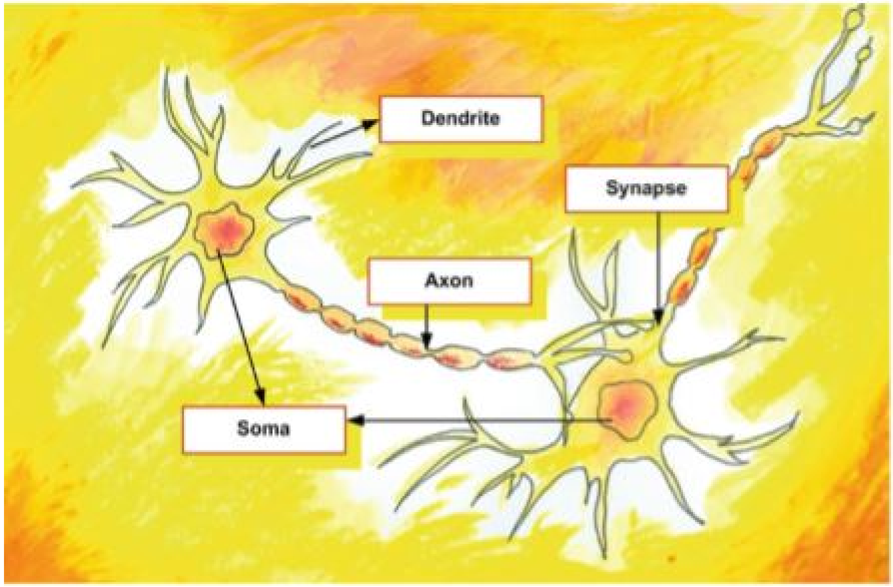
Machine Learning Practical -2 Artificial Neural Network (ANN)

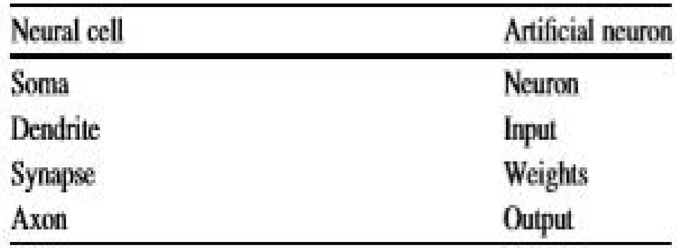
Lecturer:

Dr Hamid Khayyam (Australia)
Email: hamid.khayyam@rmit.edu.au

RMIT UNIVERSITY

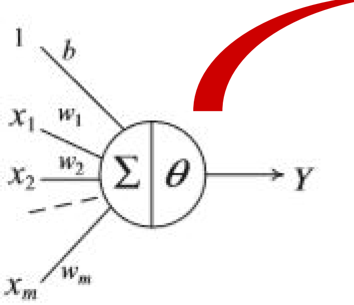
Recap





Neural cell	Artificial neuron
Soma	Neuron
Dendrite	Input
Synapse	Weights
Axon	Output

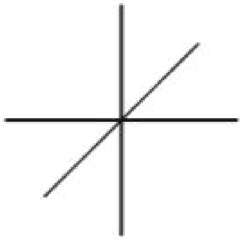
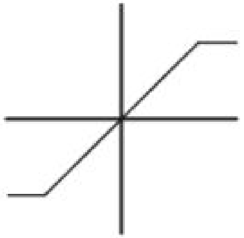
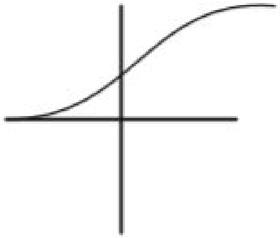
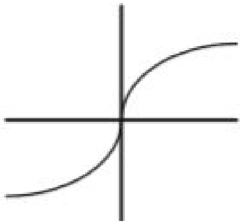
$$I = W * X + b$$



$$Y = \begin{cases} 1 \ \ if \ I \geq \theta \\ 0 \ if \ I < \theta \end{cases}$$

The mathematical relation of the functional process of an artificial neuron

Recap (cont.)

Name	Function	Graphs
Linear	$f(x) = x$	
Symmetric–saturating–linear	$f(x) = \begin{cases} \delta & x \geq \theta \\ x & -\theta \leq x \leq \theta \\ -\delta & x \leq -\theta \end{cases}$	
Log sigmoid	$f(x) = \frac{1}{1+e^{-\alpha x}} \alpha > 0$	
Tangent sigmoid	$f(x) = \left(\frac{2}{1+e^{-\alpha x}}\right) - 1 \;\; \alpha > 0$	

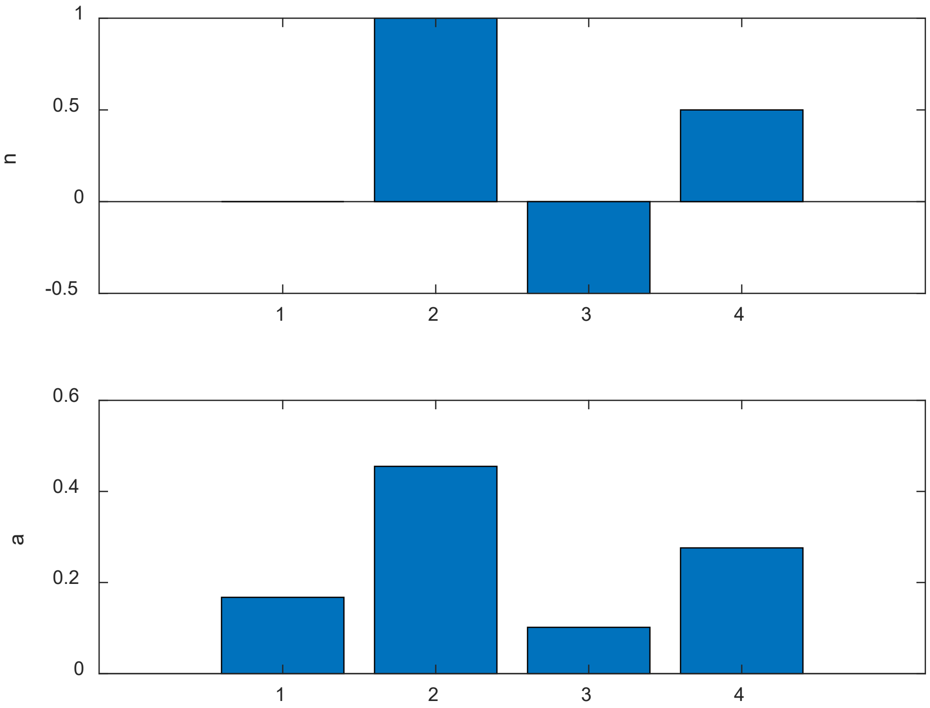
Example of transfer functions

Recap (cont.)

- **Softmax Activation functions (classification)**

Example :

```
n = [0; 1; -0.5; 0.5];
a = softmax(n); % softmax(n)=exp(n)/sum(exp(n))
subplot(2,1,1)
bar(n)
ylabel('n')
subplot(2,1,2)
bar(a)
ylabel('a')
```



```
net.layers{2}.transferFcn = 'softmax' ;          % for output layer(classification)
```

Example of classification :

This dataset is consist of :

Input (x) - a 699x9 matrix

Output (t) - a 699x2 matrix

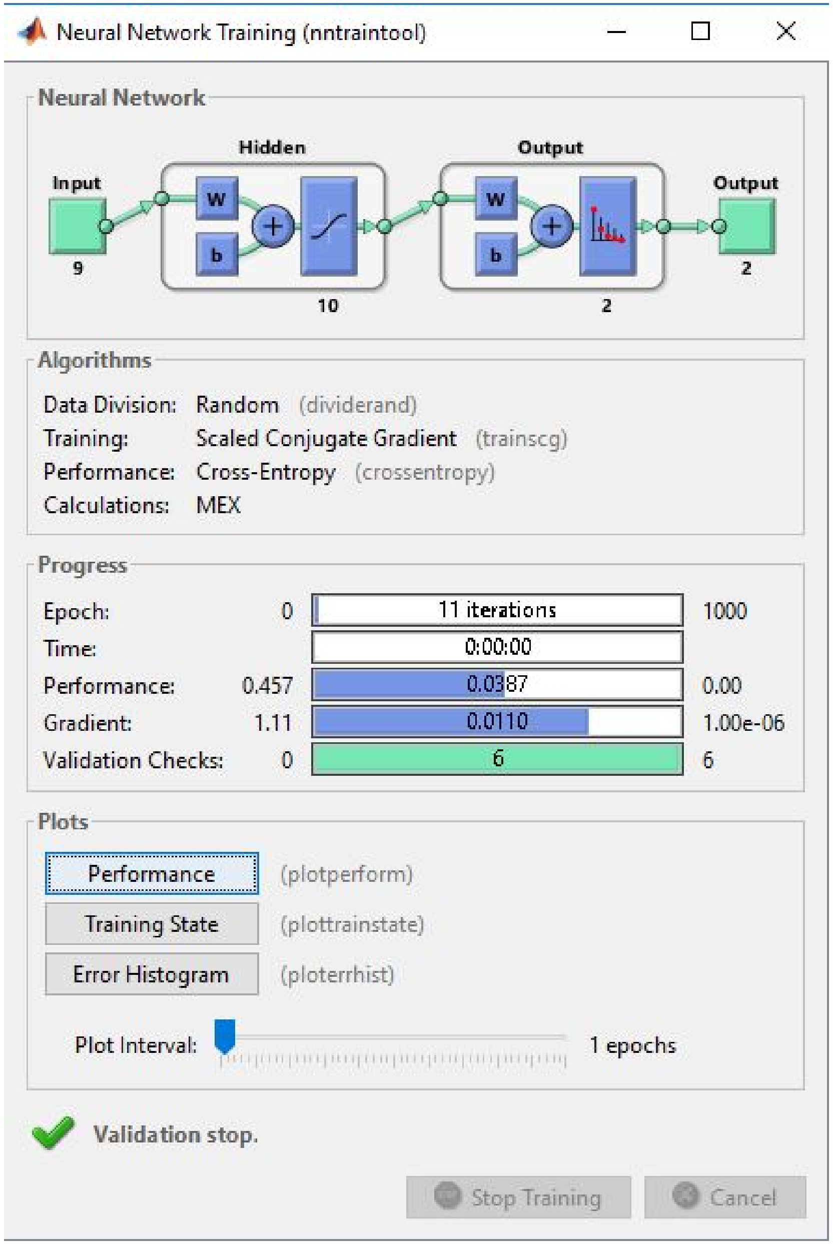
```
clear;clc;

Filename='classification.xlsx'; %classification.xlsx should be
used %

Sheetread='x';

Input1='A1:I699';

Sheetread1='t';

Target1 ='A1:B699';

Input=xlsread(Filename,Sheetread,Input1);

Target=xlsread(Filename,Sheetread1,Target1 );

x=Input';

t=Target';
```

Example of classification: (cont.)

```
hiddenLayerSize = 10;

trainFcn = 'trainscg'; %Training function for classification

net = patternnet(hiddenLayerSize, trainFcn); % Network Architecture

net.input.processFcns = {'mapminmax'}; % To standardize data

RandStream.setGlobalStream (RandStream ('mrg32k3a’)); %Data division

net.divideMode = 'sample';

net.divideParam.trainRatio = 70/100;

net.divideParam.valRatio = 15/100;

net.divideParam.testRatio = 15/100;

net.performFcn = 'crossentropy';  % Cross-Entropy for network
performance in classification

net.plotFcns = {'plotperform','plottrainstate','ploterrhist'};

[net,tr] = train(net,x,t); % train the network

view(net)                  % view neural network

y = net(x);                % Compute predictions from the inputs

e = gsubtract(t,y);        % Compare the predictions to the targets
(true values)
```

Example of classification : (cont.)

```
% Performance plot
figure;plottrainstate(tr);        % Trainstate plot
performance = perform(net,t,y);   % Network performance
figure;plotperform(tr);
trainTargets = t .* tr.trainMask{1}; % Apply a mask (0's and 1's to select the proper targets)
valTargets = t .* tr.valMask{1}; %Select validation data
testTargets = t .* tr.testMask{1}; %select test data
trainPerformance = perform(net,trainTargets,y) %calculate training performance
valPerformance = perform(net,valTargets,y) %calculate validation performance
testPerformance = perform(net,testTargets,y) %calculate Testing performance
```



Algorithms

Data Division: Random (dividerand)
Training: Scaled Conjugate Gradient (trainscg)
Performance: Cross-Entropy (crossentropy)
Calculations: MEX

Progress

Epoch:	0	11 iterations	1000
Time:		0:00:00	
Performance:	0.457	0.0387	0.00
Gradient:	1.11	0.0110	1.00e-06
Validation Checks:	0	6	6

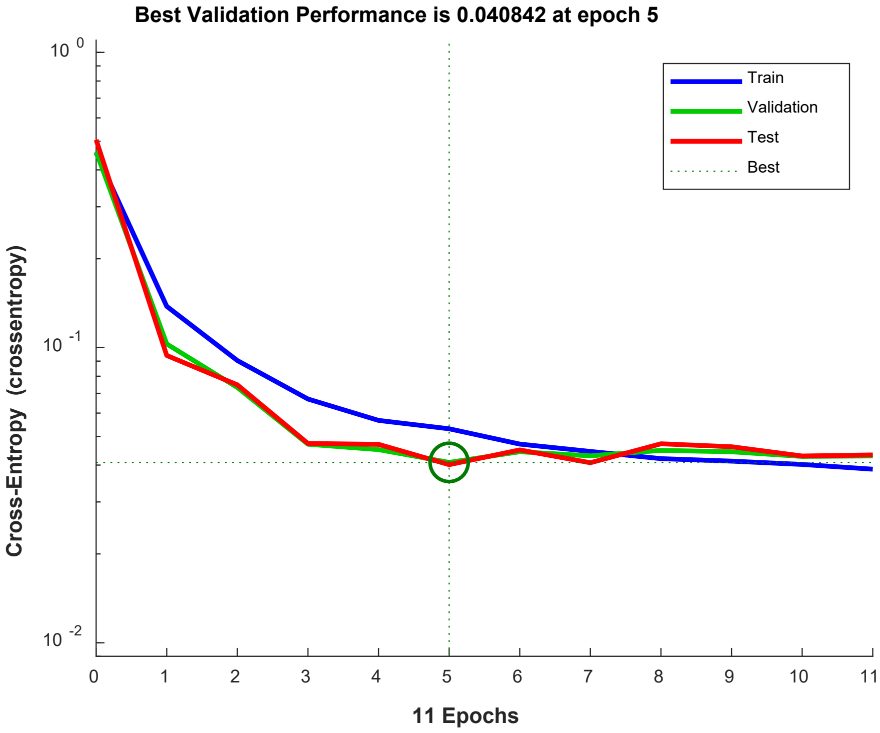
Plots

Performance (plotperform)
Training State (plottrainstate)
Error Histogram (ploterrhist)

Plot Interval: 1 epochs

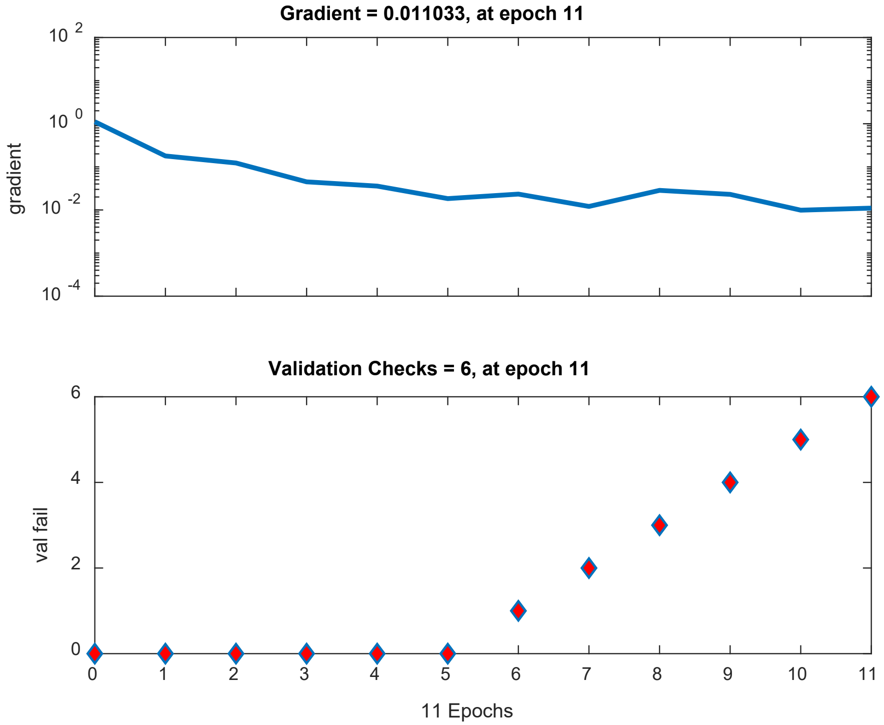
✓ Validation stop.

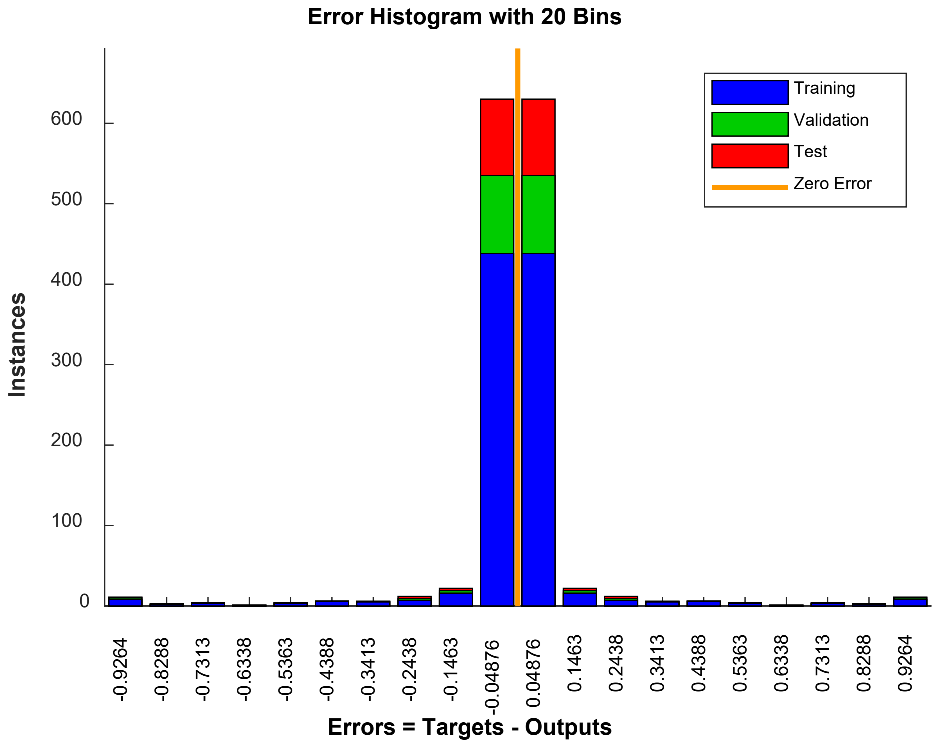
Neural Network Training



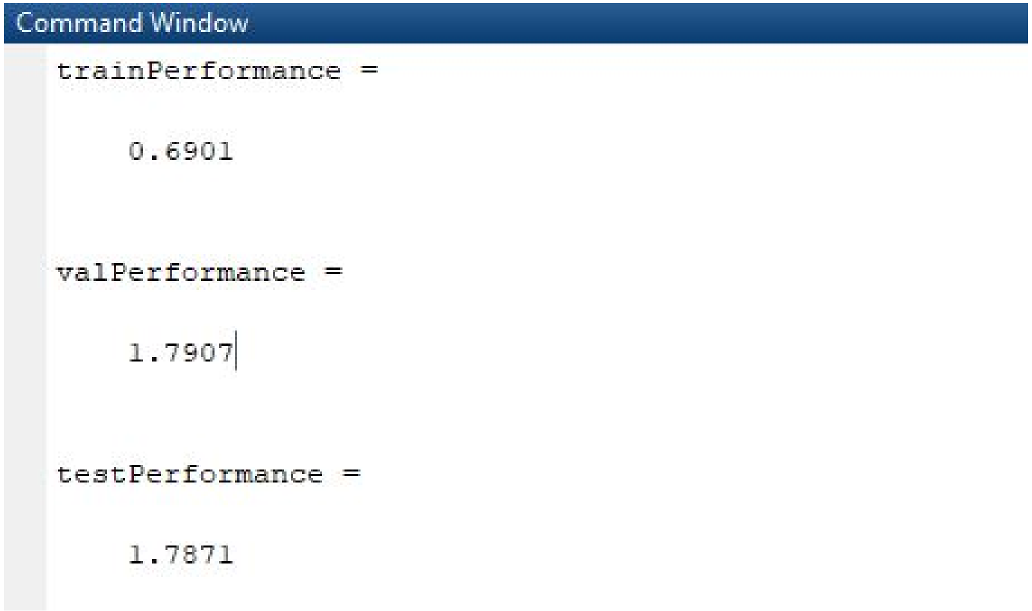
Plotperform

Plottrainstate





Ploterrhist



```
trainPerformance =

    0.6901


valPerformance =

    1.7907


testPerformance =

    1.7871
```

Example of regression (1):

This dataset is consist of :

Input (x) - a 214x13 matrix

Output (t) - a 214x1 matrix

```
clear;clc;

Filename='Regression1.xlsx'; %regression1.xlsx should be used %

Sheetread='x';

Input1='A1:M214';

Sheetread1='t';

Target1 ='A1:A214';

Sheetread2='xnew'; %load xnew and tnew for further testing the net
with new data.

Input2='A1:M38';

Sheetread3='tnew';

Target2 ='A1:A38';

Input=xlsread(Filename,Sheetread,Input1);

Target=xlsread(Filename,Sheetread1,Target1 );

Inputnew=xlsread(Filename,Sheetread2,Input2);
```

Example of regression (1): (cont.)

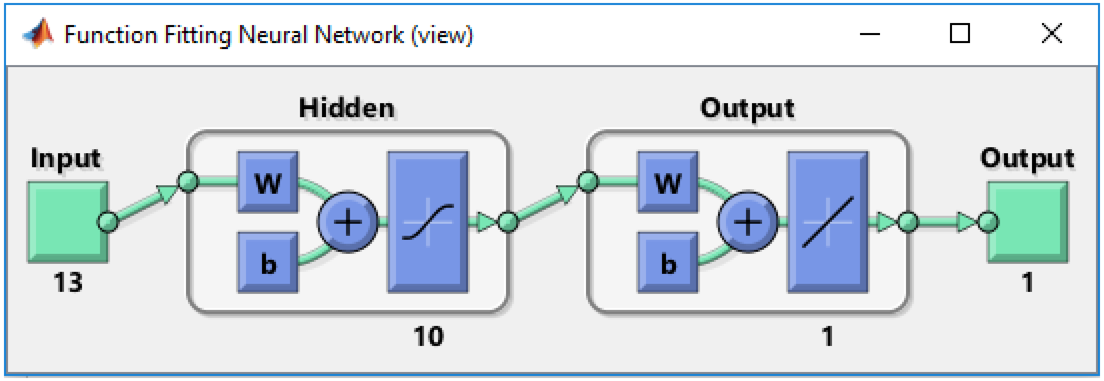
```
Targetnew=xlsread(Filename,Sheetread3,Target2 );

x=Input';

t=Target';

xnew=Inputnew';

tnew=Targetnew';

trainFcn = 'trainlm'; hiddenLayerSize = 10;

net = fitnet(hiddenLayerSize,trainFcn);

net.input.processFcns = {'mapminmax'}; % To standardize the input

RandStream.setGlobalStream (RandStream ('mrg32k3a'));% Just to get the same results;

% net.divideFcn = 'dividerand';  % Divide data randomly

net.divideMode = 'sample';

net.divideParam.trainRatio = 70/100;

net.divideParam.valRatio = 15/100;

net.divideParam.testRatio = 15/100;

net.performFcn = 'mse';  % Choose MSE for performance

[net,tr] = train(net,x,t);
```

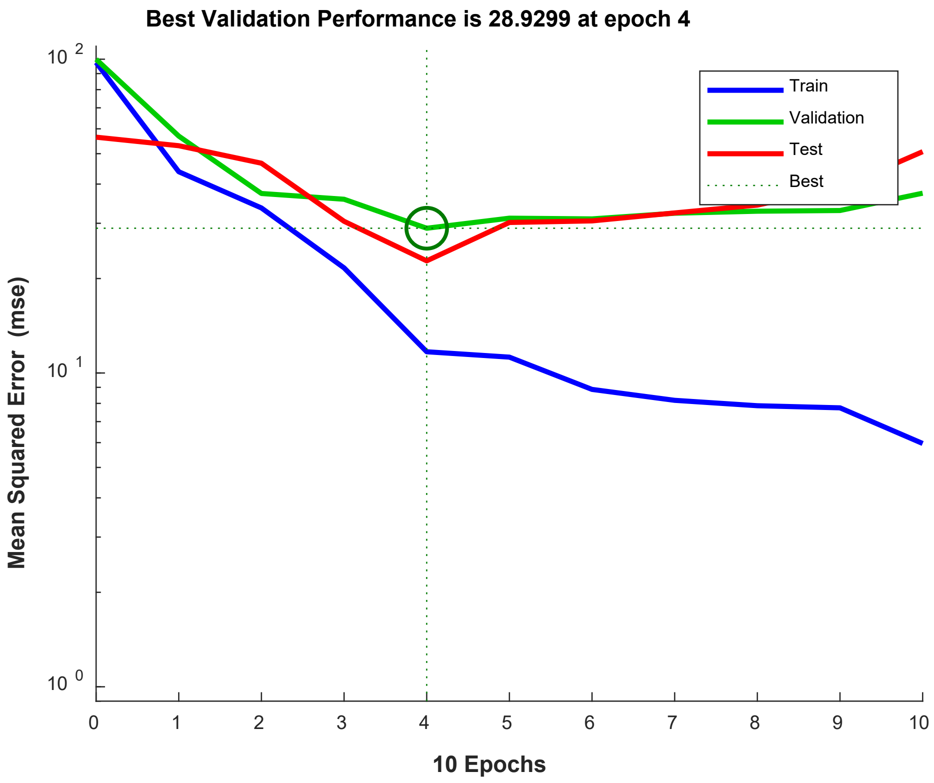
Example of regression (1): (cont.)

```
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y);
figure;plotperform(tr)
figure;plottrainstate(tr)
figure, plotregression(t,y)   % Regression plot
trainTargets = t .* tr.trainMask{1}; % Apply a mask (0's and 1's to select the proper targets) to select train data
valTargets = t .* tr.valMask{1};  %Select validation data
testTargets = t .* tr.testMask{1};  %select test data
trainPerformance = perform(net,trainTargets,y)  % training data performance
valPerformance = perform(net,valTargets,y)      % validation data performance
testPerformance = perform(net,testTargets,y)    %test data performance
```

Example of regression(1): (cont.)

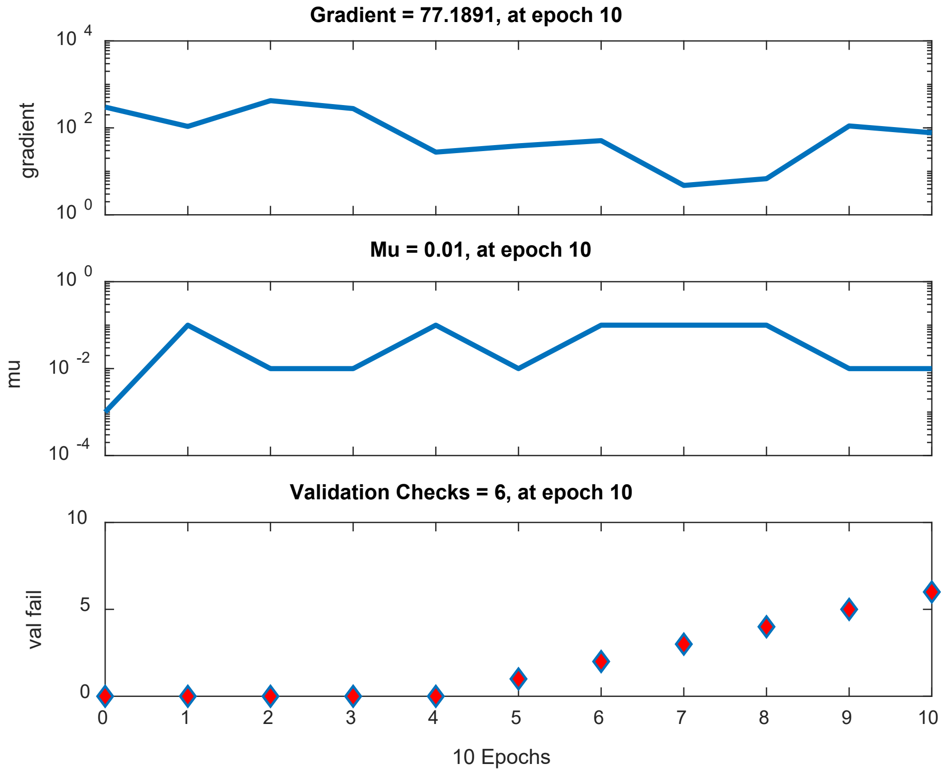
```
Ynew=net(xnew); %Test the net with new data to calculate
the performance and make predictions.

Newperformance=mse(tnew,Ynew); %MSE for new data

table( tnew( 1: 10)',Ynew( 1: 10)', 'VariableNames',...

{'Actual_data_tnew',' Predicted_data_Ynew'}) % Prediction
of tnew and Ynew for first 10th data
```

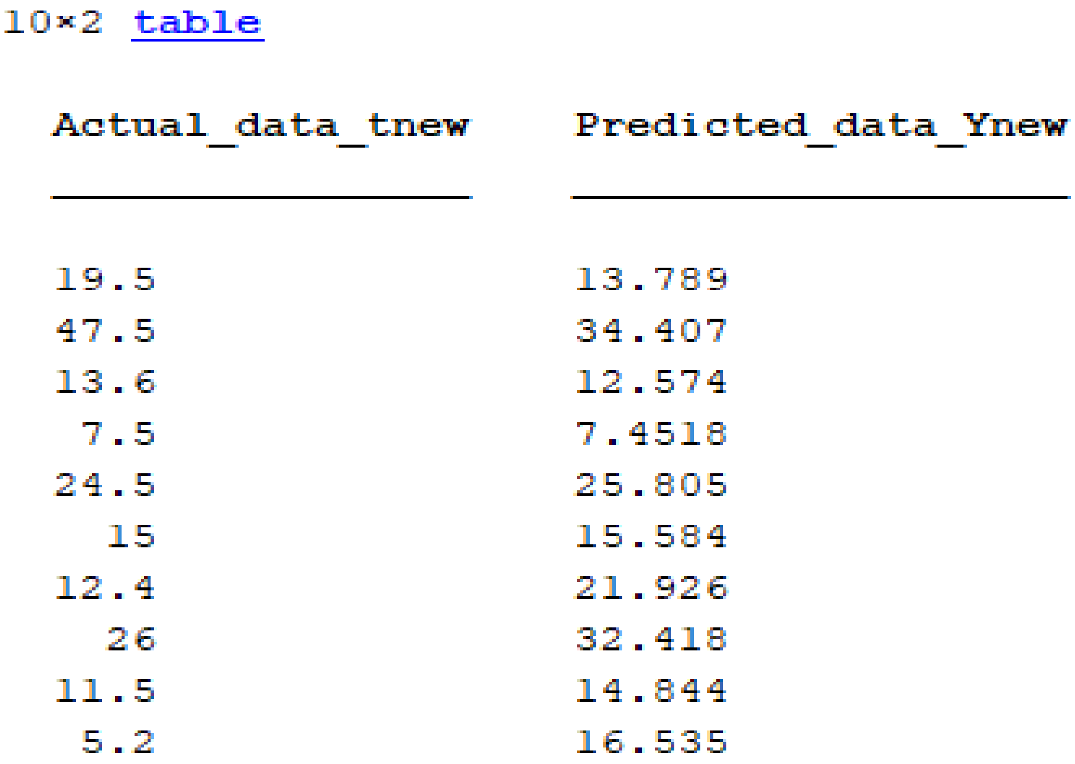




Plotperform

Plottrainstate





```
10×2 table

  Actual_data_tnew    Predicted_data_Ynew
  ________________    ___________________

  19.5                13.789
  47.5                34.407
  13.6                12.574
   7.5                7.4518
  24.5                25.805
    15                15.584
  12.4                21.926
    26                32.418
  11.5                14.844
   5.2                16.535
```

Prediction results

Example of regression (2):

This dataset is consist of :

Input (x) - a 423x8 matrix

Output (t) - a 423x1 matrix

```
clear;
clc;
load x.mat   %Regression2.xlsx should be imported %
load t.mat
x = x';
t = t';
trainFcn = 'trainlm';
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize,trainFcn)
```

Example of regression (2): (cont.)

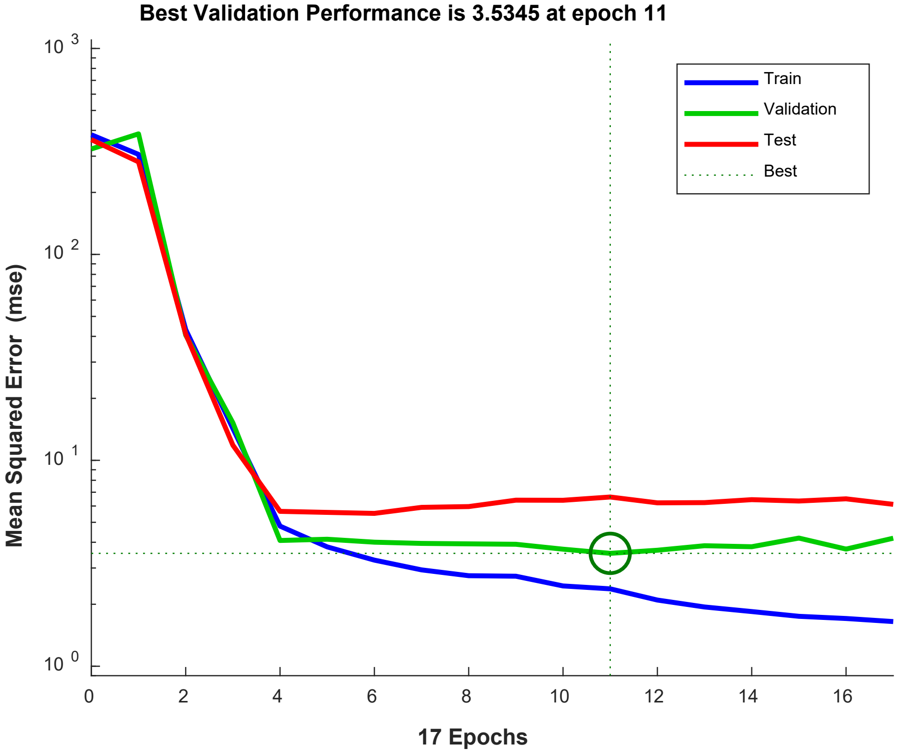
```
net.input.processFcns = {'mapstd'}; %Normalization of
input data

net.output.processFcns = {'mapstd'};%Normalization of
output data

net.layers{1}.transferFcn = 'logsig'; %Activation function
for hidden layer

net.layers{2}.transferFcn = 'purelin'; % Activation
function for output layer

% net.divideFcn = 'dividerand';

RandStream.setGlobalStream (RandStream ('mrg32k3a')); %To
get the same results as slides

net.divideMode = 'sample';

net.divideParam.trainRatio = 70/100;%Divide up the samples

net.divideParam.valRatio = 15/100;

net.divideParam.testRatio = 15/100;

net.performFcn = 'mse';
```

Example of regression (2): (cont.)

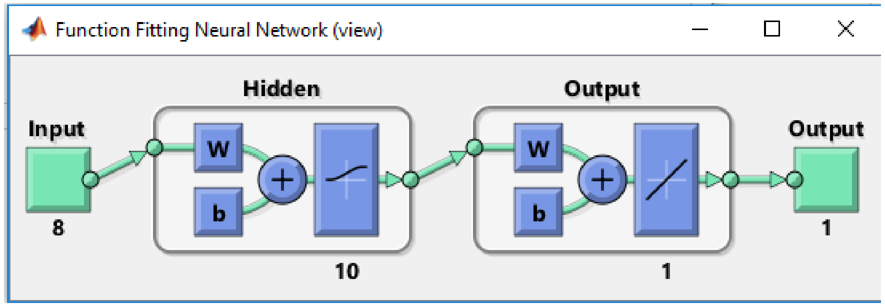
```
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', ...
    'plotregression'};
[net,tr] = train(net,x,t); %Train the network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y);
view(net);
figure, plotperform(tr);
figure, plottrainstate(tr);
figure, plotregression(t,y);
trainTargets = t .* tr.trainMask{1}; % Apply a mask (0's and 1's
to select the proper targets)
valTargets = t .* tr.valMask{1}; %Select validation data
testTargets = t .* tr.testMask{1}; %select test data
```

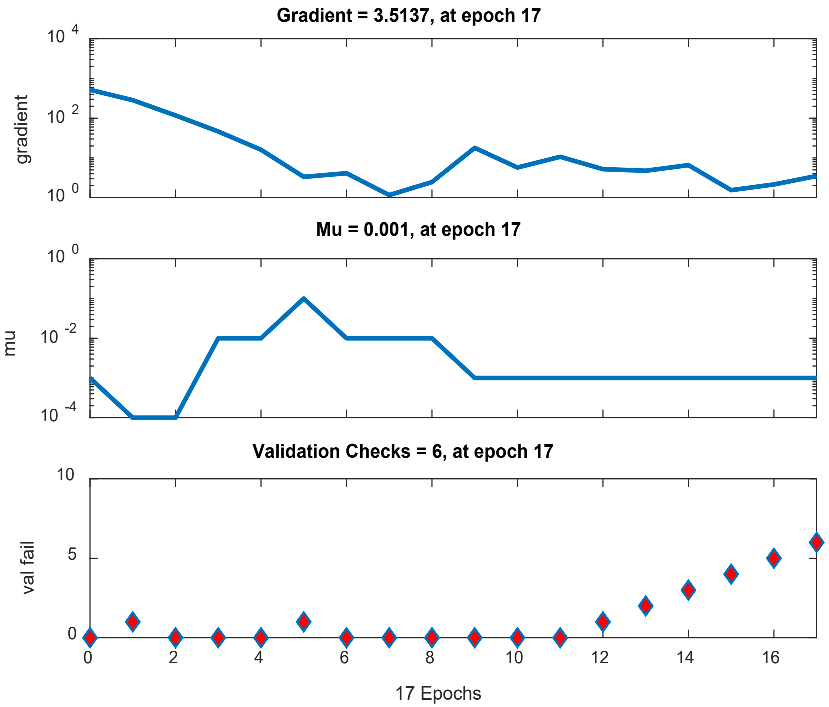
Example of regression (2): (cont.)

```
trainPerformance = perform(net,trainTargets,y) % Compute
performance for the training data

valPerformance = perform(net,valTargets,y) % Compute
performance for validation data

testPerformance = perform(net,testTargets,y) % compute
performance for test data

load xnew.mat; %load new data(unknowndata) for prediction

load tnew.mat;

xnew=xnew';

tnew=tnew';

Ynew=net(xnew);

Newperformance=mse(tnew,Ynew); %MSE for new data

table( tnew( 1: 10)',Ynew( 1: 10)', 'VariableNames',...

{'Actual_data',' Predicted_data'})

Errorpercentage=((tnew-Ynew)./tnew)*100; % Calculate error
percentage for tnew and ynew for first ten data
```

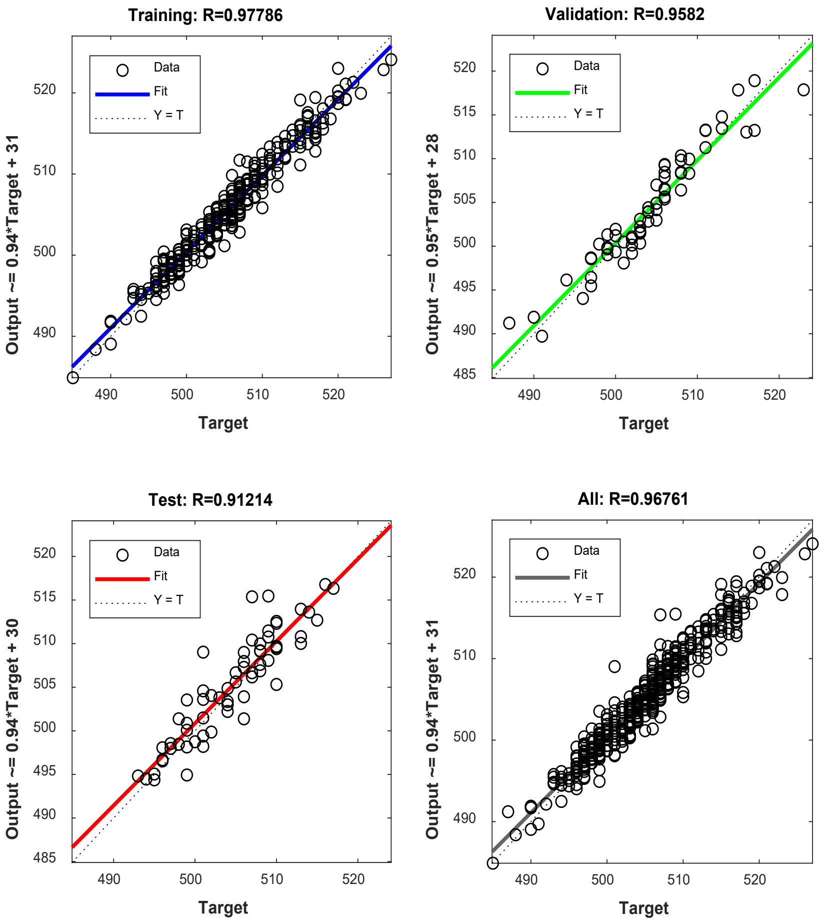


Plotperform

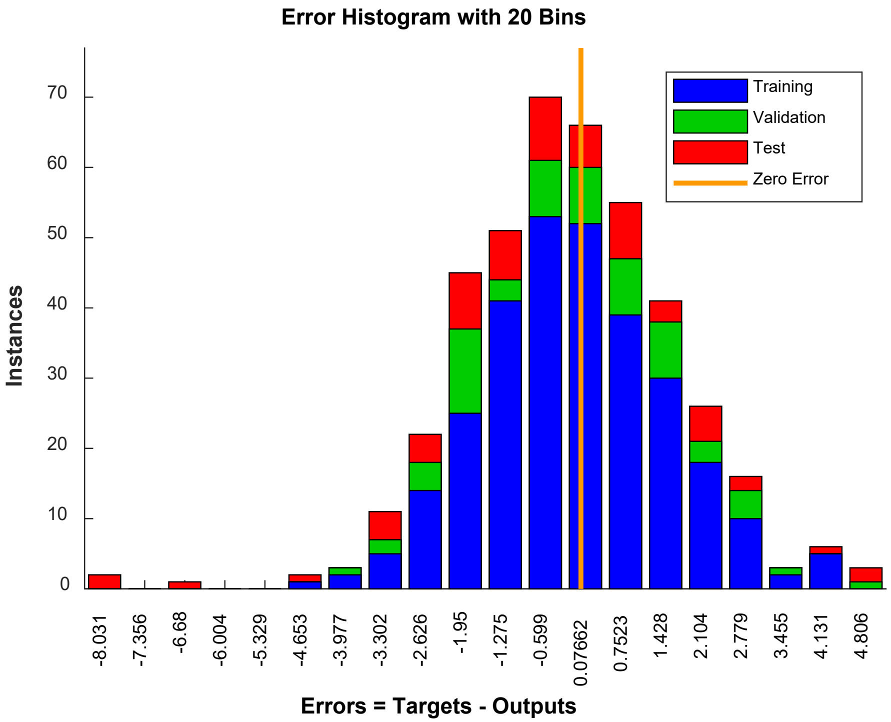




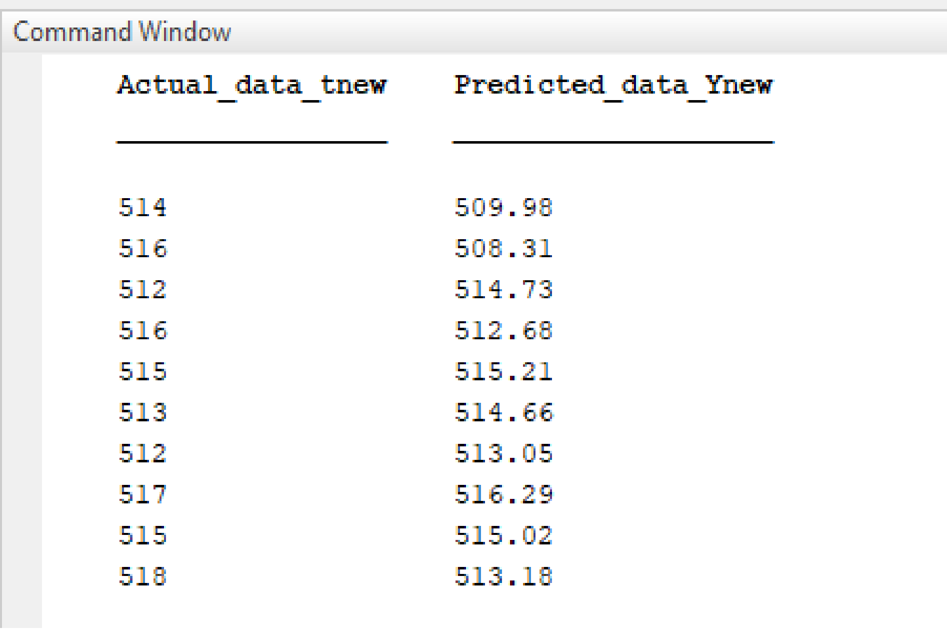
Plottrainstate



Plotregression

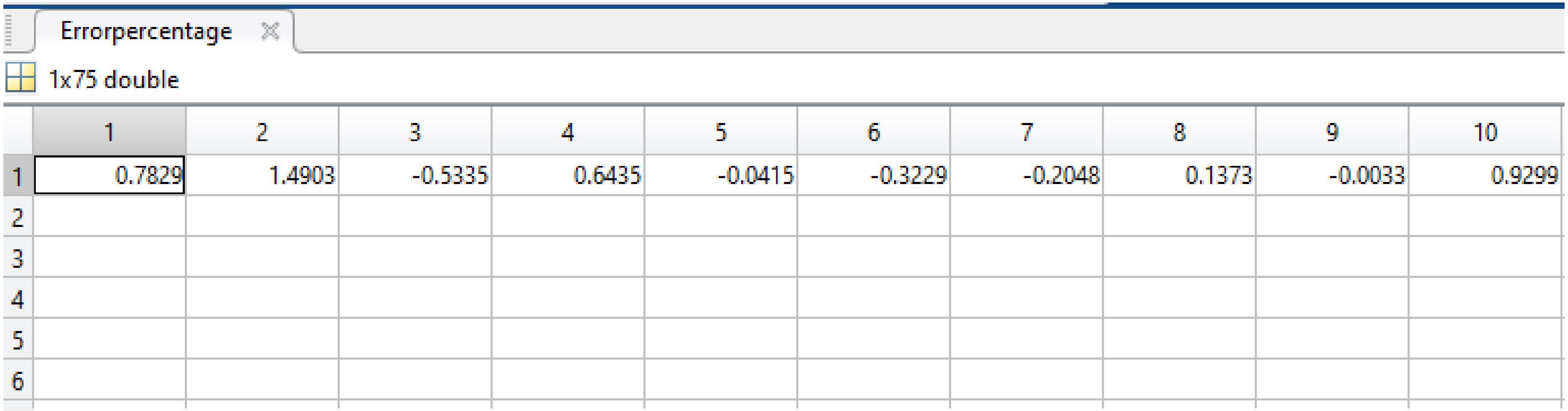


Ploterrhist



```
Actual_data_tnew      Predicted_data_Ynew
________________      ___________________

514                   509.98
516                   508.31
512                   514.73
516                   512.68
515                   515.21
513                   514.66
512                   513.05
517                   516.29
515                   515.02
518                   513.18
```

Prediction results



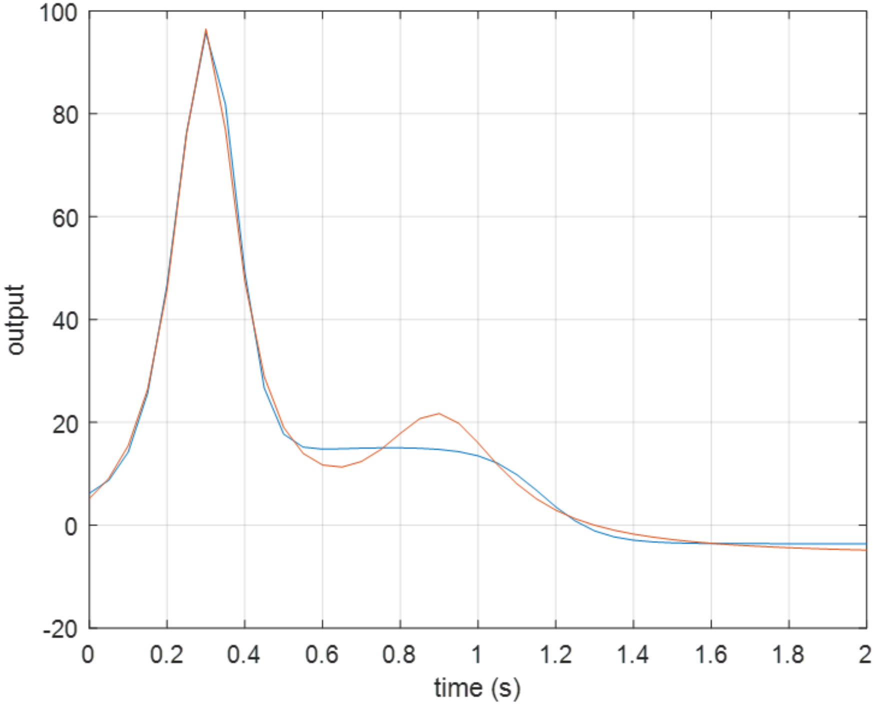
	1	2	3	4	5	6	7	8	9	10
1	0.7829	1.4903	-0.5335	0.6435	-0.0415	-0.3229	-0.2048	0.1373	-0.0033	0.9299
2										
3										
4										
5										
6										

Example of regression (3):

```
clear;clc; %The answers may vary due to randomness

x = 0:.05:2;

y = 1 ./ ((x-.3).^2 + .01) + 1 ./ ((x-.9).^2 + .04) - 6;

P=x; T=y;

plot(P,T,'*')

grid on;

xlabel('time (s)');

ylabel('output');

title('humps function');

net=fitnet(5); %Build the network with 5 neurons in the hidden layer

net.layers{1}.transferFcn = ‘logsig'; %activation function for hidden layer

net.layers{2}.transferFcn = 'purelin'; % activation function for output layer
```

Example of regression (3): (cont.)

```
net.divideFcn = 'dividerand'; %Run several times to get the best fit
net.divideMode = 'sample';
net.trainParam.epochs =1000; % Max number of iterations
net = train(net, P, T); % Training
a= net (P); % prediction of output
plot(P,a, P,T); grid; % Plot result and compare
xlabel('time (s)');
ylabel('output');
```


output
time (s)