Aniket Amar Thopte

Electric Load Forecasting using different techniques
Artificial Neural Networks

Introduction:

The problem statement deals with hourly load forecasting. We have been given dataset from 2002 to 2005 (i.e. 4 years data) on hourly basis. I have used **2-layer feed forward artificial neural network** technique for forecasting the load for 2006 year on hourly basis.

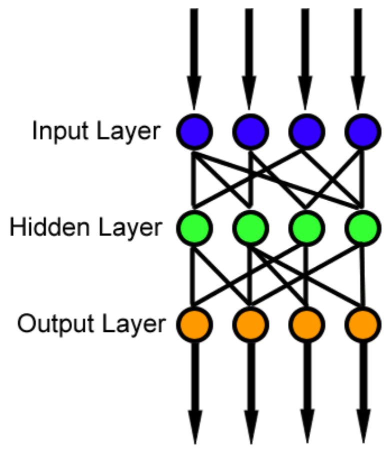
Dataset:

The Dataset file is named **'history.csv'**. The dataset provided has 4 variable columns namely 'Date' (from 1-Jan-2002 to 31st Dec 2005), 'Hour' ranging from 1 to 24, 'Temperature' & 'Load'. The total **number of observations given is 35064**. Another excel file containing the 2006 data is provided named '**fcst.csv**' in which load column is kept blank to be forecasted.

Methodology:

The ANN was built in **MATLAB** using '**nntool**' feature. The various steps used in the process are elaborated further.

A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle. As such, it is different from its descendant: recurrent neural networks. The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network.



Source: Wikipedia

In our case the target variable is Load and the input/feature variable are Hour of day, Day of week, Month, Temperature.

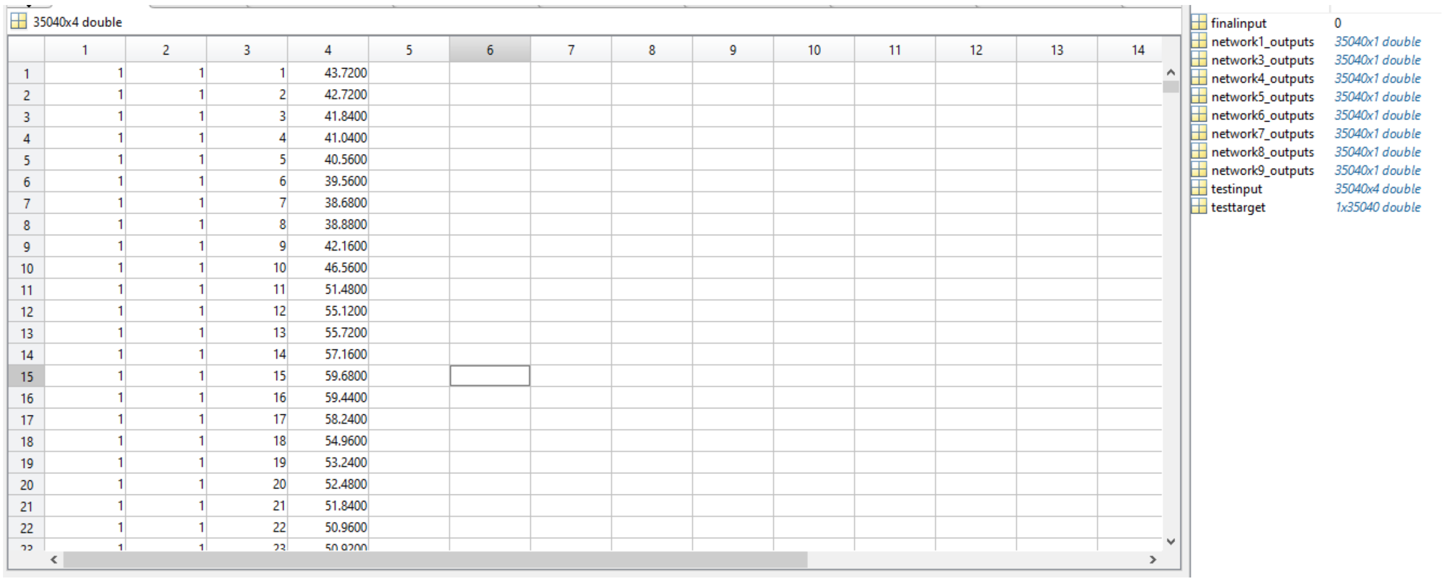
Data-Preprocessing:

The first and foremost step taken is to delete the 24 observations for the 29th February timestamp in the leap year 2004. This is done as these values for the date 29th Feb are seen only once throughout the four years.

We have converted the date column into two different feature columns named day of week and month.

MATLAB work and step-by-step process.

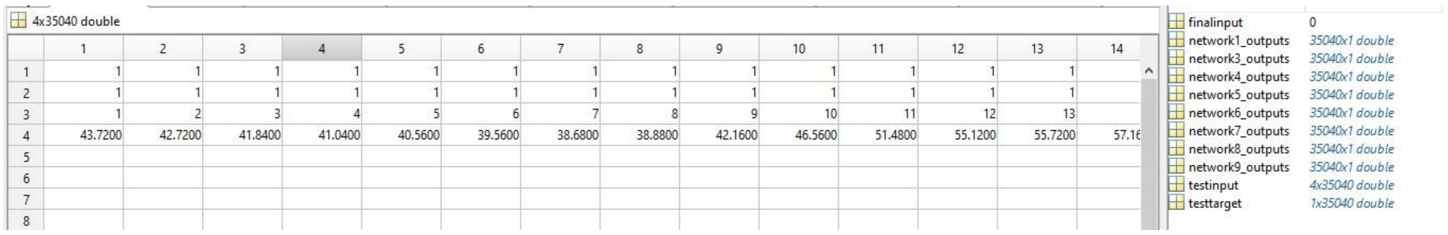
Initially we create 3 files in the workspace window named ‘testinput’, ‘testtarget’ and ‘finalinput’ to store the data from our History and Fcst files for working.


35040x4 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	43.7200										
2	1	1	2	42.7200										
3	1	1	3	41.8400										
4	1	1	4	41.0400										
5	1	1	5	40.5600										
6	1	1	6	39.5600										
7	1	1	7	38.6800										
8	1	1	8	38.8800										
9	1	1	9	42.1600										
10	1	1	10	46.5600										
11	1	1	11	51.4800										
12	1	1	12	55.1200										
13	1	1	13	55.7200										
14	1	1	14	57.1600										
15	1	1	15	59.6800										
16	1	1	16	59.4400										
17	1	1	17	58.2400										
18	1	1	18	54.9600										
19	1	1	19	53.2400										
20	1	1	20	52.4800										
21	1	1	21	51.8400										
22	1	1	22	50.9600										
23	1	1	23	50.9200										

finalinput	0
network1_outputs	35040x1 double
network3_outputs	35040x1 double
network4_outputs	35040x1 double
network5_outputs	35040x1 double
network6_outputs	35040x1 double
network7_outputs	35040x1 double
network8_outputs	35040x1 double
network9_outputs	35040x1 double
testinput	35040x4 double
testtarget	1x35040 double

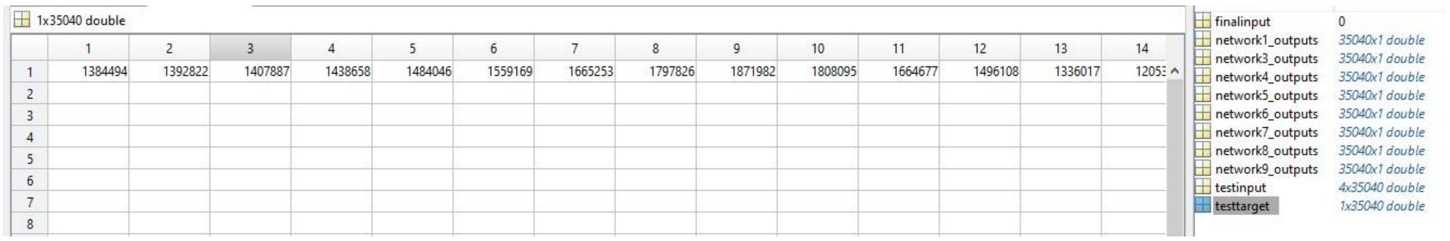
We do a transpose function on the matrix we created as MATLAB cannot process the data as it is.


4x35040 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	1	1	1	1	1	1	1	1	1	1	1	1	
2	1	1	1	1	1	1	1	1	1	1	1	1	1	
3	1	2	3	4	5	6	7	8	9	10	11	12	13	
4	43.7200	42.7200	41.8400	41.0400	40.5600	39.5600	38.6800	38.8800	42.1600	46.5600	51.4800	55.1200	55.7200	57.16
5														
6														
7														
8														

finalinput	0
network1_outputs	35040x1 double
network3_outputs	35040x1 double
network4_outputs	35040x1 double
network5_outputs	35040x1 double
network6_outputs	35040x1 double
network7_outputs	35040x1 double
network8_outputs	35040x1 double
network9_outputs	35040x1 double
testinput	4x35040 double
testtarget	1x35040 double

testtarget data input:-

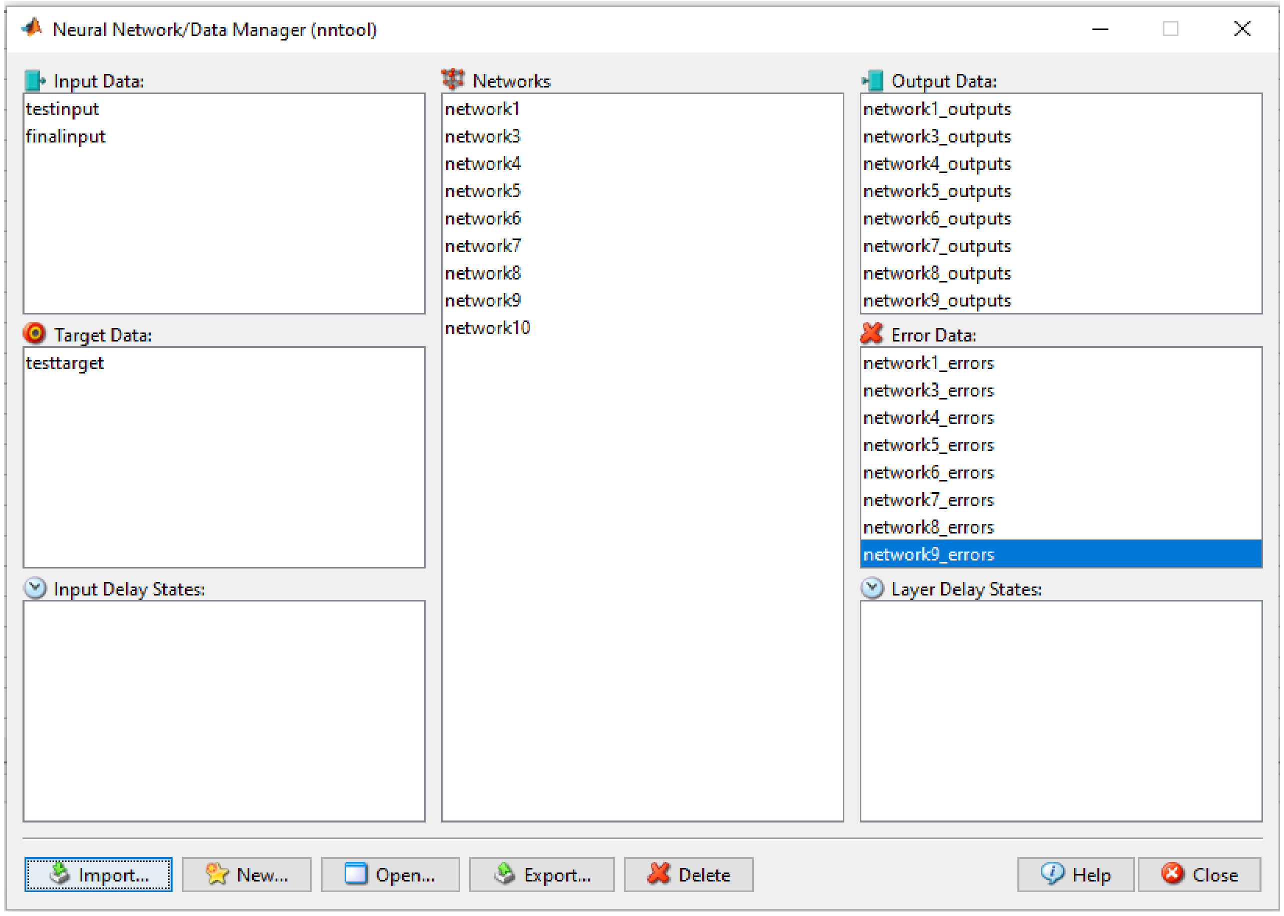

1x35040 double

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1384494	1392822	1407887	1438658	1484046	1559169	1665253	1797826	1871982	1808095	1664677	1496108	1336017	12053
2														
3														
4														
5														
6														
7														
8														

finalinput	0
network1_outputs	35040x1 double
network3_outputs	35040x1 double
network4_outputs	35040x1 double
network5_outputs	35040x1 double
network6_outputs	35040x1 double
network7_outputs	35040x1 double
network8_outputs	35040x1 double
network9_outputs	35040x1 double
testinput	4x35040 double
testtarget	1x35040 double

After that we call ‘***nntool’*** function from the command window of MATLAB.

nntool:- Open Network/Data Manager. Neural network fitting tool. nntool leads you through solving a data fitting problem, solving it with a two-layer feed-forward network trained with Levenberg-Marquardt.


Neural Network/Data Manager (nntool)

Input Data:
testinput
finalinput

Target Data:
testtarget

Input Delay States:

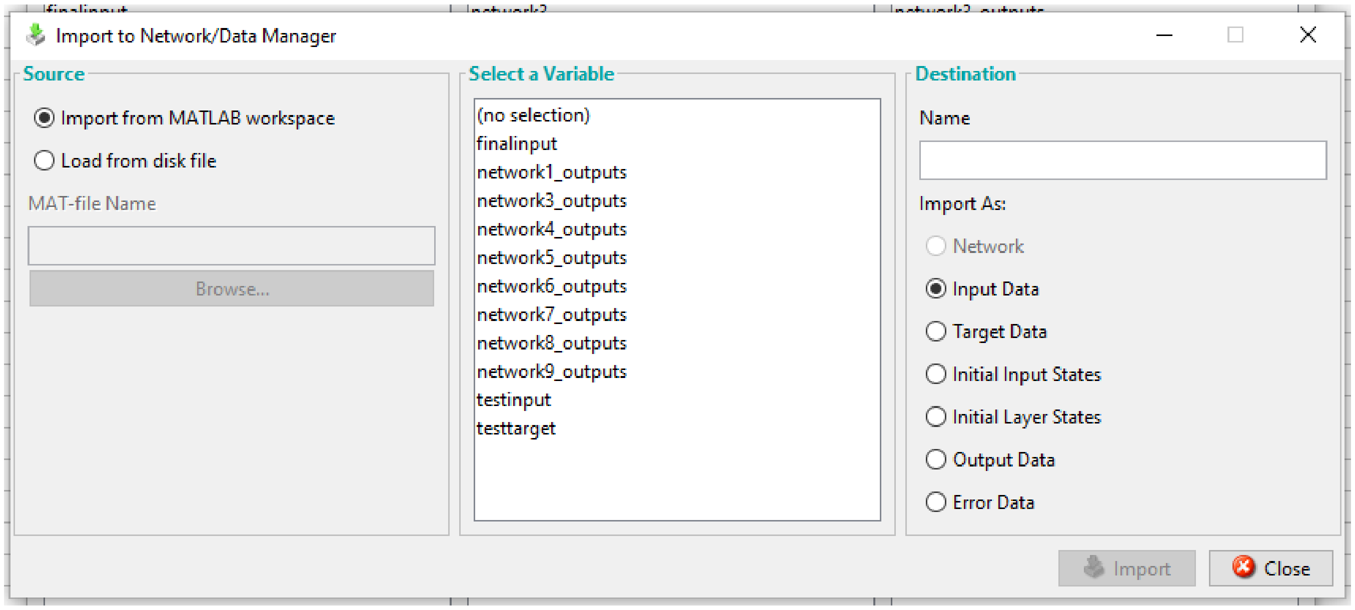
Networks
network1
network3
network4
network5
network6
network7
network8
network9
network10

Output Data:
network1_outputs
network3_outputs
network4_outputs
network5_outputs
network6_outputs
network7_outputs
network8_outputs
network9_outputs

Error Data:
network1_errors
network3_errors
network4_errors
network5_errors
network6_errors
network7_errors
network8_errors
network9_errors

Layer Delay States:

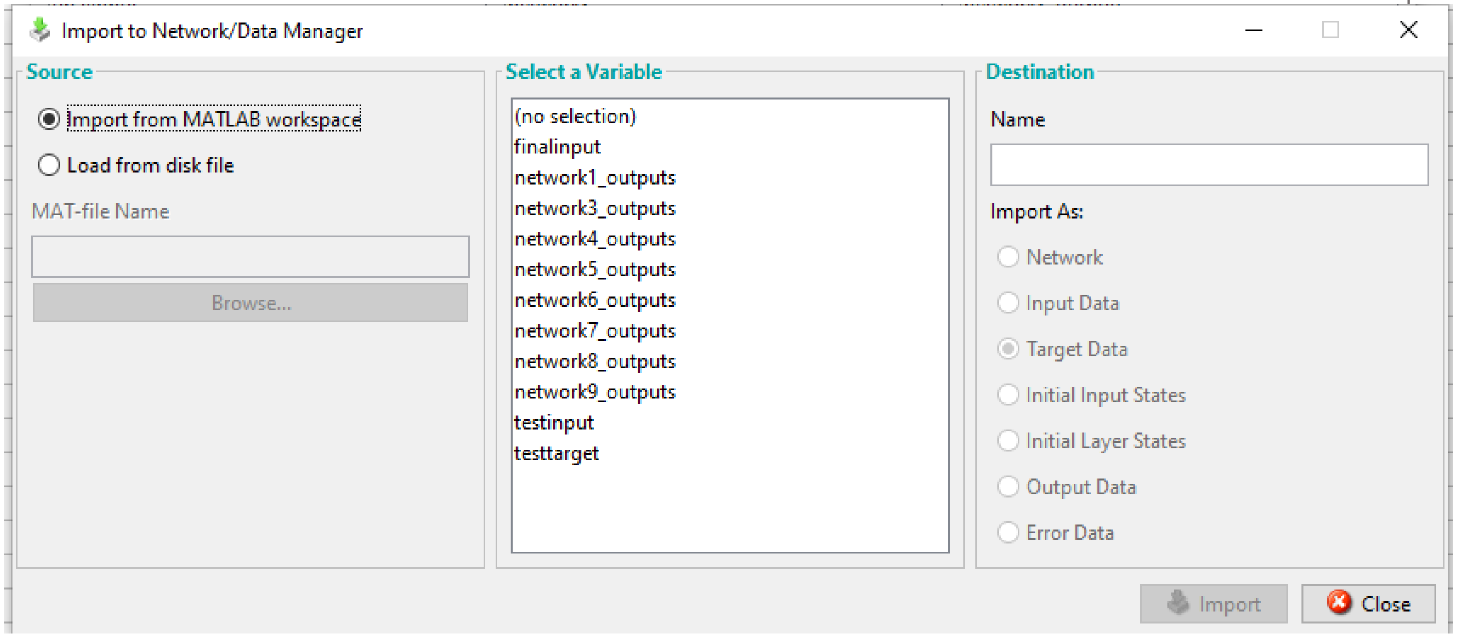
We then input data using import.


Import to Network/Data Manager

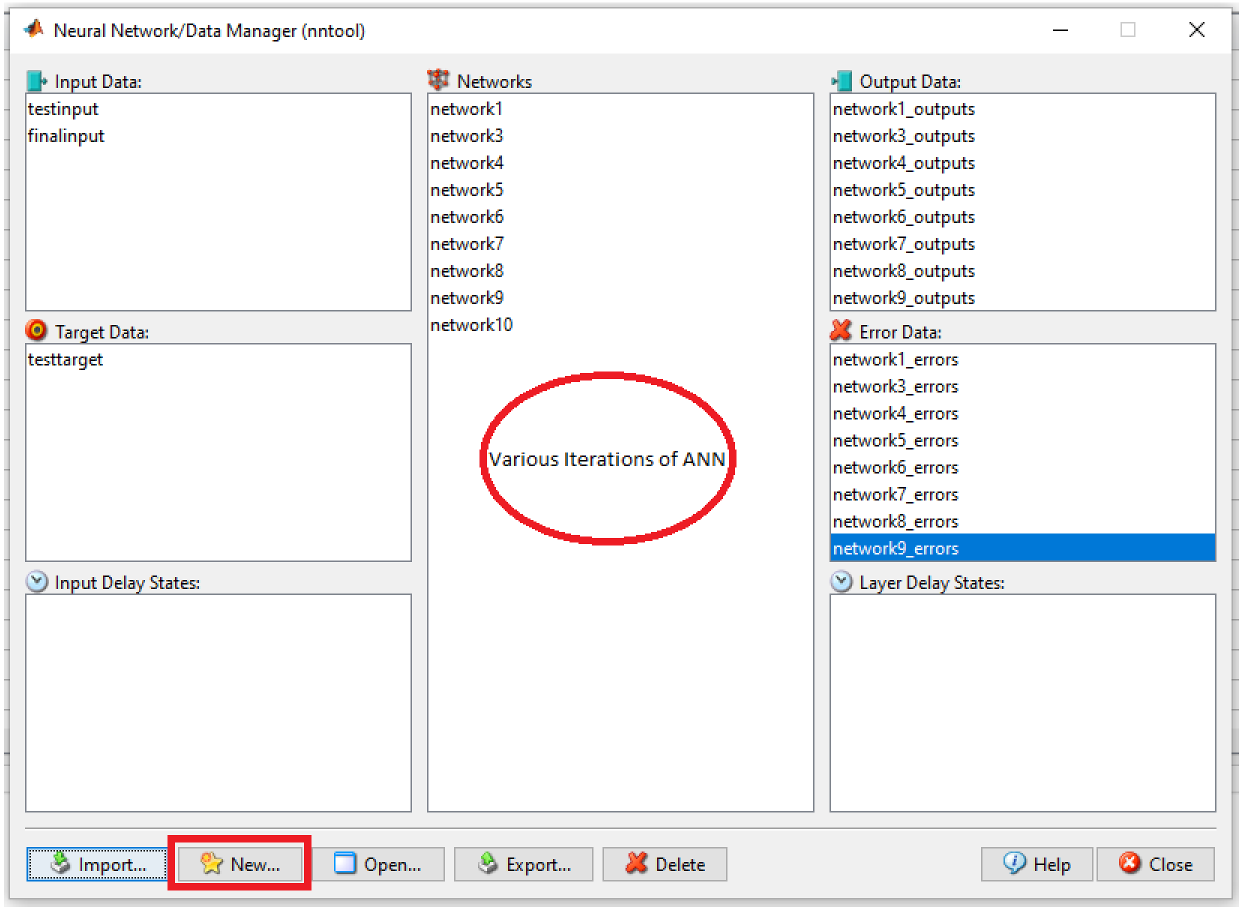
Source
[✓ Import from MATLAB workspace]
Load from disk file
MAT-file Name

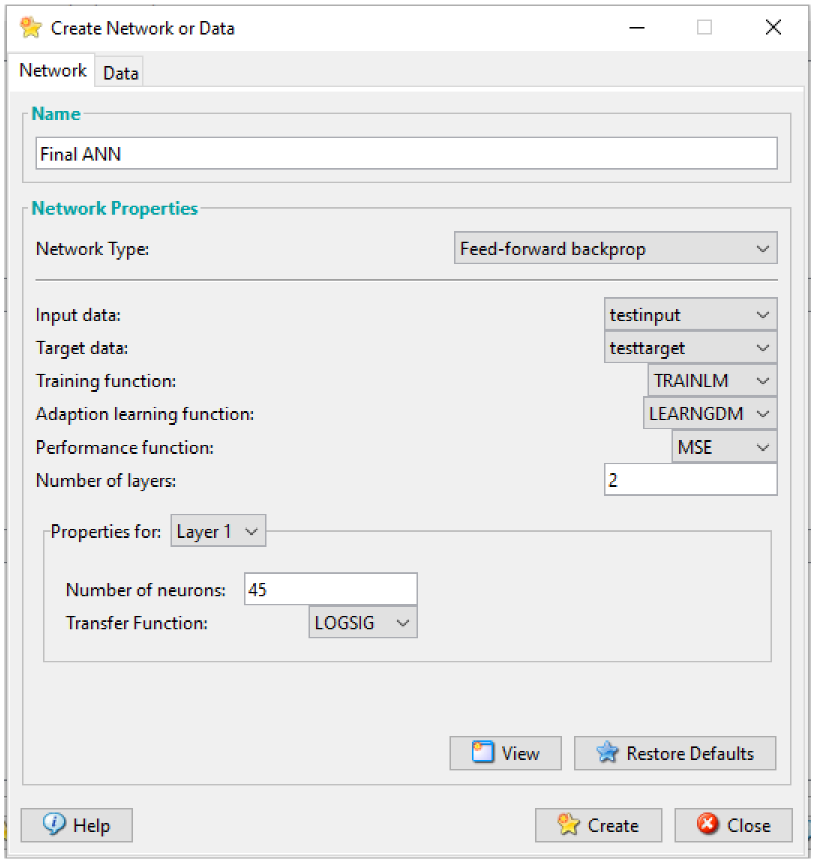
Select a Variable
(no selection)
finalinput
network1_outputs
network3_outputs
network4_outputs
network5_outputs
network6_outputs
network7_outputs
network8_outputs
network9_outputs
testinput
testtarget

Destination
Name
Import As:
Network
[✓ Input Data]
Target Data
Initial Input States
Initial Layer States
Output Data
Error Data



After that next step is to build the Neural Network. Click new to open the ANN window.





Name: Final ANN

Network Properties

Network Type:	Feed-forward backprop
Input data:	testinput
Target data:	testtarget
Training function:	TRAINLM
Adaption learning function:	LEARNGDM
Performance function:	MSE
Number of layers:	2

Properties for: Layer 1

Number of neurons:	45
Transfer Function:	LOGSIG

Network Name: FinalANN

Network Type:

The Backpropagation algorithm is a supervised learning method for multilayer feed-forward networks from the field of Artificial Neural Networks. Feed-forward neural networks are inspired by the information processing of one or more neural cells, called a neuron. The principle of the backpropagation approach is to model a given function by modifying internal weightings of input signals to produce an expected output signal. The system is trained using a supervised learning method, where the error between the system's output and a known expected output is presented to the system and used to modify its internal state.

Training Function: Levenberg-Marquardt Back-Propagation Algorithm.

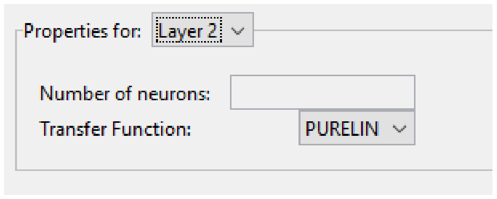
Learning Function: 'learngdm' is the gradient descent with momentum weight and bias learning function.

Performance Function: Mean Square Error.

Number of Layers: 2

Layer 1: Number of neurons = 45, Activation function = 'logsig'.

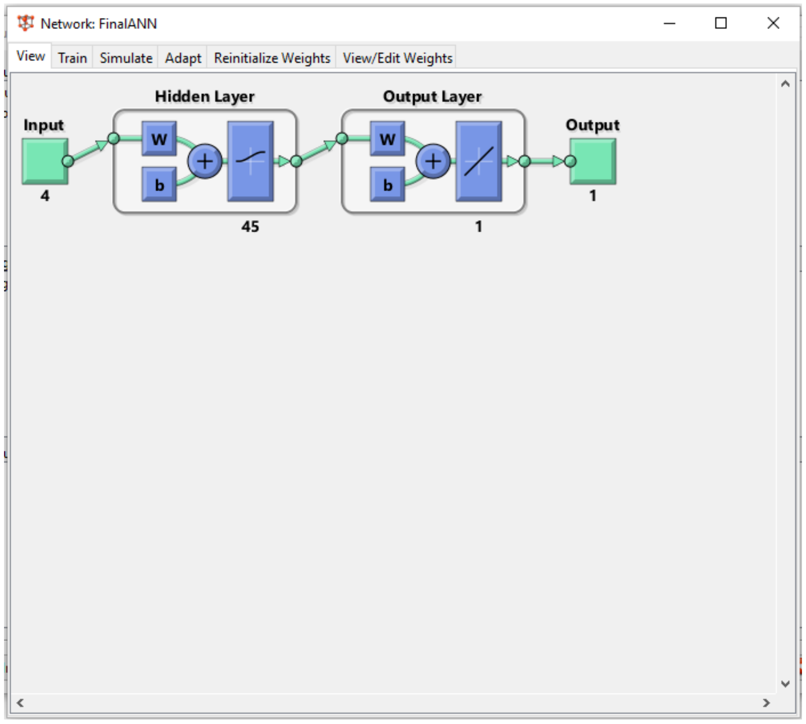
Layer 2: Number of neurons = 1, Activation function = 'purelin'



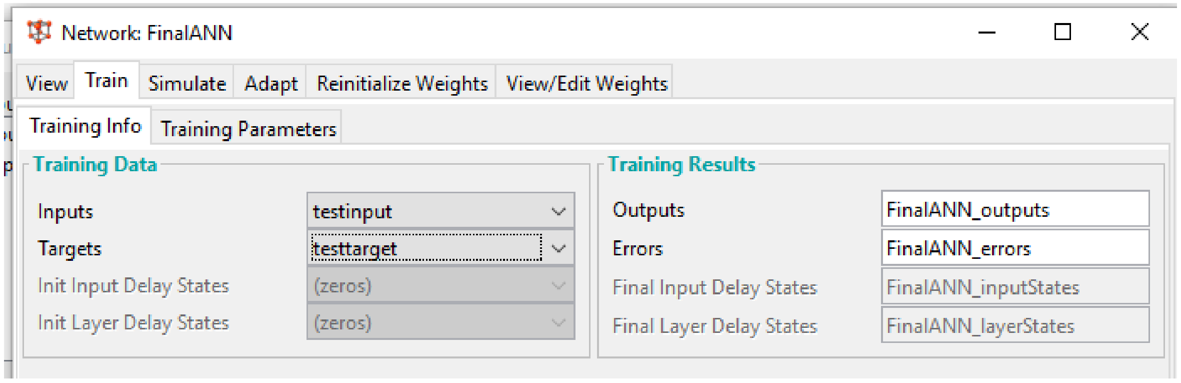
Properties for: Layer 2

Number of neurons:	
Transfer Function:	PURELIN

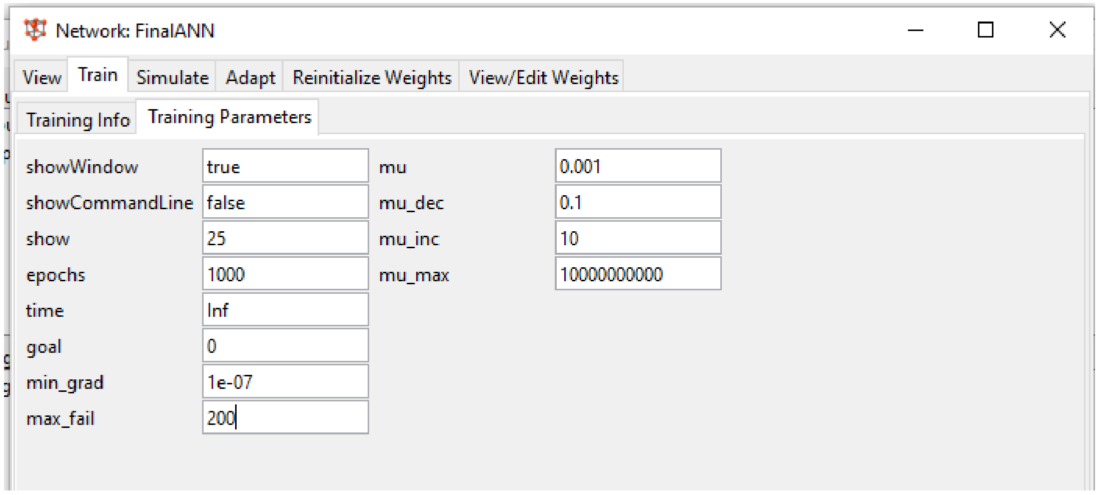
Then we train and test the model.



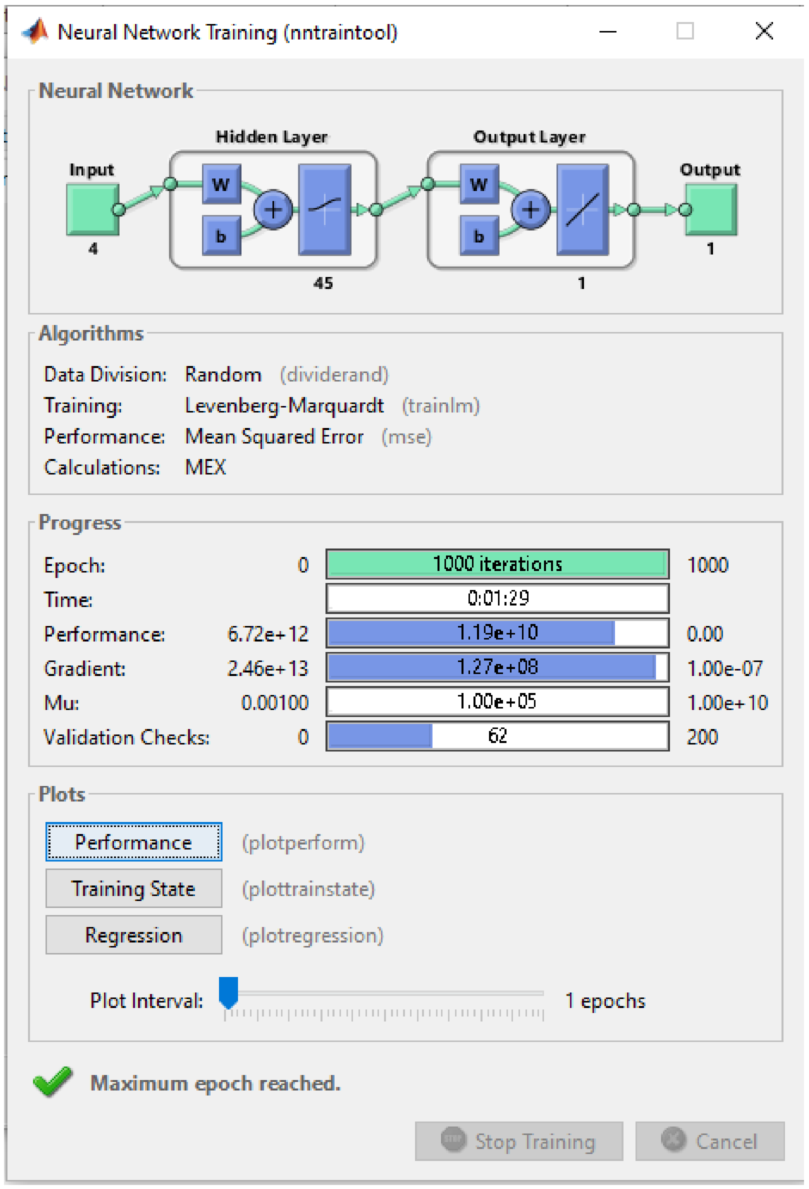
Click on train and select the test input and target values. Change the validation check number to 200 and epoch number to 1000 (iterations). Rest all is default.



Inputs	testinput	Outputs	FinalANN_outputs
Targets	testtarget	Errors	FinalANN_errors
Init Input Delay States	(zeros)	Final Input Delay States	FinalANN_inputStates
Init Layer Delay States	(zeros)	Final Layer Delay States	FinalANN_layerStates



showWindow	true	mu	0.001
showCommandLine	false	mu_dec	0.1
show	25	mu_inc	10
epochs	1000	mu_max	10000000000
time	Inf		
goal	0		
min_grad	1e-07		
max_fail	200		


Neural Network Training (nntraintool)
Neural Network
Hidden Layer
Output Layer
Input
Output
W
b
W
b
4
45
1
1

Algorithms

Data Division: Random (dividerand)
Training: Levenberg-Marquardt (trainlm)
Performance: Mean Squared Error (mse)
Calculations: MEX

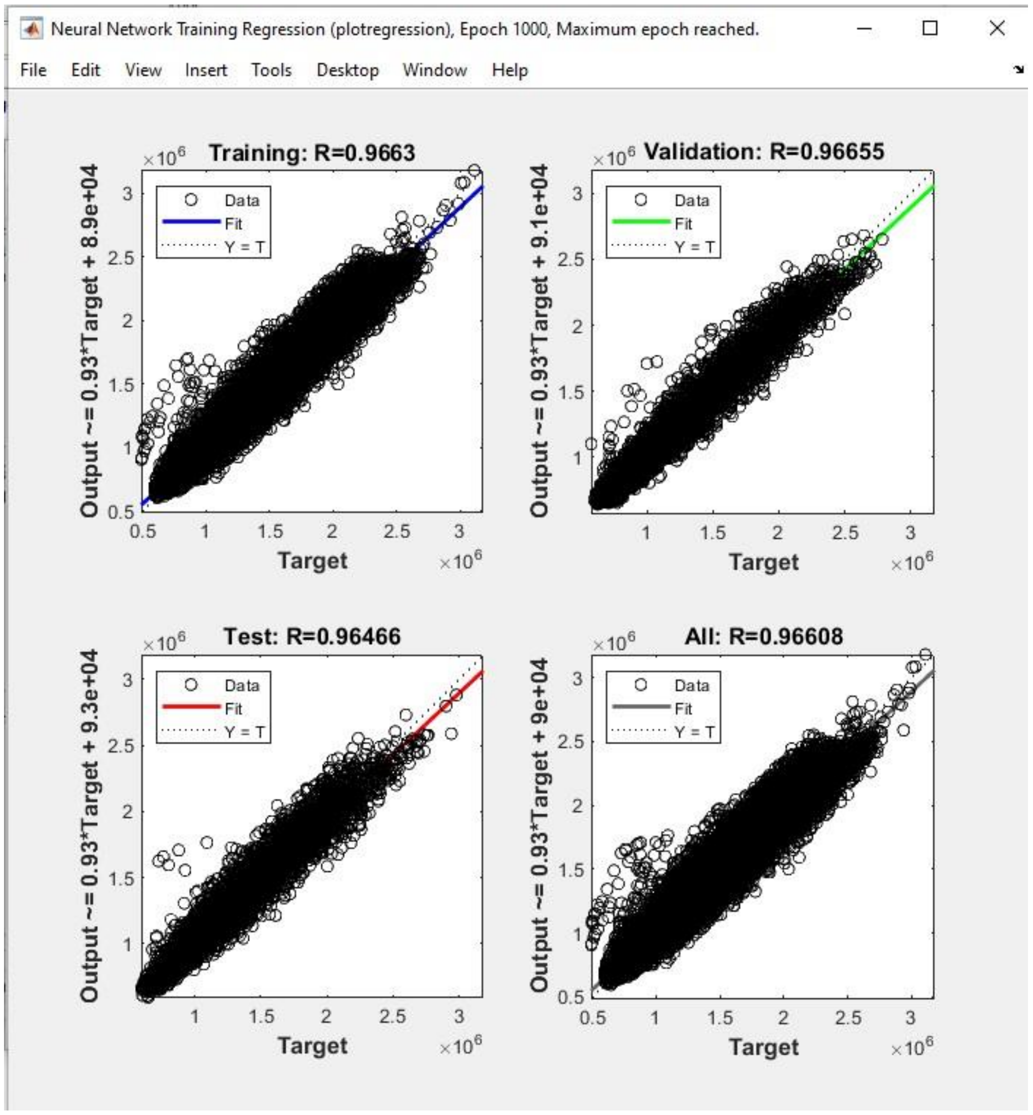
Progress

Epoch:	0	1000 iterations	1000
Time:		0:01:29	
Performance:	6.72e+12	1.19e+10	0.00
Gradient:	2.46e+13	1.27e+08	1.00e-07
Mu:	0.00100	1.00e+05	1.00e+10
Validation Checks:	0	62	200

Plots
Performance (plotperform)
Training State (plottrainstate)
Regression (plotregression)
Plot Interval: 1 epochs

Maximum epoch reached.

The regression plot shows the R value around 0.96… for training, validation, test and all. This shows our model is satisfactory and acceptable.



Final Step: Forecasting

We now do the forecasting for the year 2006 (8760 Values on hourly basis) using the simulate function in the network window from where we did the train command. We select the data from year 2006 stored inside the 'finalinput' file we earlier created and start the forecasting. The output file is exported to the workspace and finally to the Fcst.csv file. Hence, we have used the ANN model for forecasting year ahead hourly based load. Our model is working.

Conclusion:

We have tried different iterations (Neural Nets) in the form of network 1 to 10 having different number of neurons to get the lowest ***MAPE*** for our model which comes to be ***10.62%.***