

Report

Project 2: Classification

Name: Pratik Rajendra Kolhe

Student Number: 50098065

Multiclass Classification using Logistic Regression

Training Phase

Given Data

- Number of features for each image= 512
- Number of Feature Vectors used for training X= 19978
- Number of classes K= 10 (0 to 9)

Function used to predict the posterior probabilities for a vector belonging to a class

- Softmax function

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

j=1 to 10 for class labels 0-9 respectively

k belongs to j

$$a_k = \mathbf{w}_k^T \mathbf{x}$$

Error Function used

- Cross Entropy Error function

$$E(\mathbf{w}_1, \dots, \mathbf{w}_K) = -\ln p(\mathbf{T}|\mathbf{w}_1, \dots, \mathbf{w}_K) = -\sum_{n=1}^N \sum_{k=1}^K t_{nk} \ln y_{nk}$$

w=parameter vectors.

T=target vectors / ground truth

Formula used to update the parameters

- $\mathbf{w}_{\text{new}} = \mathbf{w}_{\text{old}} - \alpha * \text{Gradient of the error function}$

Method used to find the Gradient

- Batch Gradient Decent

Parameters to decide

- alpha - learning rate
- number of iterations for Gradient Decent

Choosing alpha (learning rate)

The value of learning rate alpha is important because for very small of alpha its takes time to reach the global minima and for higher value of alpha there is a possibility that global minima is missed, as the gradient of the error is convex function.

Keeping number of **iterations=500**, the cross entropy error for different values of learning rate was examined.

The following results were observed:

alpha (learning rate)	cross entropy error
0.01	NaN (not a number)/ Infinity
0.001	0.0222
0.0001	0.0435
0.00001	0.1857
0.000001	0.8361

From the above table it can be observed that the minimum cross entropy error was for **alpha=0.001**, Further decrease in alpha increased the cross entropy error also further decrease in alpha increased the crossed entropy error.

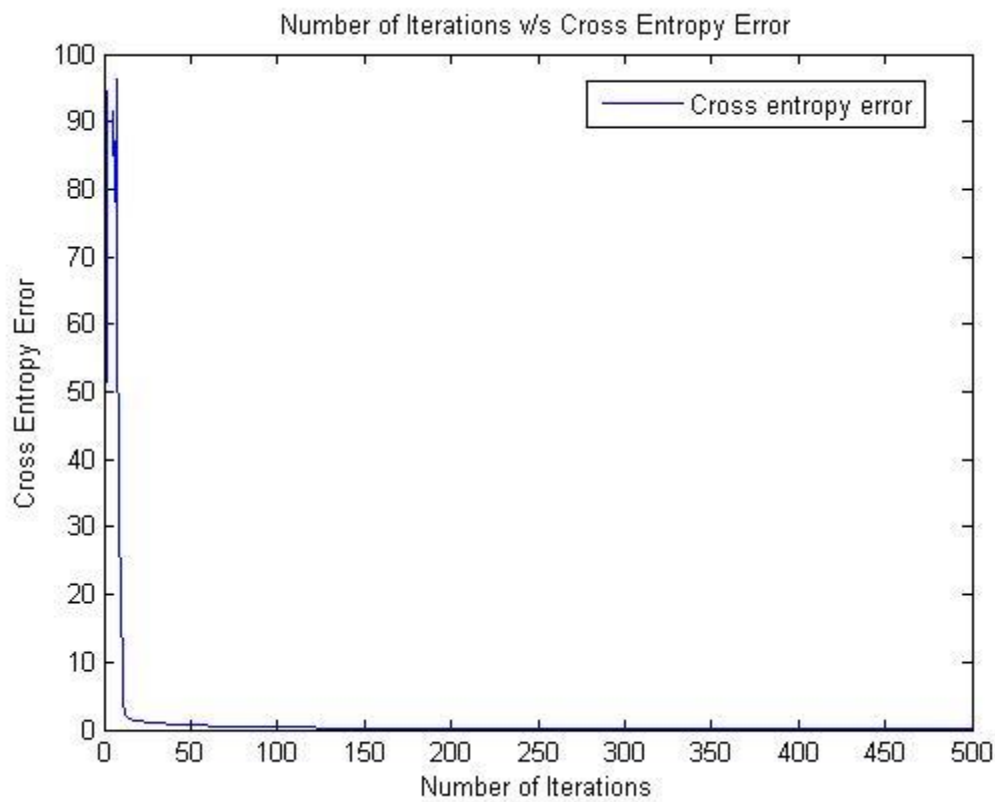
The right amount of value for learning rate **alpha** also helped in finding the minimum value of gradient in few iterations.

Deciding number of iterations for Gradient Decent

Deciding the number of iterations for Gradient Decent was important. This is because after certain number of iterations the cross entropy error doesn't vary much, this is when we decide that the gradient has converged.

Keeping **alpha=0.01** (selected from above analysis), the graph for cross entropy error and number of iterations was plotted.

Following were the observations:



It can be observed that after 125 iterations the gradient value converged.

Hence **total number of iterations=125** was chosen for the Gradient Decent algorithm.

The cross entropy error after 125 iterations= 0.2585

Testing Phase

After training the model and learning the parameters, the model was tested to classify 1500 feature vectors in 10 classes one for each digit.

Following were the results observed:

Class Label	Correct Predictions out of 150
0	150
1	147
2	145
3	146
4	150
5	145
6	148
7	140
8	139
9	144

Correctly identified labels= 1454/1500

Error rate= 46/1500 =0.0307

Cross Entropy Error for test=0.0502

***Note:** The results slightly vary each time the model is trained. The reason is the parameters are initially randomly assigned and may vary each time the model is trained. However there is no significant difference in the results.

Multiclass Classification using Neural Networks

Network Details

Input Layer= 513 (Number of features + Bias)

Output Layer=10 (Number of classes 0-9)

Activation function used for Hidden Layers= Sigmoid

Activation function used for Output Layer= Softmax

Choosing the number of neurons in the Hidden Layers and defining network architecture

The number of neurons in the hidden layers were chosen to be **350** $\approx (\text{Input Layer} + \text{Output Layer}) * 2/3$

Hidden Layer-1- 175 neurons

Hidden Layer-2- 175 neurons

Forward Propagation

Following are the implementation details of forward propagation:

Input:

X_train= 19978 X 513 Matrix

Parameters:

w1=513 x 175 matrix

w2=175 x 175 matrix

w3=175 x 10 matrix

a1=x_train;

z2=a1*w1;

a2=sigmoid(z2);

z3=a2*w2;

a3=sigmoid(z3,);

z4=a3*w3;

y_train=softmax(z4)

Output

y_train=19978 x 10 Matrix

Back-Propagation

Implementation details of Back Propagation.

Calculating delta at each node

```
d4=y_train-t_train      .... {t_train=19978 x 10 matrix (Ground Truth)}
d3=d4*w3T.*(a3).*(1-a3)
d2=d3*w2T.*(a2).*(1-a2)
```

Parameter Updation

```
w3=w3-(alpha*(a3T*d4))
w2=w2-(alpha*(a2T*d3))
w1=w1-(alpha*(a1T*d2))
```

where alpha = learning rate

Choosing alpha (Learning Rate)

Keeping **number of iterations =100** for back propagation the network was trained to learn the parameters using different values of alpha.

Following were the observed results:

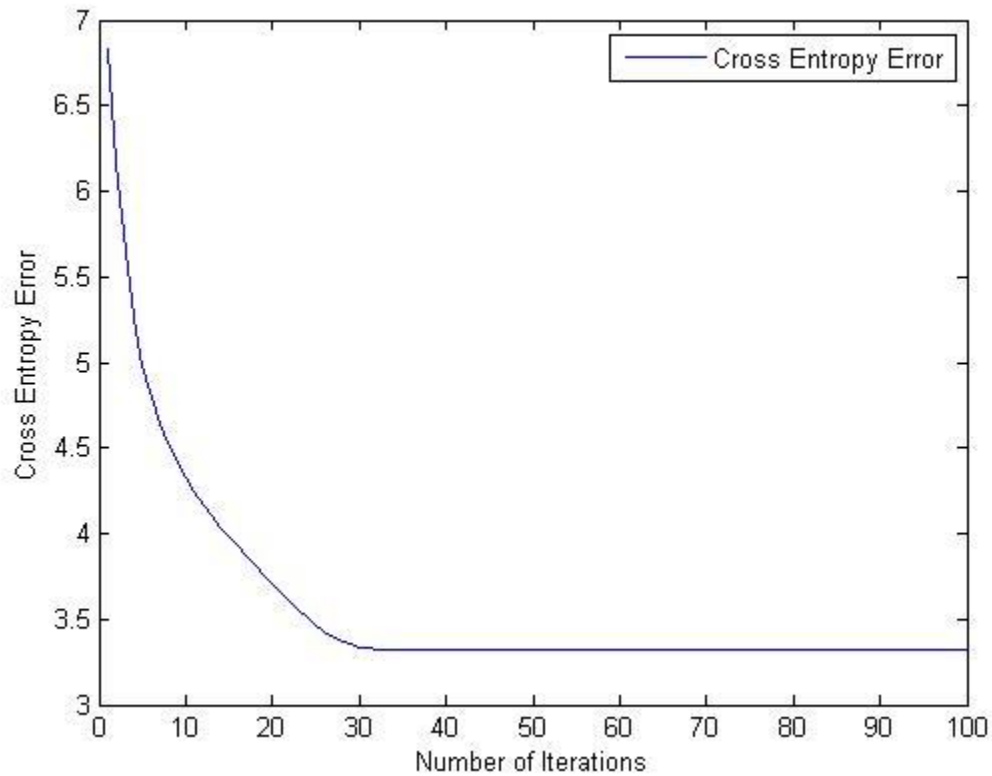
alpha(learning rate)	cross entropy error
0.001	NaN(not a number)/Infinity
0.00001	36.672
0.000001	3.3219
0.0000001	3.3219
0.00000001	21.3837

Hence **alpha=0.000001** was chosen as a learning rate.

Choosing Number of Iterations to find the gradient

Keeping **alpha=0.000001** (from the above analysis) the network was trained for 100 iterations to find the gradient.

Following were the results observed:



It can be observed that after **35 iterations** the gradient of cross entropy error converged.

Hence **total 35 iterations** were used to calculate the gradient.

Testing phase

Following were the results observed:

Class Label	Correct Predictions out of 150
0	0
1	0
2	0
3	0
4	0
5	150
6	0
7	0
8	0
9	0

Correctly identified labels= 150/1500

Error rate= 1350/1500 =0.9

It can be observed that the implemented neural network model over-fitted and gave poor results.

Multiclass Classification using NN-tool

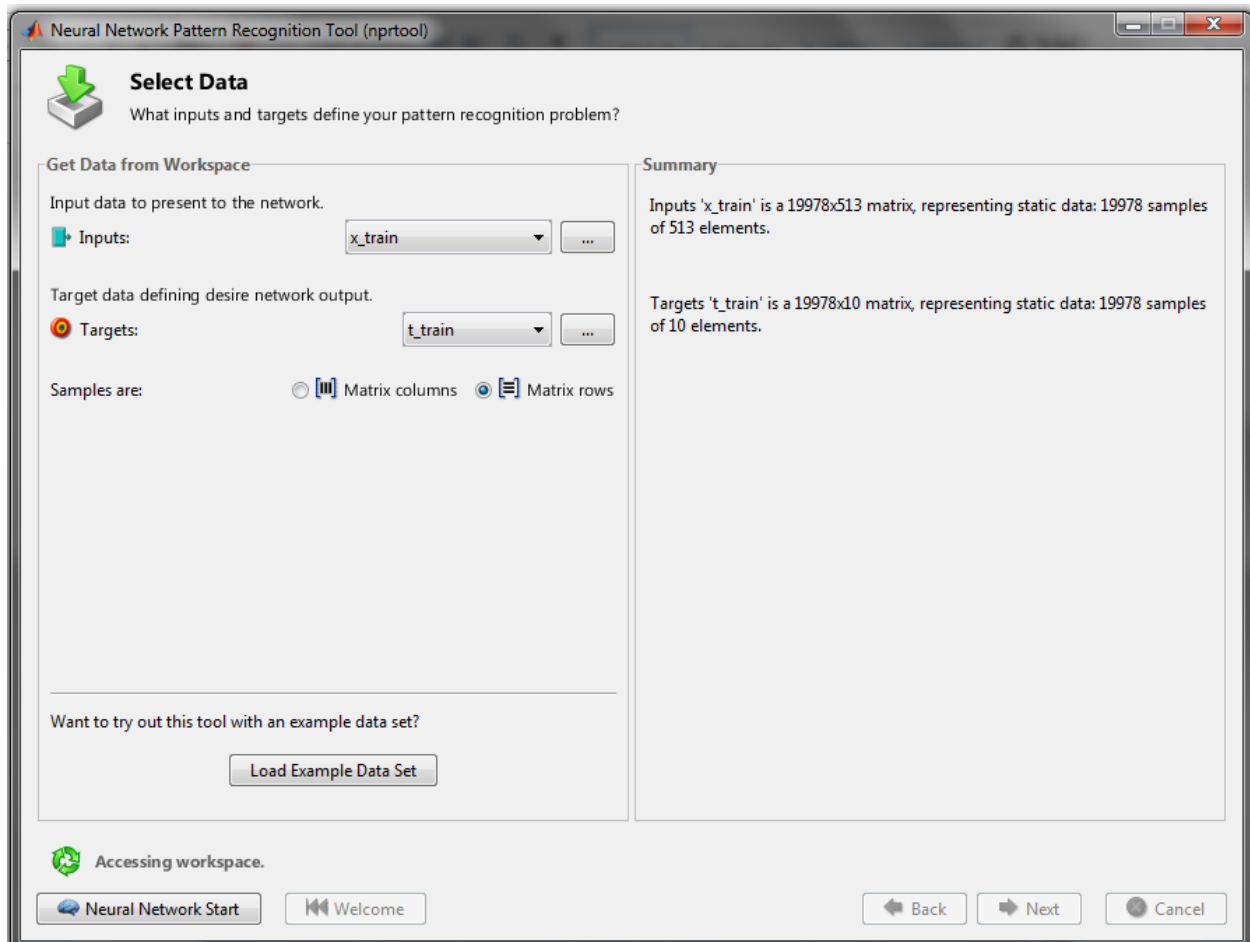
Classification was also carried out using NN-tool of Matlab.

Training Phase

Following is the setup:

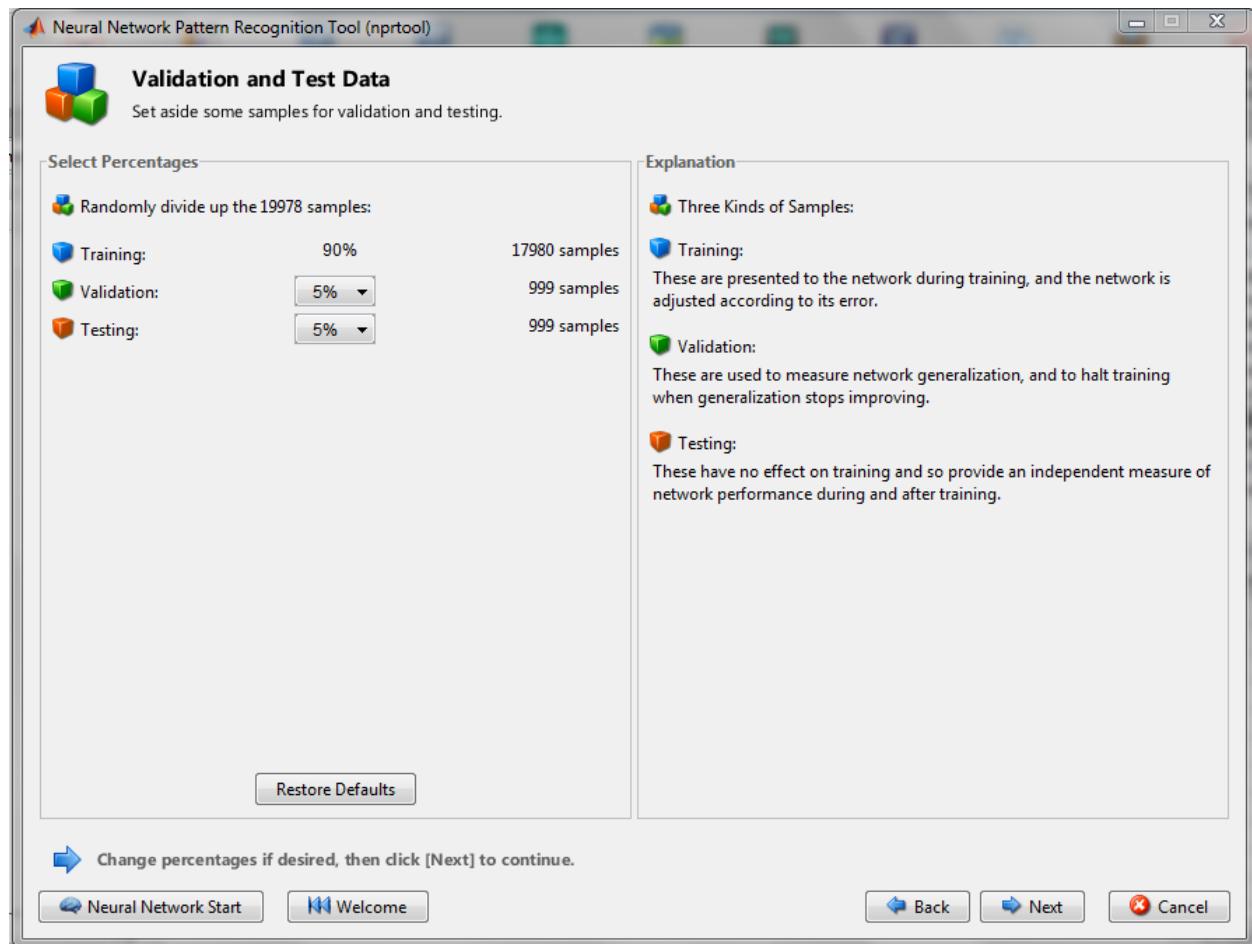
Input= 19978 feature vectors.

Target= matrix of ground truth for all 19978 feature vectors.



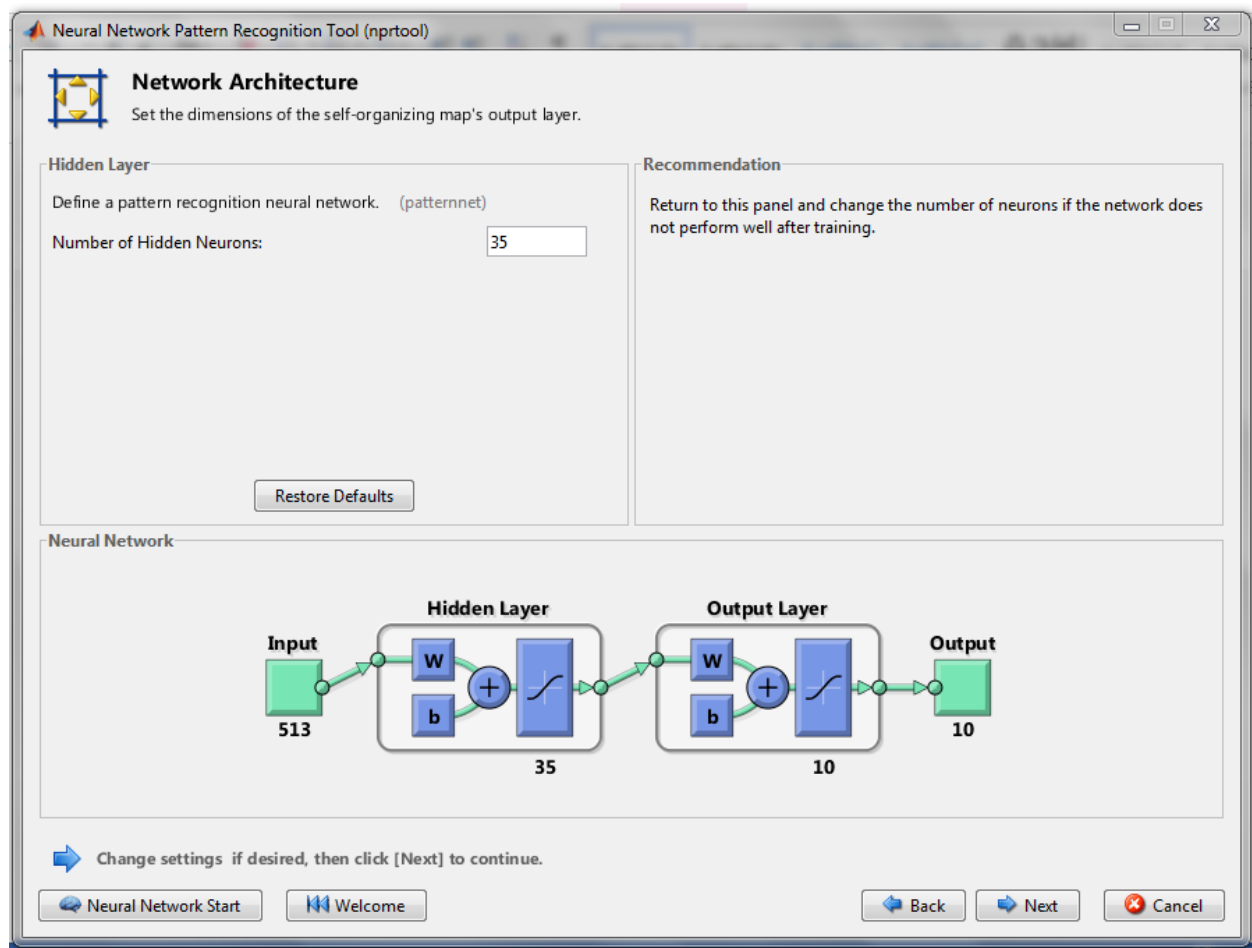
19978 samples were divided as follows:

- Training- 90%
- Validation- 5%
- Testing- 5%

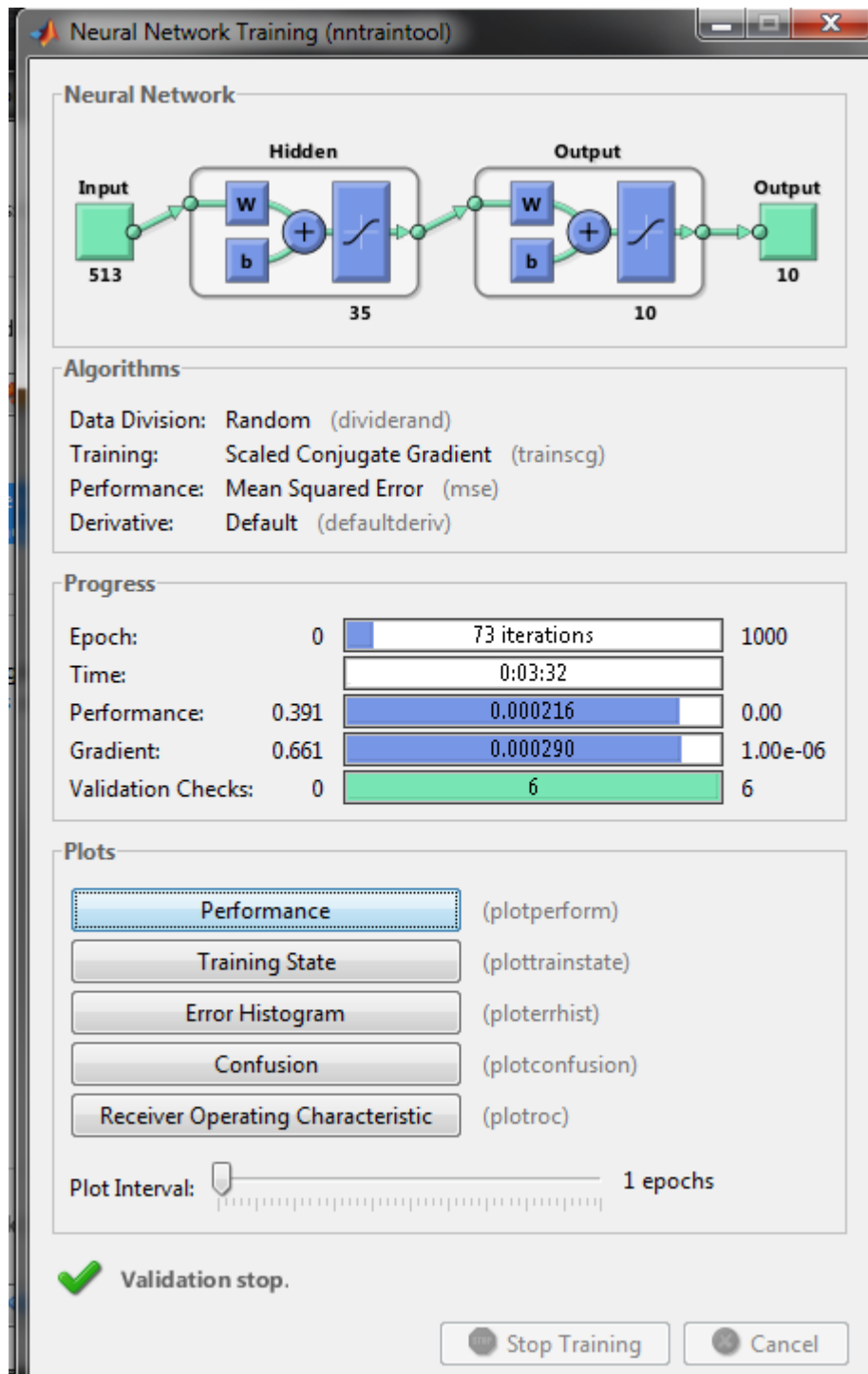


Total Number of Neurons = 35







Activation Function for each neuron = Sigmoid



- The Data Division was random.
- Scaled Conjugate Gradient decent is used for training.
It has the advantage of being faster for supervised learning.
- Gradient value of 0.000290 was reached after 73 iterations.



Following were the results after training:



Results			
	 Samples	 MSE	 %E
 Training:	17980	2.95082e-4	1.61290e-1
 Validation:	999	2.06575e-3	1.00100e-0
 Testing:	999	2.98289e-3	2.30230e-0



Testing Phase



In testing phase 1500 feature vectors belonging to 0-9 digits were classified using the above trained model.

Following were the observations:

Optionally perform additional tests


 Inputs: 


 Targets: 


Samples are: ☐  Matrix columns ☒  Matrix rows

Inputs 'x_test' is a 1500x513 matrix, representing static data: 1500 samples of 513 elements.

Targets 't_test' is a 1500x10 matrix, representing static data: 1500 samples of 10 elements.

 Test Network

 MSE 8.60468e-3

 %E 4.93333e-0

Confusion Matrix										
Output Class	1	2	3	4	5	6	7	8	9	10
	147 9.8%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	2 0.1%	0 0.0%	5 0.3%	0 0.0%
	0 0.0%	147 9.8%	3 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	1 0.1%
	1 0.1%	1 0.1%	142 9.5%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	9 0.6%	1 0.1%
	0 0.0%	0 0.0%	1 0.1%	147 9.8%	0 0.0%	3 0.2%	0 0.0%	1 0.1%	0 0.0%	1 0.1%
	0 0.0%	0 0.0%	0 0.0%	0 0.0%	149 9.9%	0 0.0%	0 0.0%	3 0.2%	0 0.0%	12 0.8%
	0 0.0%	0 0.0%	0 0.0%	2 0.1%	0 0.0%	146 9.7%	0 0.0%	1 0.1%	8 0.5%	0 0.0%
	2 0.1%	1 0.1%	2 0.1%	0 0.0%	1 0.1%	0 0.0%	148 9.9%	0 0.0%	1 0.1%	1 0.1%
	0 0.0%	0 0.0%	1 0.1%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	144 9.6%	0 0.0%	5 0.3%
	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	127 8.5%	0 0.0%
	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	129 8.6%
98.0% 98.0% 94.7% 98.0% 99.3% 97.3% 98.7% 96.0% 94.7% 96.0% 95.1% 2.0% 2.0% 5.3% 2.0% 0.7% 2.7% 1.3% 4.0% 5.3% 4.0% 4.9%										
Target Class										

Below are the observations from the Confusion matrix:

Class Label	Correct Predictions out of 150
0	147
1	147
2	142
3	147
4	149
5	146
6	148
7	144
8	127
9	129

Correctly identified labels= 1426/1500

Error rate= 74/1500 =0.0493

Comparison of models

	Error Rate
Logistic Regression	0.0307
Implemented Neural Network	0.9
Neural Network using NN-Tool	0.0493

Due to over-fitting the neural network model implemented performed poorly than other models.

The best results for classification were observed by Logistic Regression Model.