
The output after running runfile_nn.m is as follows:

```
>> runfile_nn
The no. of misclassified images are 32

The error rate is 2.1 %

The reciprocal rank is 1.0
Elapsed time is 120.564364 seconds.
```

The predicted labels for the net test data are then saved in classes_nn.txt.

Comparison with Neural Network Package:

The default MATLAB neural network package is configured as follows:



Neural Network Pattern Recognition Tool (nprtool)

Validation and Test Data

Set aside some samples for validation and testing.

Select Percentages

Randomly divide up the 19978 samples:

Training:	90%	17980 samples
Validation:	5%	999 samples
Testing:	5%	999 samples

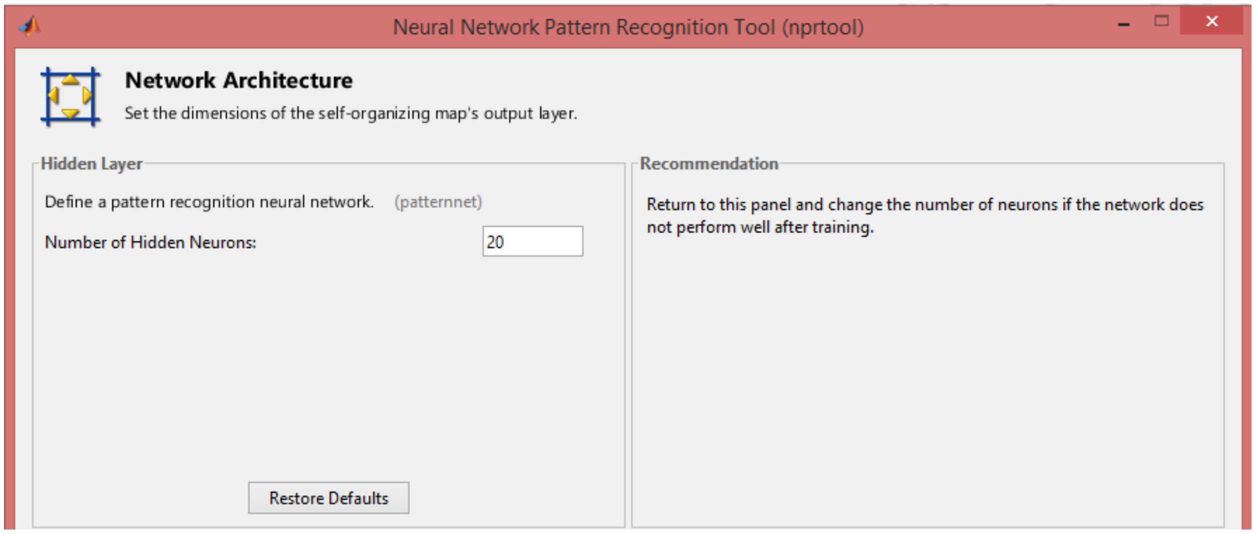
Explanation

Three Kinds of Samples:

Training:
These are presented to the network during training, and the network is adjusted according to its error.

Validation:
These are used to measure network generalization, and to halt training when generalization stops improving.

Testing:
These have no effect on training and so provide an independent measure of network performance during and after training.



Neural Network Pattern Recognition Tool (nprtool)

Network Architecture

Set the dimensions of the self-organizing map's output layer.

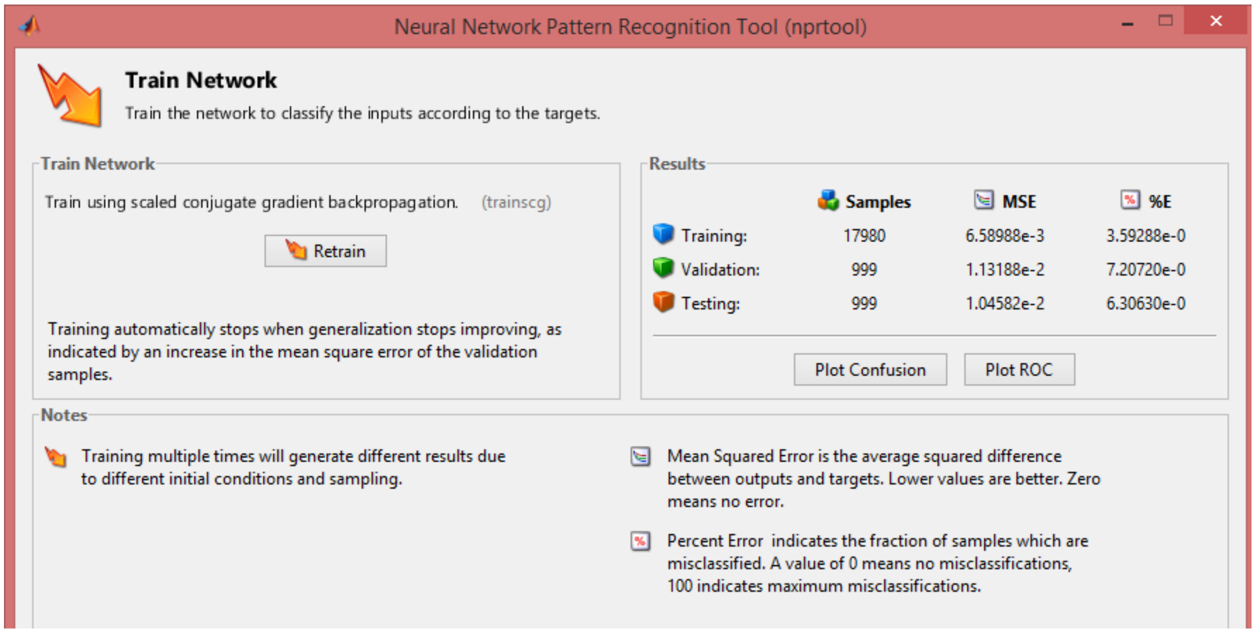
Hidden Layer

Define a pattern recognition neural network. (patternnet)

Number of Hidden Neurons: 20

Recommendation

Return to this panel and change the number of neurons if the network does not perform well after training.



Neural Network Pattern Recognition Tool (nprtool)

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:	17980	6.58988e-3	3.59288e-0
Validation:	999	1.13188e-2	7.20720e-0
Testing:	999	1.04582e-2	6.30630e-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.

The different error comparisons are as follows

Sr. No	Model	Error percentage(%)	Reciprocal rank
1	Logistic Regression	2.6	1
2.	Neural Network	2.1	1
3.	Neural Network Package	6.3	