Neural Network and Fuzzy Systems (EECE 5860)

Project-2 Report

Submitted by – Mohammad Saber
(MUID: 005852837)

CONTENTS

PROBLEM ONE: DEVELOP SUPERVISED ANN (WITH 2-2-1) ARCHITECTURE TO SOLVE THE WELL-KNOWN XOR PROBLEM

A) DEVELOP AN ANN WITH THE SIGMOID OUTPUT PE:

In MATLAB, I have typed 'nntool' to open the neural network toolbox GUI which looks like figure1.

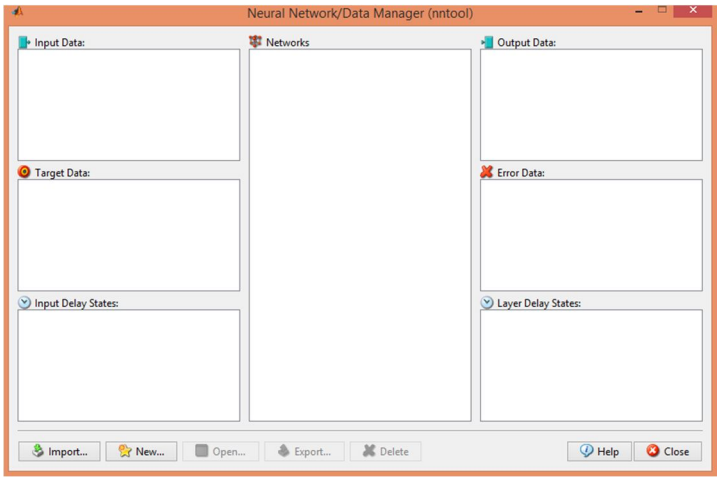

Figure1: Neural Network Toolbox GUI in MATLAB

Now we have entered the truth table of XOR as input and target data in neural network toolbox. If we click on 'New' button and then select the 'Data' tab, we will see the option for entering input and target data. After entering data in matrix form, we have to click on 'Create button to save those matrices.

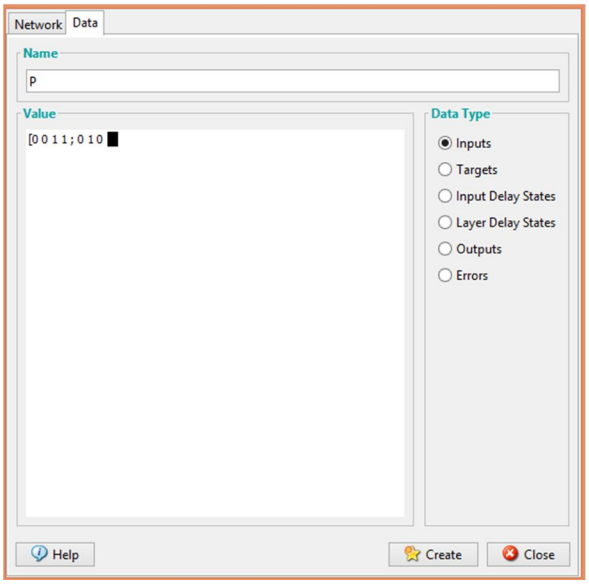

Figure2: Entering input and target data in Neural Network Toolbox

Input (P)		Target (T)
0	0	0
0	1	1
1	0	1
1	1	0

Table1: Truth table for XOR problem

To create a neural network, we can click on 'New' button and then click on 'Network' tab. Then we choose number of network as 2 and select target and input data from drop down list. Now we change transfer function to 'TANSIG' for both layers (layer1: hidden layer and layer2: output layer). At last we give a name of this network and click on 'Create' to finish the process.

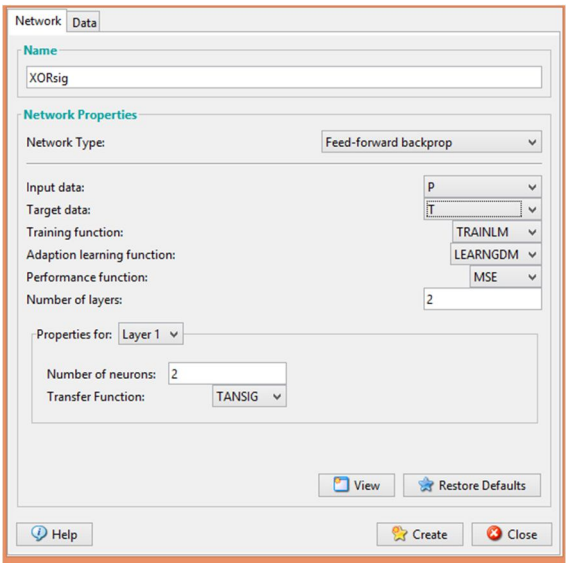

Figure3: Creating 2-2-1 neural network using sigmoid transfer function

Now, we can select our created network from 'Network' and double click to view the selected network.

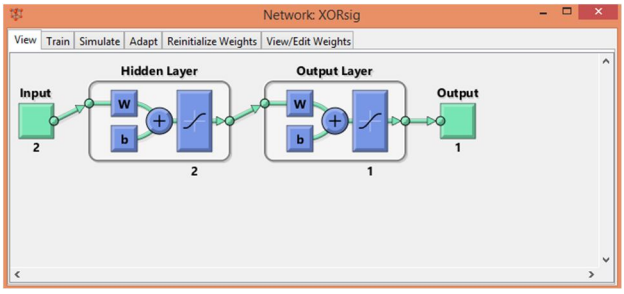

Figure4: Viewing 2-2-1 neural network (with sigmoid PE)

If we click on 'Train' tab then we will see GUI like figure5 where we have two tabs ('Training Info' and 'Training Parameters'). We have to select input and target data matrix in 'Training Info' tab. Then we have to setup some parameters in 'Training Parameters' tab like number of epoch we want to run the training, excepted error etc. Now, we can click on 'Train Network' button to see the output.

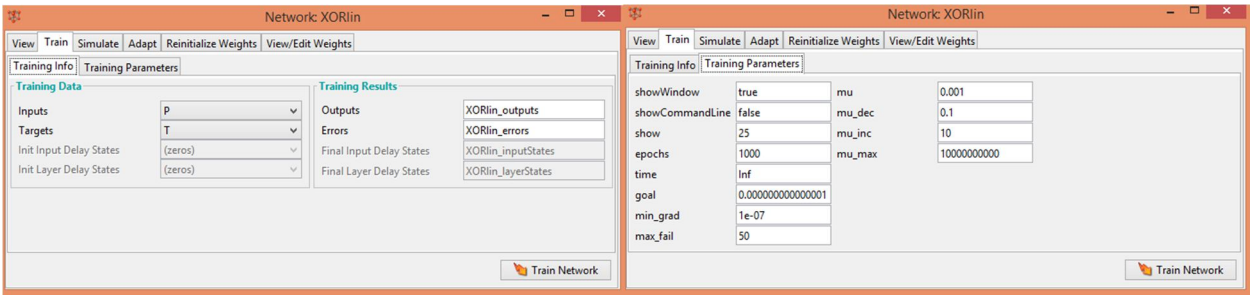

Figure5: Selecting training info and parameters in neural network GUI

After running the training operation we will get the result in matrix form at 'Output Data' and 'Error Data' window (figure1). In the case of selecting sigmoid transfer function, we get the following results:

Output Data = [0.98001 0.883 0.39732 0.77429]

Error Data = [-0.98001 0.117 0.60268 -0.77429]

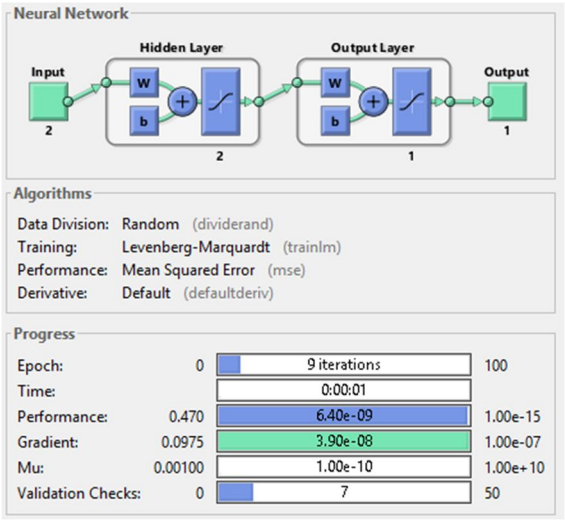

Figure6: Neural Network training output using sigmoid transfer function

B) DEVELOP AN ANN WITH THE ALL LINEAR PE:

Designing a neural network for XOR problem using linear transfer function is almost same as part (A). Just in figure3, we have to choose 'PURELIN' transfer function instead of 'TRANSIG' and select ADALINE algorithm for the network. Rest of the process are same as part (A). In the case of selecting sigmoid transfer function, we get the following results:

Output Data = [0.64497 0.38797 0.25447 -0.0025299]

Error Data = [-0.64497 0.61203 0.74553 0.0025299]

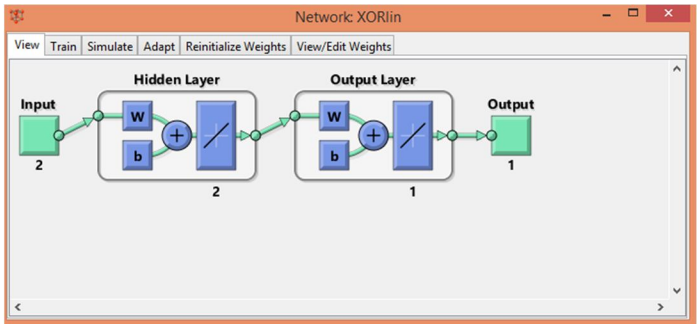

Figure7: Viewing 2-2-1 neural network (with linear PE)

(C) COMPARE THE RESULTS OF PART(A) &PART(B):

We get the following performance curve for sigmoid PE and linear PE respectively.

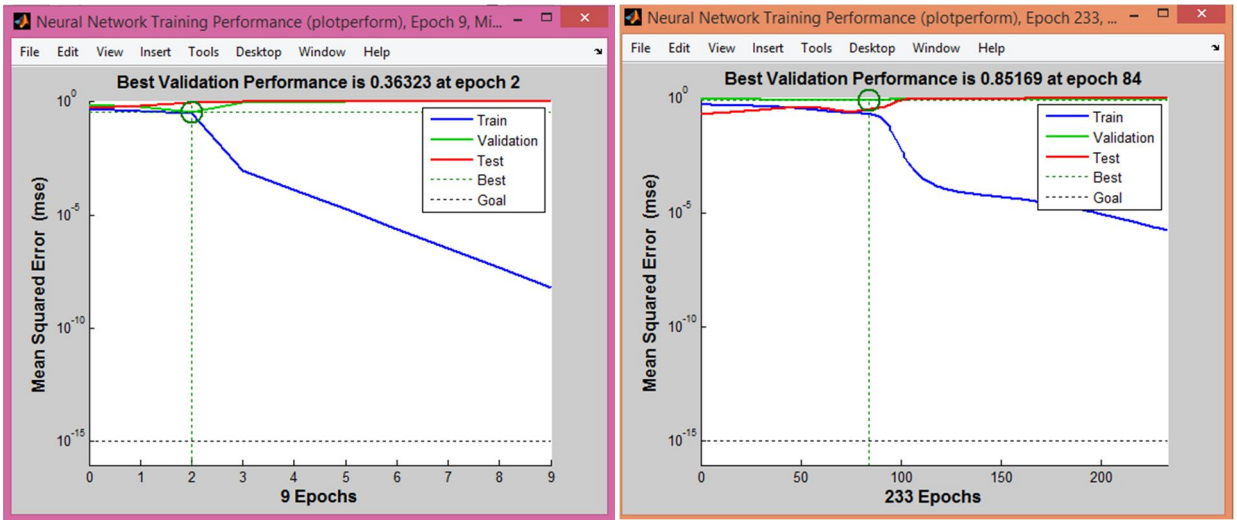

Figure8: Performance result for sigmoid PE (left) and linear PE (right)

We can see from the performance curves, sigmoid function gives the best solution after 2 epochs and reach to desired mean squared error after 9 epochs. On the other hand, linear PE and ADALINE network took long time to reach desired mean squared error (233 epochs). It reaches to the best solution after 84 epochs.

PROBLEM TWO: NON-LINEAR FUNCTION MAPPING

A) $f(x) = \frac{1}{x}$ $x \in (0.1,1)$:

I wrote a MATLAB code (dataANN_1.m) which can generate 3 different sets of input patters (training data, testing data & validation data) from the given function. The code is provided with the project.

The output of the code is given below:

P_train = [0.1000 0.1462 0.1923 0.2385 0.2846 0.3308 0.3769 0.4231 0.4692 0.5154 0.5615 0.6077 0.6538 0.7000 0.7462 0.7923 0.8385 0.8846 0.9308 0.9769]

P_test = [0.1231 0.2154 0.3077 0.4000 0.4923 0.5846 0.6769 0.7692 0.8615 0.9538]

P_valid = [0.1692 0.2615 0.3538 0.4462 0.5385 0.6308 0.7231 0.8154 0.9077 1.0000]

T_train = [10.0000 6.8421 5.2000 4.1935 3.5135 3.0233 2.6531 2.3636 2.1311 1.9403 1.7808 1.6456 1.5294 1.4286 1.3402 1.2621 1.1927 1.1304 1.0744 1.0236]

T_test = [8.1250 4.6429 3.2500 2.5000 2.0313 1.7105 1.4773 1.3000 1.1607 1.0484]

T_valid = [5.9091 3.8235 2.8261 2.2414 1.8571 1.5854 1.3830 1.2264 1.1017 1.0000]

Here, we have 3 pair of variables. P_Train and T_train contain 20 input and target data respectively for training data set. Similarly P_test and T_test contain 10 input and target data respectively for testing data set and P_valid and T_valid contain 10 input and target data respectively for validation data set. After running the back propagation algorithm, I get the following results.

Neural Network	Epochs	MSE	MATLAB code
1-2-1	83	0.2221 (best solution)	bp_1_2_1.m
1-3-1	102	0.1762 (best solution)	bp_1_3_1.m
1-4-1	120	0.1781 (best solution)	bp_1_4_1.m
1-4-1-1	529	0.0120 (best solution)	bp_1_4_1_1.m
1-7-2-1	30000	0.0007 (More improvement possible)	bp_1_7_1_1.m

Table2: Mean squared error for different ANN to solve f(x)=1/x using backpropagation algorithm

From Table2, we can observe that when we use only one hidden layer then 1-4-1 network gives the best solution after 120 epochs. In this case the mean squared error using our validation data is 0.1781. On the other hand when we use 2 hidden layers, we can generate more accurate result with the cost of expensive calculation. The 1-7-2-1 network gives the most accurate result for this function. I have tried till 30000 epochs and it seemed still improving with the increment of epochs.

(B) $f(x,y) = x^2 + y^2 \qquad x \in (-1,1)\ and\ y \in (-1,1)$

I wrote a MATLAB code (dataANN_2.m) which can generate 3 different sets of input patters (training data, testing data & validation data) from the given function. The code is provided with the project.

The output of the code is given below:

P_train =[-1.0000 -1.0000; -0.8974 -0.8974; -0.7949 -0.7949; -0.6923 -0.6923; -0.5897 -0.5897; -0.4872 -0.4872; -0.3846 -0.3846; -0.2821 -0.2821; -0.1795 -0.1795; -0.0769 -0.0769; 0.0256 0.0256; 0.1282 0.1282; 0.2308 0.2308; 0.3333 0.3333; 0.4359 0.4359; 0.5385 0.5385; 0.6410 0.6410; 0.7436 0.7436; 0.8462 0.8462; 0.9487 0.9487]

P_test = [-0.9487 -0.9487; -0.7436 -0.7436; -0.5385 -0.5385; -0.3333 -0.3333; -0.1282 -0.1282; 0.0769 0.0769; 0.2821 0.2821; 0.4872 0.4872; 0.6923 0.6923; 0.8974 0.8974]

P_valid = [-0.8462 -0.8462; -0.6410 -0.6410; -0.4359 -0.4359; -0.2308 -0.2308; -0.0256 -0.0256; 0.1795 0.1795; 0.3846 0.3846; 0.5897 0.5897; 0.7949 0.7949; 1.0000 1.0000]

T_train = [2.0000; 1.6108; 1.2636; 0.9586; 0.6956; 0.4747; 0.2959; 0.1591; 0.0644; 0.0118; 0.0013; 0.0329; 0.1065; 0.2222; 0.3800; 0.5799; 0.8218; 1.1059; 1.4320; 1.8001]

T_test = [1.8001; 1.1059; 0.5799; 0.2222; 0.0329; 0.0118; 0.1591; 0.4747; 0.9586; 1.6108]

T_valid = [1.4320; 0.8218; 0.3800; 0.1065; 0.0013; 0.0644; 0.2959; 0.6956; 1.2636; 2.0000]

Here, variables P_train, T_train, P_test, T_test, P_valid and T_valid contain training, testing and validation data sets repectively.

Neural Network	Epochs	MSE	MATLAB code
2-2-1	15000	0.0019 (More improvement possible)	bp_2_2_1.m
2-3-1	15000	0.0017 (More improvement possible)	bp_2_3_1.m
2-4-1-1	15000	0.0011 (More improvement possible)	bp_2_4_1_1.m

Table3: Mean squared error for different ANN to solve $f(x,y)=x^2+y^2$ using backpropagation algorithm

From Table3, we can observe that when we use multiple hidden layers, then we get better result. I have tried for 15000 epochs for all above cases and mean squared error was improving till 15000 epochs. In next part we will compare our result of backpropagation with Levengber-Marquardt method.

(C) COMPARING THE RESULT OF BACKPROPAGATION (BP) WITH LEVENGBER-MARQUARDT (LM) METHOD:

Table4 clearly shows that Levengber-Marquardt (LM) algorithm is faster than Backpropagation (BP) algorithm. Everytime LM converges faster than BP and also mean squared error is less for LM method. Thus I think LM algorithm is better than BP algorithm.

Function	Neural Network	BP method MSE	LM method MSE
f(x) = 1/x	1-2-1	0.2221 (83 epochs)	0.0184 (6 epochs)
	1-3-1	0.1762 (102 epochs)	0.0003 (1000 epochs)
	1-4-1	0.1781 (120 epochs)	0.1087 (7 epochs)
$f(x,y) = x^2+y^2$	2-2-1	0.0019 (15000 epochs)	0.0001 (30 epochs)
	2-3-1	0.0017 (15000 epochs)	0.0001 (100 epochs)

Table4: Comparison of Backpropagation and Levenber-Marquardt method

One typical performance graphs and function fit graph is given below in figure9 and figure10 respectively.

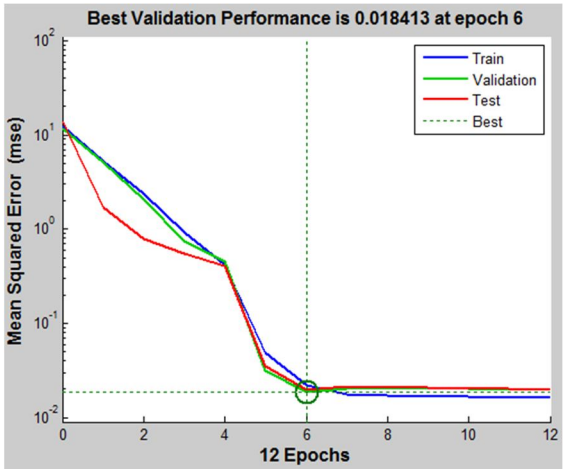

Figure9: Performance graph for 1-2-1 network using LM algorithm

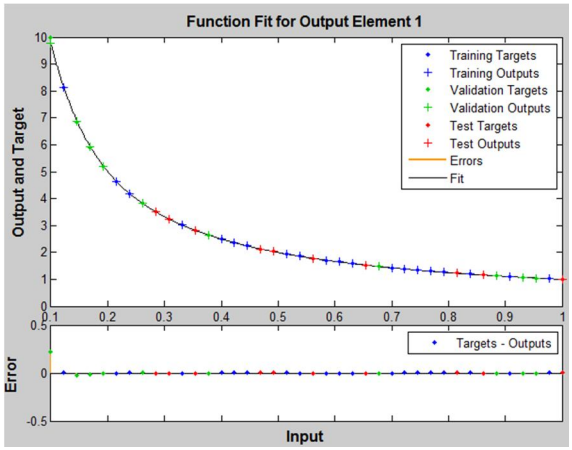

Figure10: Function fit graph for y=1/x using LM algorithm

PROBLEM THREE: DEVELOP AN ANN THAT PREDICTS THE S&P500 DATA

At first we imported the 'SP500.csv' file in MATLAB column by column. Thus we have 2 columns named 'Date1' and 'ClosingPrice'. There are several tools in MATLAB to analysis and predict time series. We have used 'Neural Network Time Series' tool to do the job. This tool takes time series input as number, thus we had to convert into number. We use the following commands in MATLAB to convert date to number and open neural network time series tool box.

```
>> Date = datenum(Date1);

>> ntstool
```

Then we selected 'Nonlinear autoregressive with external (exogenous) input (NARX)' model to predict our given time series data. We have selected date as input and closing price of stock as target value in next window. Our given dataset have 820 values, we separate this dataset into 3 parts: 574 values for training (70% of total data), 123 values for testing (15% of total data) and 123 values for validation (15% of total data. Training, testing and validation datasets are randomly selected from original 820 data. We use 1-10-1 neural network (one hidden layer with 10 neurons, with number of delay =2) to train our given time series. Our neural network performances are illustrated below using some error plots.

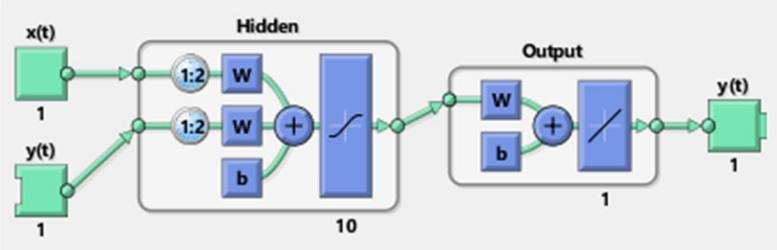

Figure11: Neural Network (1-10-1) used to predict time series data

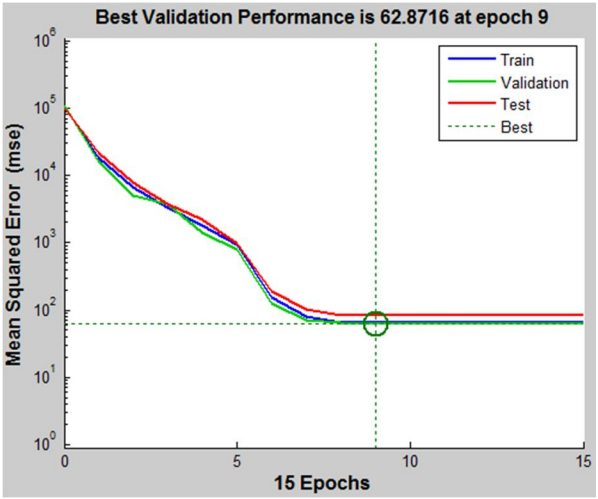

Figure12: Performance graph for time series data fitting

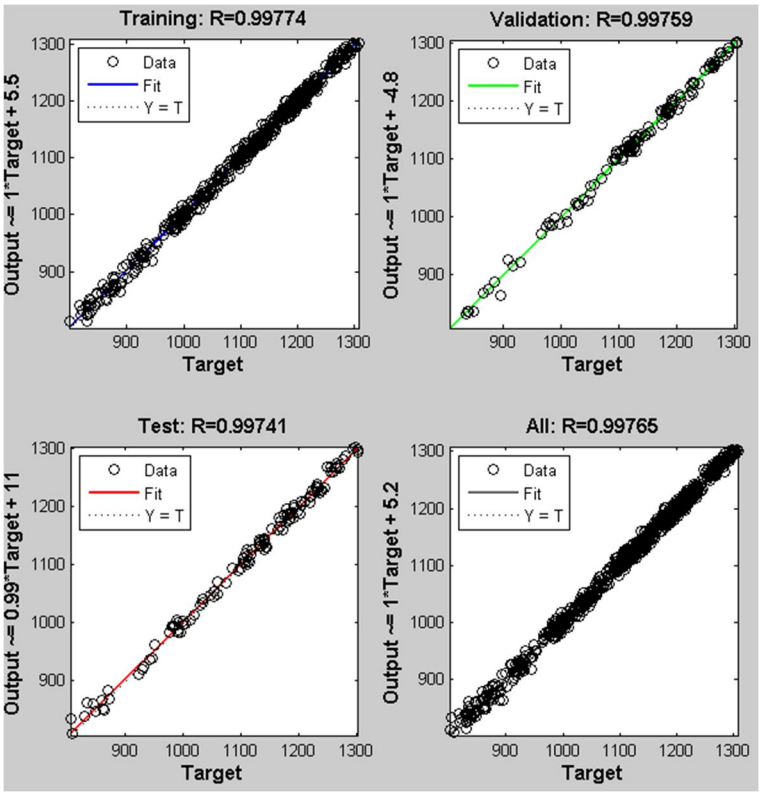

Figure13: Training, testing and validation performance of given time series

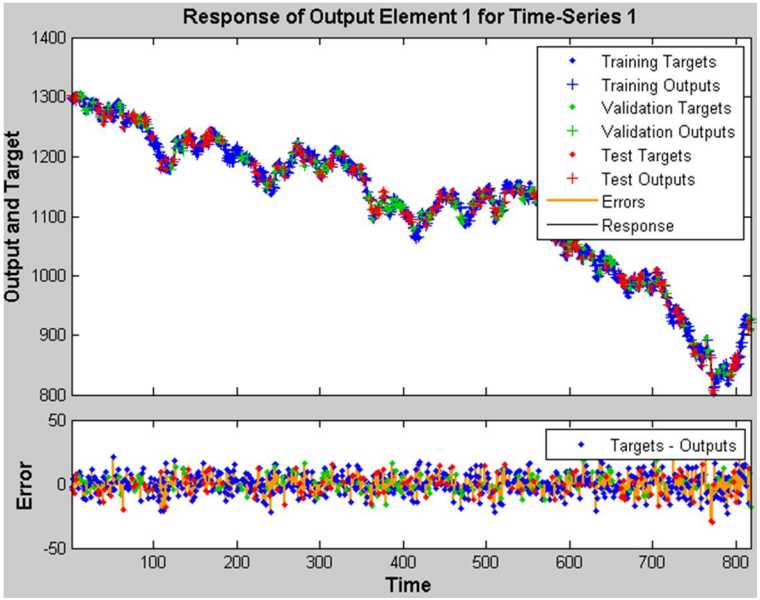

Figure14: Response of output element for given time series