Machine Learning
Practical -1
Artificial Neural Network (ANN)

Lecturer:

Dr Hamid Khayyam (Australia)

Email: hamid.khayyam@rmit.edu.au

RMIT UNIVERSITY

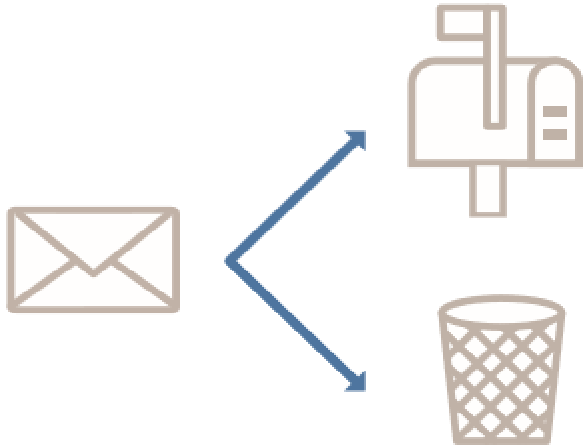
Machine learning definition and terms

Machine Learning: use computational methods to "learn" information directly from data without relying on a predetermined equation as a model.

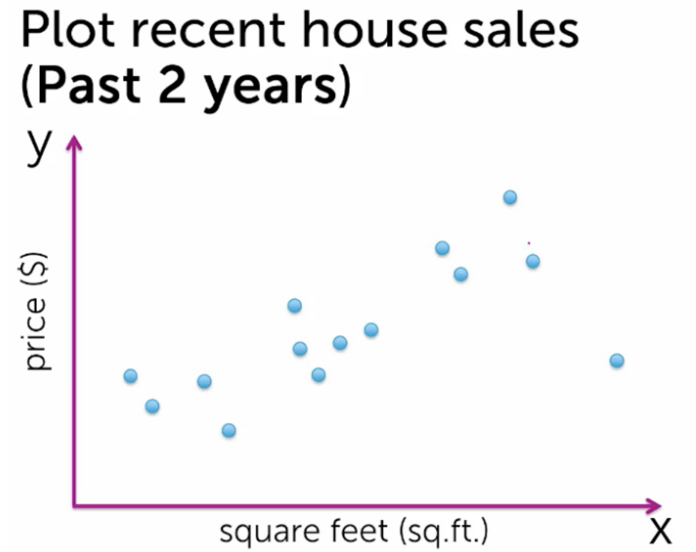
- **Machine learning algorithms:**
  - **Supervised :**
    - Classification: Discrete values output (e.g. 0 or 1, red, blue or green)
    - Regression: Predict continuous values output (e.g. price, temperature)
  - **Unsupervised (outside course scope) :**
    - Clustering

Reference : H. Khayyam, G. Golkarnaranji, R. Nakhaie Jazar, (2017) "Limited Data Modelling Approaches for Engineering Applications", Nonlinear Approaches in Engineering Applications, 978-3-319-69479-5, International publication Springer, (2017) .

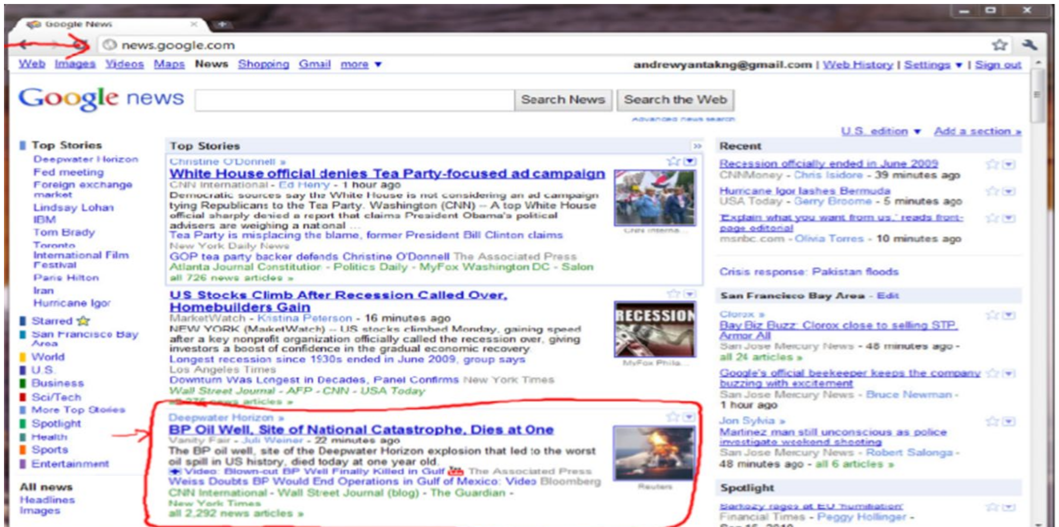
Examples of supervised and unsupervised learning



Email classification: supervised learning

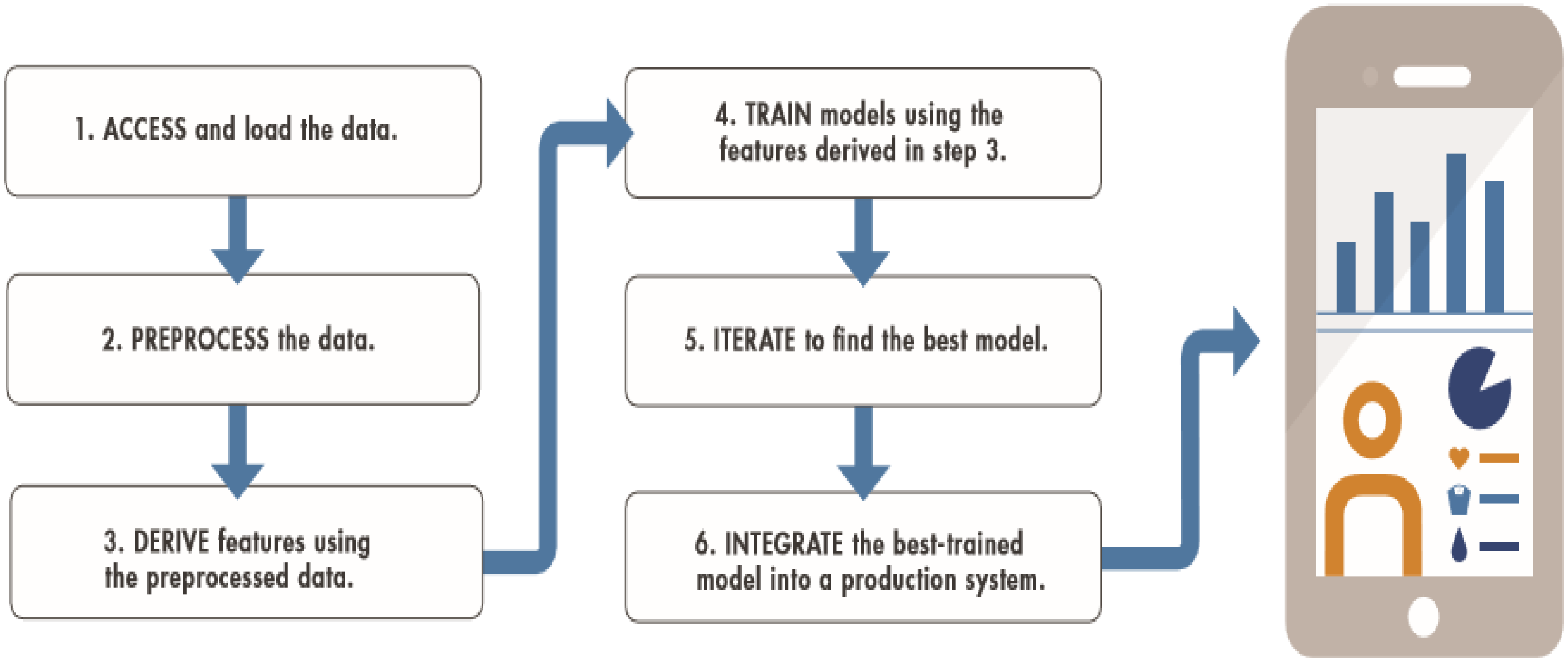


House sale prediction(regression): supervised learning



Google news grouping(clustering): unsupervised learning

Work flow for machine learning



1. Access and load the data

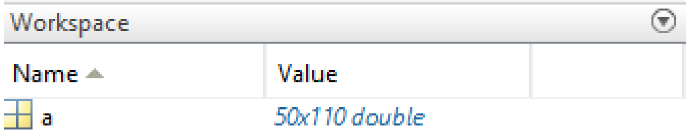
- **Load variables from file into workspace**

  Syntax:

load filename.mat or load('filename.mat');

  Example:

```
Clear;clc;
a=rand(50,110);              % create random data
save('output.mat','a');      % save variable in the output.mat file
clear;                       % clear the workspace
load('output.mat');          % load output.mat file
```



1. Access and load the data (cont.)

- **Read Microsoft Excel spreadsheet file**

Syntax:

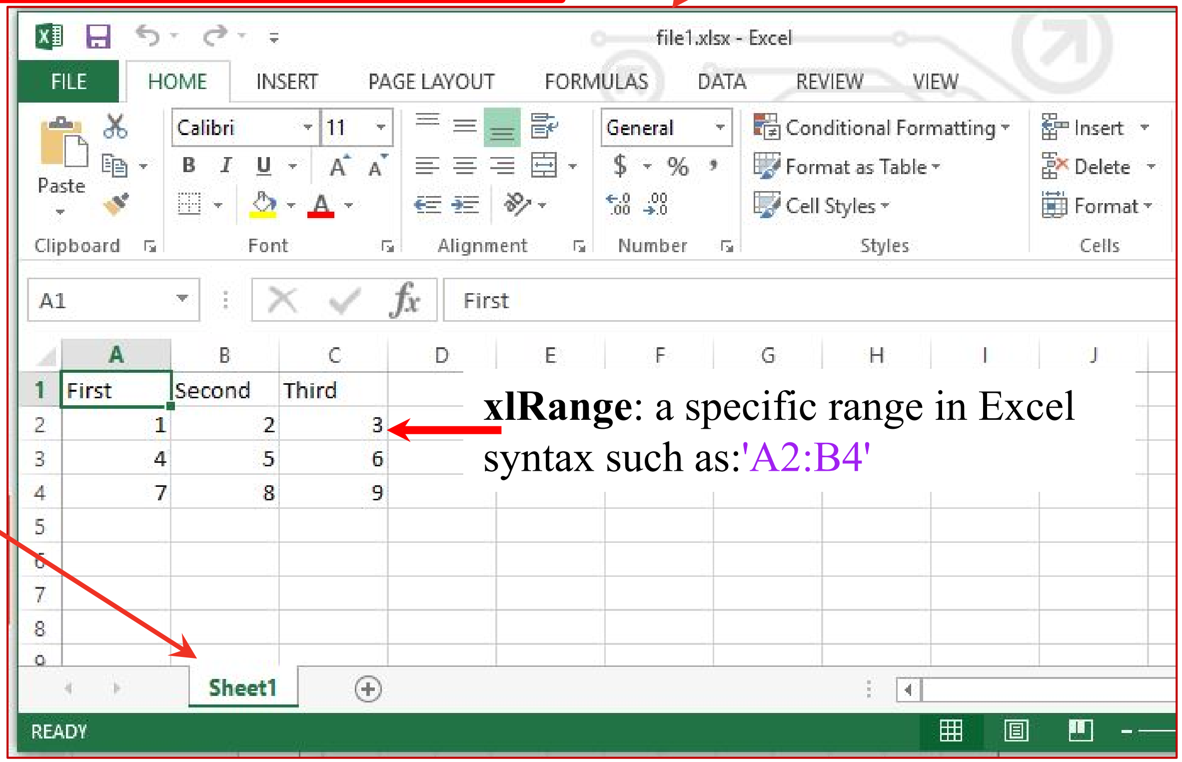
Matlab functions

```
filename = 'file1.xlsx';
sheet='Sheet1';
xlRange = 'A2:B4';
num = xlsread(filename);
num = xlsread(filename,sheet,xlRange);
```

Variables

Filename

Sheet



xlRange: a specific range in Excel syntax such as:'A2:B4'

1. Access and load the data (cont.)

Example:

Workspace

Name	Value
Filename	'load.xlsx'
Input	[1,2;4,5;7,8]
Input1	'A2:B4'
output1	'C2:C4'
Sheetread	'Sheet1'
Target	[3;6;9]

```
clc;
clear;
Filename='load.xlsx';
Sheetread='Sheet1';
Input1='A2:B4';
output1='C2:C4';
Input=xlsread(Filename,Sheetread,Input1); %Read Microsoft Excel
Target=xlsread(Filename,Sheetread,output1 );
x=Input;
t=Target;
```

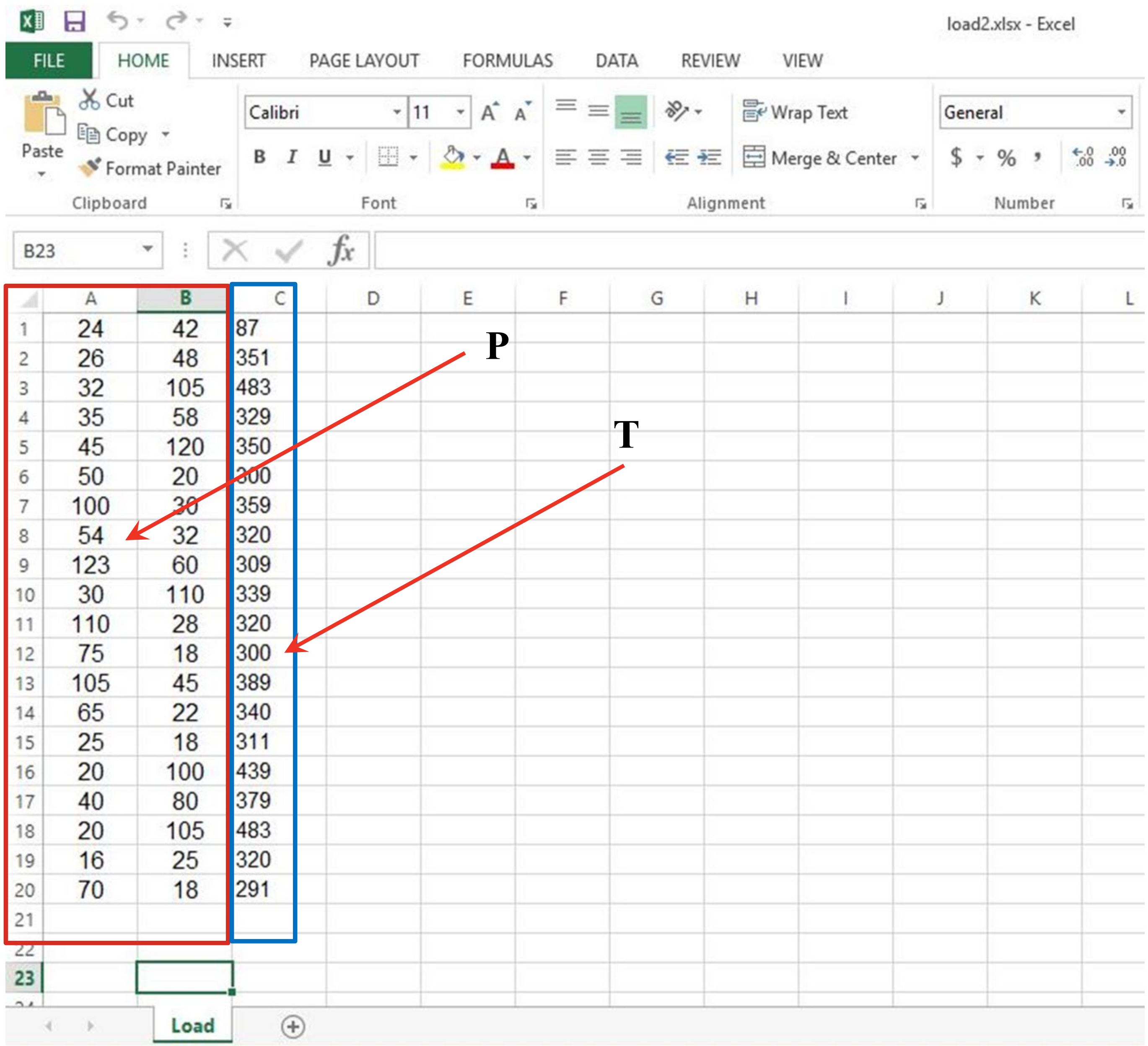
1. Access and load the data (cont.)

Example : Create an Excel file named load2.xlsx with P as inputs and T as an output.

$$P = \begin{pmatrix} 24 & 42 \\ 26 & 48 \\ 32 & 105 \\ 35 & 58 \\ 45 & 120 \\ 50 & 20 \\ 100 & 30 \\ 54 & 32 \\ 123 & 60 \\ 30 & 110 \\ 110 & 28 \\ 75 & 18 \\ 105 & 45 \\ 65 & 22 \\ 25 & 18 \\ 20 & 100 \\ 40 & 80 \\ 20 & 105 \\ 16 & 25 \\ 70 & 18 \end{pmatrix}$$

$$T = \begin{bmatrix} 87 \\ 351 \\ 483 \\ 329 \\ 350 \\ 300 \\ 359 \\ 320 \\ 309 \\ 339 \\ 320 \\ 300 \\ 389 \\ 340 \\ 311 \\ 439 \\ 379 \\ 483 \\ 320 \\ 291 \end{bmatrix}$$

1. Access and load the data (cont.)

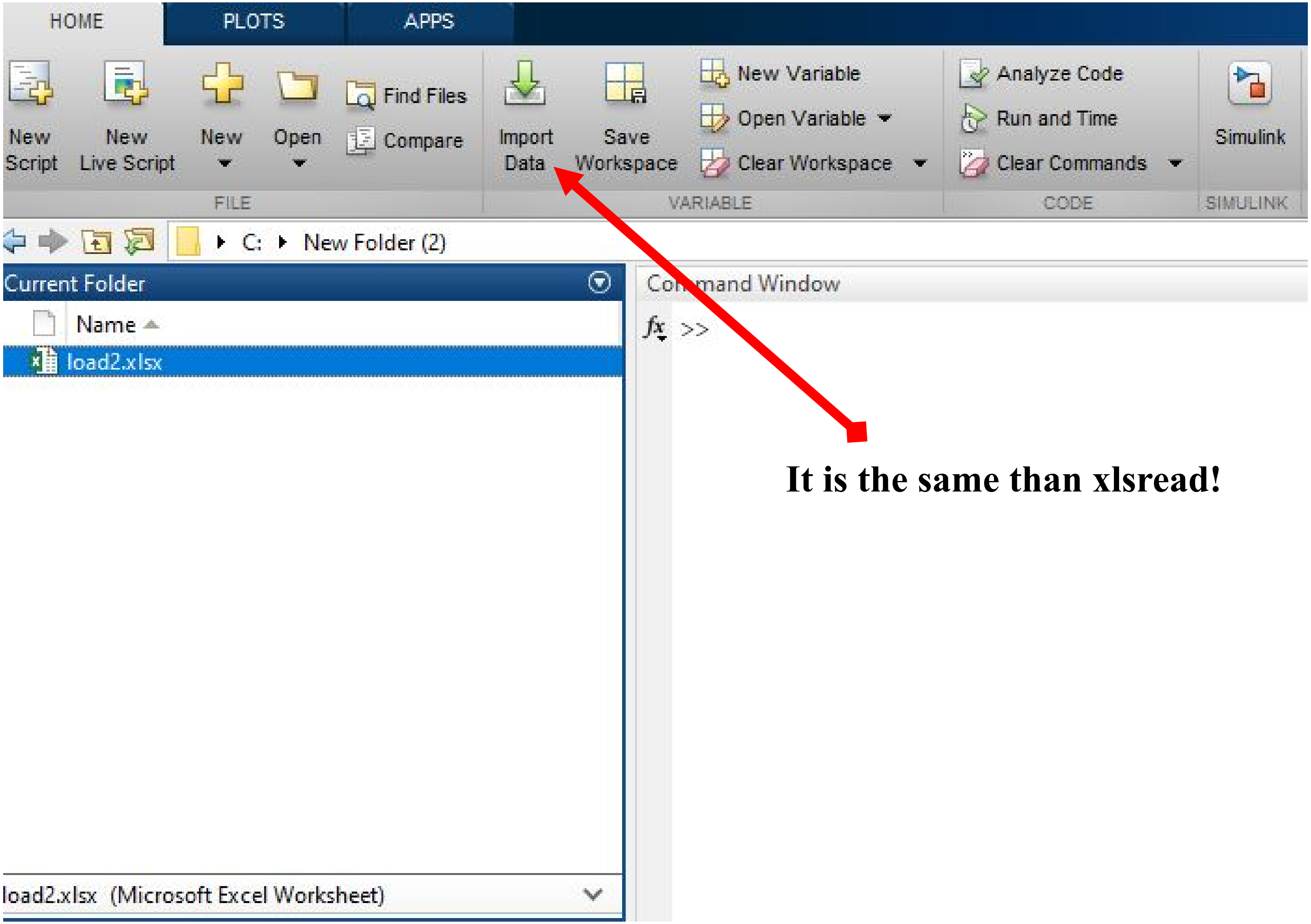


	A	B	C
1	24	42	87
2	26	48	351
3	32	105	483
4	35	58	329
5	45	120	350
6	50	20	300
7	100	30	359
8	54	32	320
9	123	60	309
10	30	110	339
11	110	28	320
12	75	18	300
13	105	45	389
14	65	22	340
15	25	18	311
16	20	100	439
17	40	80	379
18	20	105	483
19	16	25	320
20	70	18	291

1. Access and load the data (cont.)

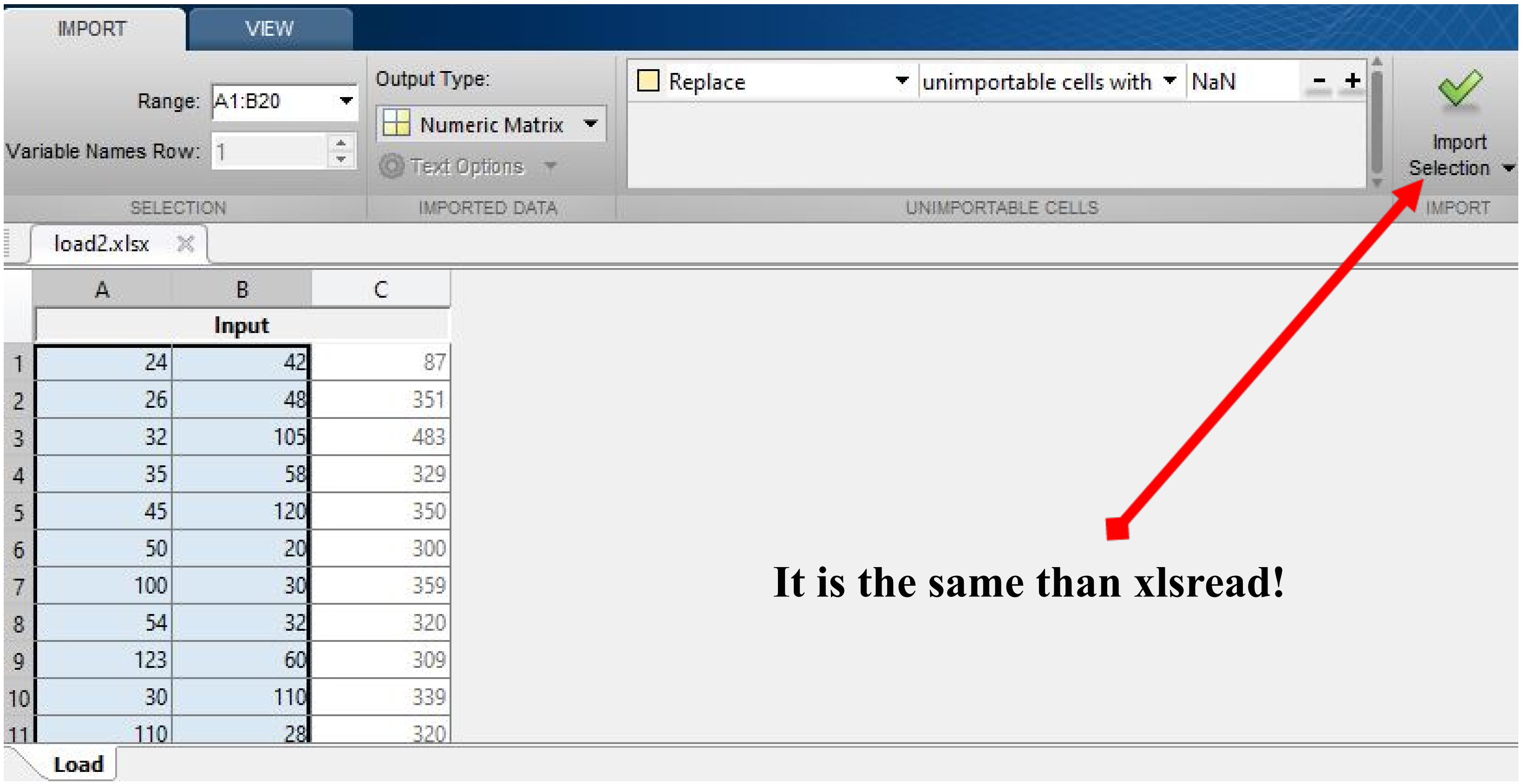
```
clc;
clear;
Filename='load2.xlsx';
Sheetread='load';
Input1='A1:B20';
output1='C1:C20';
x=xlsread(Filename,Sheetread,Input1);
t=xlsread(Filename,Sheetread,output1 );
```

1. Access and load the data (cont.)



Importing Spreadsheet(load2.xlsx) into MATLAB using import data

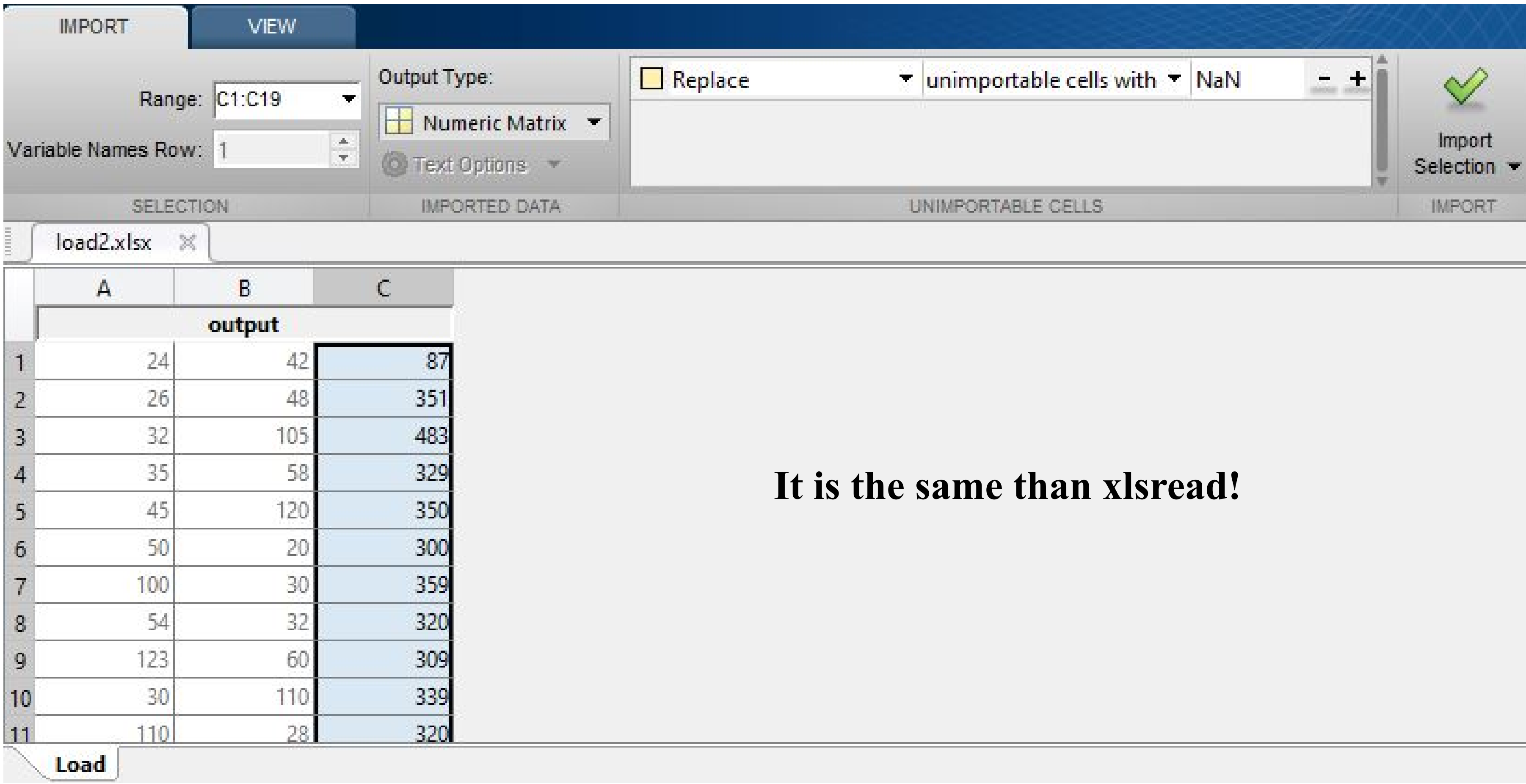
1. Access and load the data (cont.)



It is the same than xlsread!

Importing Spreadsheet(load2.xlsx) into MATLAB using import data

1. Access and load the data (cont.)



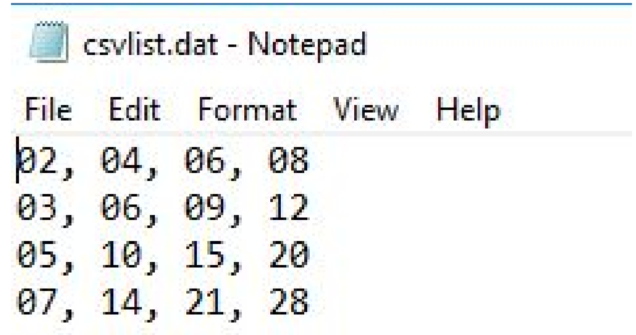
It is the same than xlsread!

Importing Spreadsheet(load2.xlsx) into MATLAB using import data

1. Access and load the data (cont.)

- **Read comma-separated value (CSV) file or text files**

  Syntax:




```
M = csvread(filename);
```

  Example :

csvlist.dat(Create a file named csvlist.dat in notepad that contains following comma-separated values)

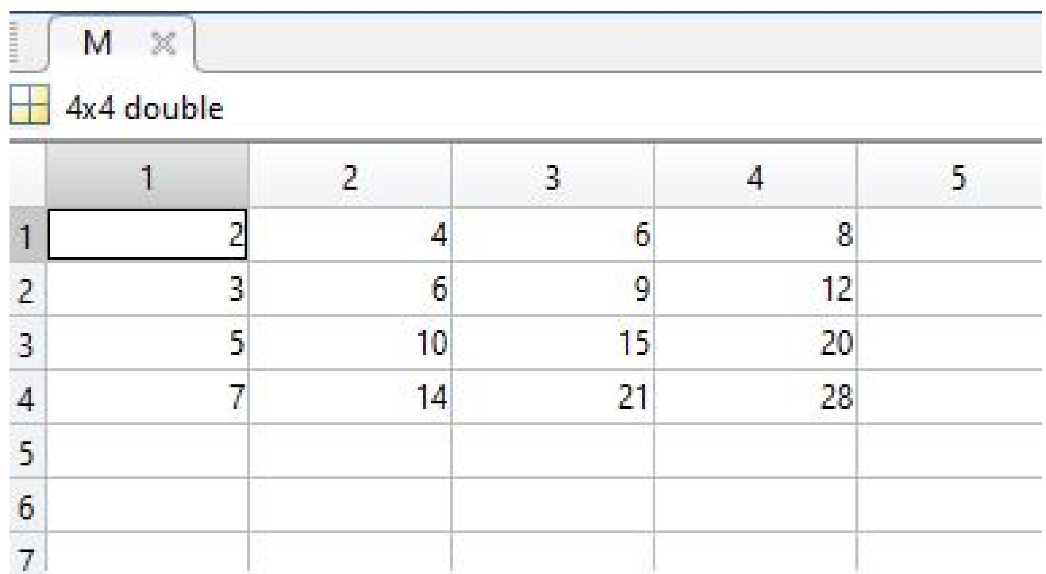
02, 04, 06, 08

03, 06, 09, 12

05, 10, 15, 20

07, 14, 21, 28

```
clear;
clc;
filename ='csvlist.dat';
M = csvread(filename);
```



1. Access and load the data (cont.)

- **Create table from File**

  Syntax:

```
T = readtable(filename);
```

Example:

```
clear;
clc;
filename = 'myCsvTable.dat';
T = readtable(filename);
```

T

5x6 table

	1 LastName	2 Gender	3 Age	4 Height	5 Weight	6 Smoker
1	'Smith'	'M'	38	71	176	1
2	'Johnson'	'M'	43	69	163	0
3	'Williams'	'F'	38	64	131	0
4	'Jones'	'F'	40	67	133	0
5	'Brown'	'F'	49	64	119	0

2. Data pre-processing

- **Find Missing data (NaN,missing)**

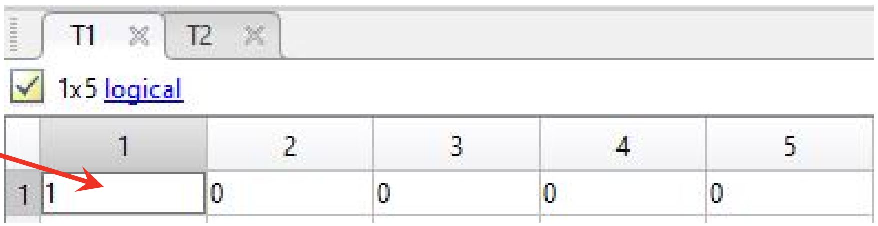
Syntax:

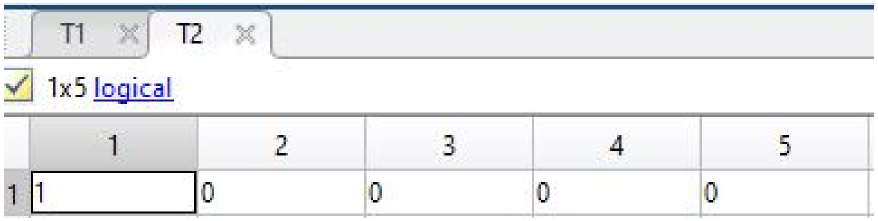
```
T1= isnan(x);
T2= ismissing(x);
```

% x is the input data (e.g. vector,matrix,table)

Example:

```
clear;clc;
x = [NaN 1 2 3 4];          % NaN:Not-a-Number;
T1= isnan(x);
T2= ismissing(x);
```





- **Replace or and Ignore missing Data**

Syntax:

```
xReplace = fillmissing(x,method);   % Replace
xRemove = rmmissing(x);             % Remove
```

Method:

'previous'   previous non-missing value
'next'   next non-missing value
'nearest'   nearest non-missing value
'linear'   linear interpolation of neighbouring
'spline'   piecewise cubic spline interpolation

2. Data pre-processing (cont.)

Example :

```
clear;clc;
x =[1 2 3 ;5 6 7;NaN NaN 2];
xFill= fillmissing(x,'previous');
xRemove = rmmissing(x);
```

xFill

3x3 double

	1	2	3
1	1	2	3
2	5	6	7
3	5	6	2

xFill | xRemove

2x3 double

	1	2	3
1	1	2	3
2	5	6	7

- **Find and replacing outliers**

Syntax:

```
xoutlier = isoutlier(x);
```

% x is the input data (e.g. vector,matrix,table)

Example:

```
clear;clc;
x = [57 59 60 100 59 58 57 58 300];
A = isoutlier(x);
```

A

1x9 logical

	1	2	3	4	5	6	7	8	9
1	0	0	0	1	0	0	0	0	1
2									
3									
4									

2. Data pre-processing (cont.)

- **Replace or and Ignore outliers**

Syntax:

$\boxed{\text{B = filloutliers(x,method);}}$

Method:

'previous'   previous non-missing value

'next'   next non-missing value

'nearest'   nearest non-missing value

'linear'   linear interpolation of neighbouring

'spline'   piecewise cubic spline interpolation

Example :

```
clear; clc;
x = [57 59 60 100 59 58 57 58 300];
B = filloutliers(x, 'previous');
```

B

1x9 double

	1	2	3	4	5	6	7	8	9
1	57	59	60	60	59	58	57	58	58
2									
3									

2. Data pre-processing (cont.)

- **Data normalization and standardization**
  - To change the range of the values to be between a specific range (e.g. range [-1,1] or [0,1]) .
  - To avoid the values with large values to dominate the results.
  - To equalize the contribution of all the inputs.

– **Data normalization:**

x_new = (x - x_min)/(x_max - x_min)

– **Description:**

x is the input data; x_min is the minimum of x; x_max, the maximum of x.

```
x_min = min(x)        % To obtain the minimum of x
x_max = max(x)        % To obtain the maximum of x
```

– **Data standardization:**

x_new = (x - μ)/σ

– **Description:**

x is the input data; **μ** is the mean(average) ; **σ** is the standard deviation.

```
μ = mean(x)        % To obtain the mean value
σ = std(x)         % To obtain the standard deviation
```

2. Data pre-processing (cont.)

- **Data standardization in Matlab:**

Syntax:

$\boxed{\text{[Y,PS] = mapminmax(X,YMIN,YMAX);}}$

Description:

Process matrices by mapping row minimum and maximum values to [-YMIN YMAX]

X: is the input.

YMIN : is the minimum value (Default: -1).

YMAX : is the maximum value (Default :1).

Y :is the output.

PS: is the process setting.

x

4x3 double

	1	2	3
1	1	2	4
2	1	1	1
3	3	2	2
4	0	0	0

pn

4x3 double

	1	2	3
1	-1	-0.3333	1
2	1	1	1
3	1	-1	-1
4	0	0	0

Example :

```
clear; clc;
x = [1 2 4; 1 1 1; 3 2 2; 0 0 0];
[pn,ps] = mapminmax(x);
x_again = mapminmax('reverse', pn,ps); % To reverse
```

2. Data pre-processing (cont.)

- **Data normalization in Matlab:**

Syntax:

$\boxed{\text{[Y,PS] = mapstd(X,ymean,ystd) ;}}$

Description:

Process matrices by mapping each row's means to 0 and deviations to 1.

X: is the input.

ymean: is the mean value for each row of Y(Default: 0).

ystd : is the standard deviation for each row of Y(Default :1).

Y :is the output.

PS: is the process setting.

X

4x3 double

	1	2	3
1	3	2	4
2	9	1	1
3	5	2	2
4	0	1	0

Y

4x3 double

	1	2	3
1	0	-1	1
2	1.1547	-0.5774	-0.5774
3	1.1547	-0.5774	-0.5774
4	-0.5774	1.1547	-0.5774

Example:

```
clear;clc;
X = [3 2 4; 9 1 1; 5 2 2; 0 1 0];
[Y,PS] = mapstd(X);
X_again = mapstd('reverse',Y,PS); % To reverse
```

3. Feature selection and Dimensionality reduction

- Removing redundant or irrelevant features(inputs)
- Combining features
- Creating new features

Common technique

- Principal component analysis (PCA) (outside course scope)

Description:

Detects linear dependencies between variables and replaces groups of correlated variables by new uncorrelated variables, the principal components (PCs).

The Steps of an application of Artificial Neural Network (ANN)

I. **Data pre-processing**

```
[Y,PS] = mapstd(X,ymean,ystd) ;
```

II. **Selecting network architecture**

```
net = feedforwardnet();
```

III. **Network training**

```
[net,tr]= train(net,inputs,targets);
```

IV. **Simulation (validation)**

```
a =net (inputs);
```

V. **Performance(Post-processing)**

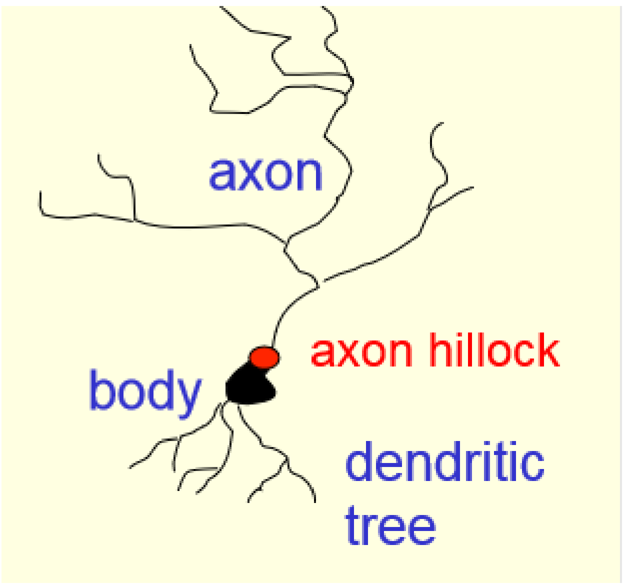
MSE: Mean squared error performance function.

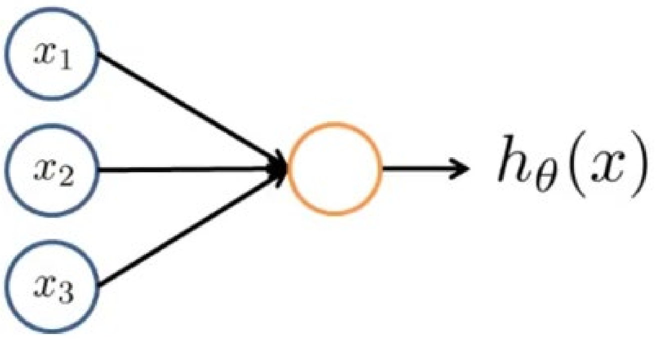
RMSE: Root Mean squared error performance function

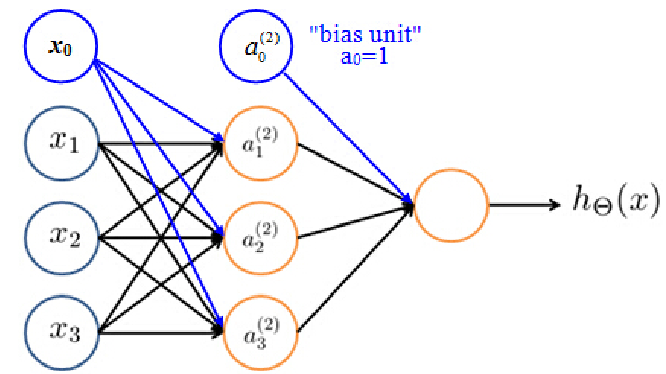
R: Coefficient of correlation.

R^2: Coefficient of determination (R-squared).

4. Build and train the models (ANN)







4. Build and train the models (ANN) (cont.)

- **Building neural network**

Syntax:

```
net = patternnet(hiddenLayerSize);
net = fitnet(hiddenLayerSize);
net = feedforwardnet (hiddenLayerSize);
```

Description :

hiddenLayerSize: number of neurons in the hidden layer .

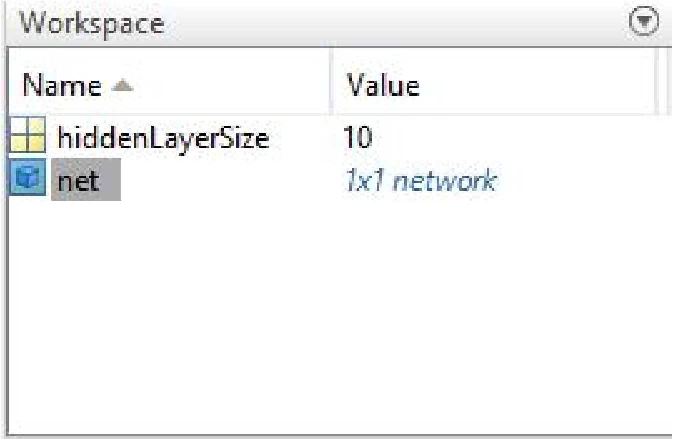
fitnet: regression and curve-fitting.

patternnet: classification and pattern-recognition.

feedforwardnet: is the generalized form of fitnet and patternnet.

Example:

```
clear;clc;
hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize);
```



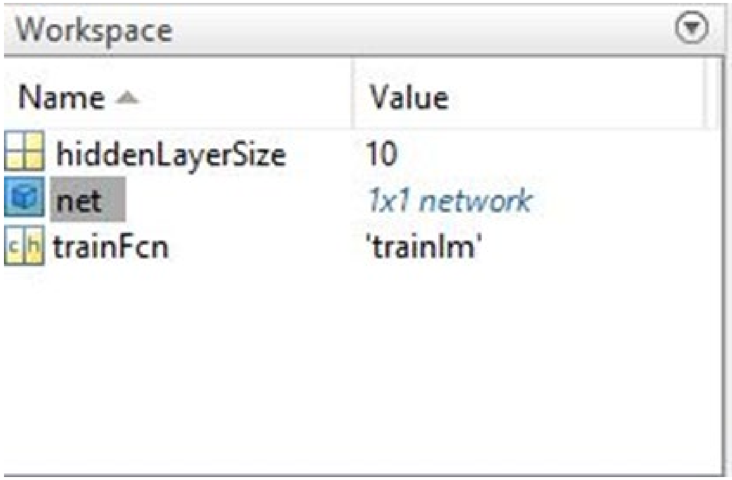
4. Build and train the models (ANN) (cont.)

- **Choose a Training Function :**
  - **'trainlm'** Levenberg-Marquardt backpropagation (fastest algorithm and the most common method).
  - **'trainbr'** Bayesian Regulation backpropagation.
  - **'trainscg'** Scaled conjugate gradient backpropagation (better in classification problems).
  - **'traingd'** Gradient descent backpropagation.

Example:

```
trainFcn = 'trainlm';
```

For a list of all training functions type: help nntrain



Example:

```
clear;clc;
hiddenLayerSize = 10;
trainFcn = 'trainlm';
net = feedforwardnet(hiddenLayerSize, trainFcn);
```

4. Build and train the models (ANN) (cont.)

- **Setup Division of Data for Training, Validation, Testing**

  dividerand : Partition indices into three sets using random indices.

  divideblock: Partition indices into three sets using blocks of indices.

Example:

```
net.divideFcn = 'dividerand';
net.divideFcn = 'divideblock';
```

A list of all data division functions type: $\boxed{\text{help nndivision}}$

net.divideMode : defines the target data dimensions which to divide up when the data division function is called.

```
net.divideMode = 'sample';   % Static networks
net.divideMode = 'time';     % Dynamic networks
```

Example:

```
net.divideMode = 'sample';  % Divide up every sample
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
```

4. Build and train the models (ANN) (cont.)

- **Choose plot Functions:**

  ploterrhist : plot error histogram

  plotperform : plot network performance

  plottrainstate : plot training state values

  plotregression: plot linear regression

Example:

```
net.plotFcns = {'plotperform','plottrainstate','ploterrhist', 'plotregression'};
plotperform();
plottrainstate();
ploterrhist();
plotregression ();
```

4. Build and train the models (ANN) (cont.)

- **Choose Activation functions**

  Linear: 'purelin'

  Sigmoid or Logistic: 'logsig'

  Tanh(Hyperbolic tangent): 'tansig'

A list of all transfer(activation) functions type: help nntransfer

❖ **Hidden layer activation function**

Sigmoid, Tanh and Relu (Deep learning)

❖ **Output layer activation function**

Regression: linear

Classification: softmax % simple sigmoid works too but softmax works better

Example:

```
net.layers{1}.transferFcn = 'logsig';        % for hidden layer
net.layers{2}.transferFcn = 'purelin' ;      % for output layer
```

4. Build and train the models (ANN) (cont.)

- **Choosing number and size of the layers**

Example:

```
net.numLayers = 3;            % number of layers in the network
net.layers{1}.size = 8;       % number of neurons
```

- **Choose training parameters**

```
net.trainParam.mu=0.005       % Marquardt adjustment parameter (trainlm and trainbr)
net.trainParam.lr=0.05        % learning rate (traingd)
net.trainParam.epochs=1000    % max epochs
net.trainParam.goal=1e-5      % minimum performance value
net.trainParam.time=60        % maximum training time in seconds
```

- **Choose a Performance Function**

MSE : Mean squared error performance function.

SSE: Sum squared error performance function.

Example:

```
net.performFcn = 'mse';
```

A list of all performance functions type: help nnperformance

4. Build and train the models (ANN) (cont.)

- **Example of different parameter settings in ANN**

```
clear;clc;
trainFcn = 'trainlm'; hiddenLayerSize = 10;
net = fitnet(hiddenLayerSize,trainFcn); %fitnet for regression
net.divideFcn = 'dividerand'; %data division
net.divideMode = 'sample';     %static network
net.divideParam.trainRatio = 70/100; % Divide up every sample
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
net.plotFcns = {'plotperform','plottrainstate','ploterrhist'};
net.layers{1}.transferFcn = 'logsig';  %for hidden layer
net.layers{2}.transferFcn = 'purelin'; %for output layer
```

4. Build and train the models (ANN) (cont.)

```
net.trainParam.mu=0.005         % Marquardt adjustment parameter
net.trainParam.epochs=1000      % max epochs
net.trainParam.max_fail=6;      % Maximum validation failure
net.trainParam.goal=1e-5        % minimum performance value
net.trainParam.time=60          % maximum training time in seconds
net.performFcn = 'mse';         % Performance function
```

4. Build and train the models (ANN)

- **Train:** train neural network

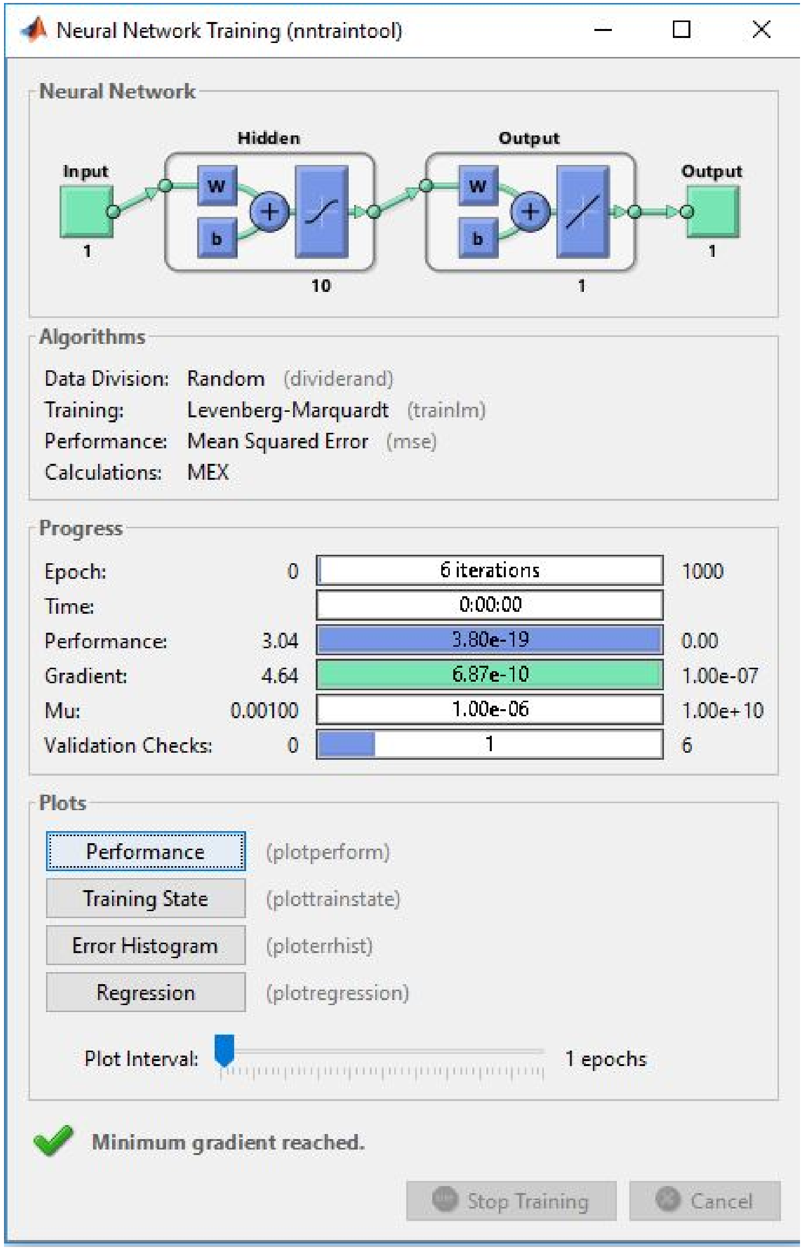
Syntax :

[net,tr] = train(net,x,t); % net: newly trained network; % tr: training record

Example :   **% The answers may vary due to randomness%**

```
clear;
clc;
x = [0 1 2 3 4 5 6 7 8];
t = [0 0.84 0.91 0.14 -0.77 -0.96 -0.28 0.66 0.99];
net = feedforwardnet(10);
net = train(net,x,t);
```

4. Build and train the models (ANN) (cont.)



5. Validation and performance (ANN)

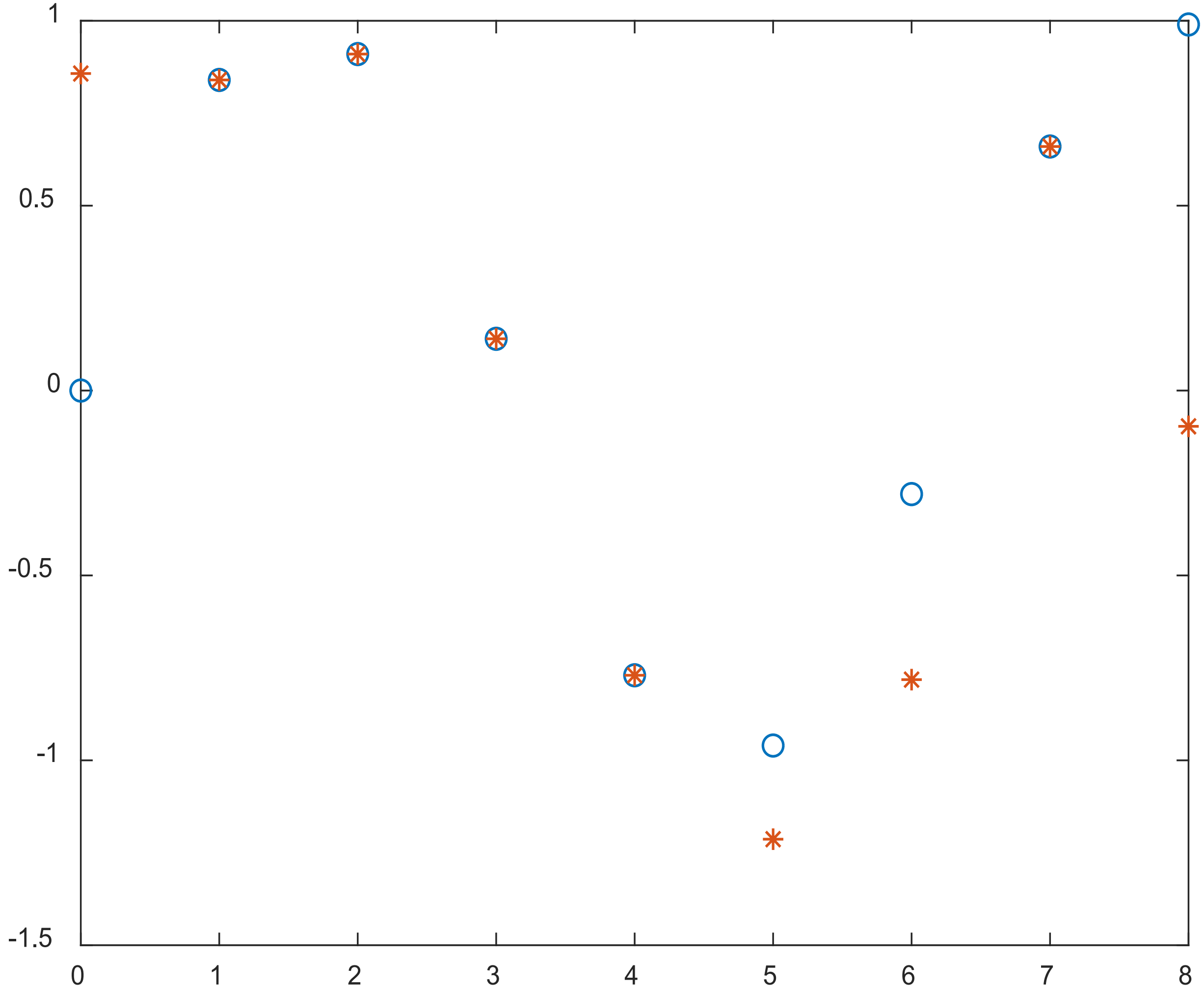
- **Test the network (validation)**

Syntax:

y=net(x); % x is the input , y is the predicted output based on input x.

Example:

```
clear;clc;
x = [0 1 2 3 4 5 6 7 8];
t = [0 0.84 0.91 0.14 -0.77 -0.96 -0.28 0.66 0.99];
plot(x,t,'o');                    % plot data
net = feedforwardnet(10);         % Build the network
net.divideParam.trainRatio = 60/100; % Divide the data
net.divideParam.valRatio = 20/100;
net.divideParam.testRatio = 20/100;
RandStream.setGlobalStream (RandStream ('mrg32k3a'));
% Just for the sake of reproducing the same results in the slides.
net = train(net,x,t);            % Train the data
y = net(x);                      % Validation
plot(x,t,'o',x,y, '*');          % plot t(actual output) versus y
(predicted output)
```

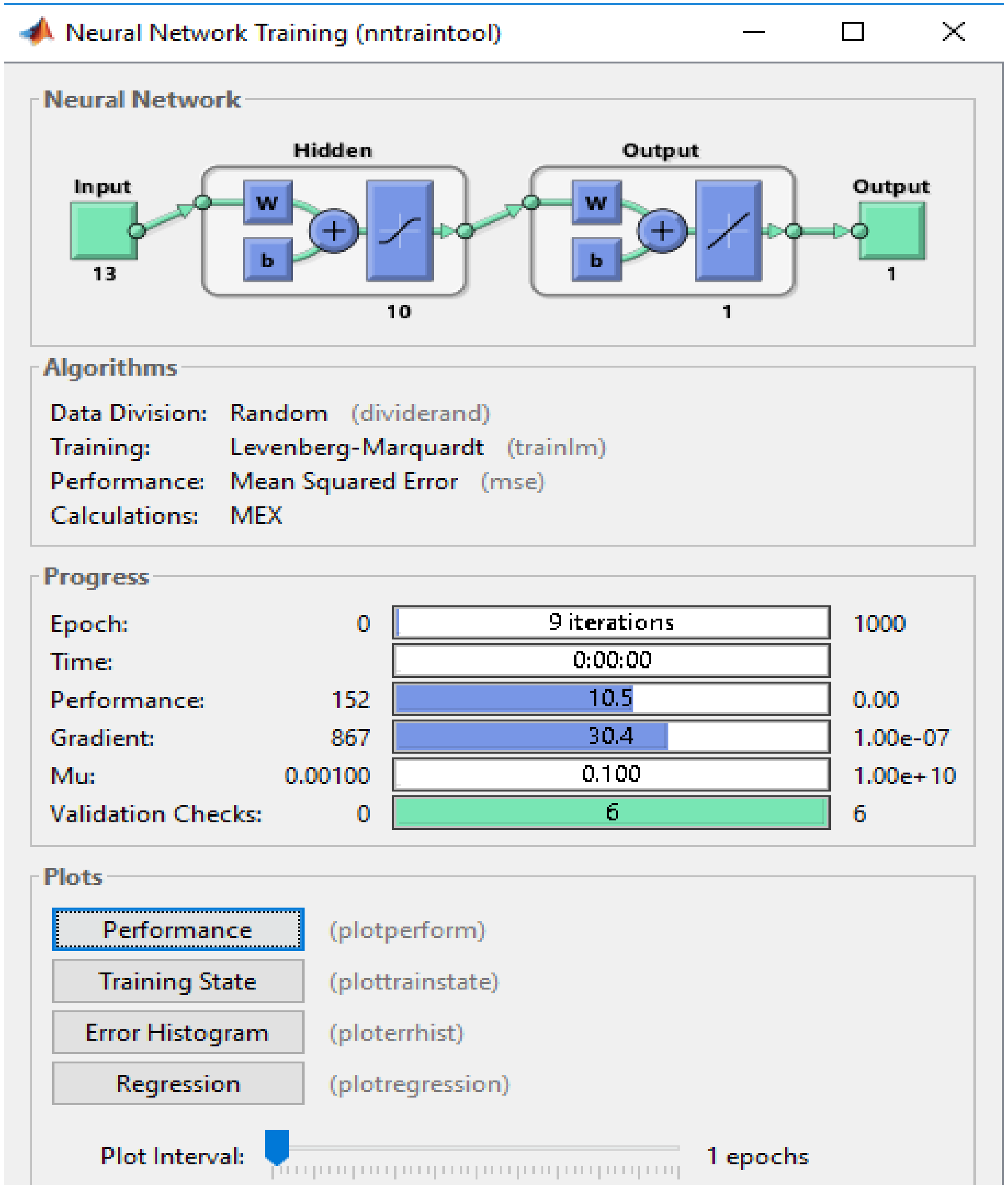


t (actual out put shown with o) versus y (predicted output shown with *)

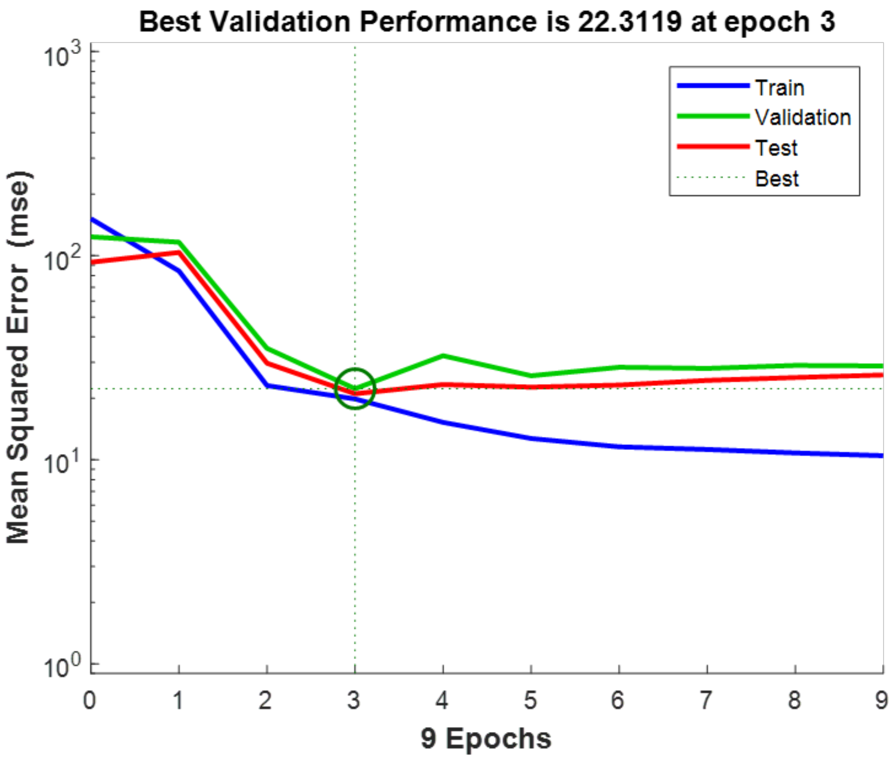
5. Validation and performance (ANN) (cont.)

Example1: `% Example1.xlsx is imported %`

```
clear;
clc;
load x.mat;
load t.mat;
x=x';                                    % Transpose Input
t=t';                                    % Transpose Output
net = feedforwardnet(10);
%net.divideFcn = 'dividerand';           % Divide data randomly
RandStream.setGlobalStream (RandStream ('mrg32k3a'));
[net,tr] = train(net,x,t);               % train the network
figure, plotperform(tr);                 % performance plot
figure, plottrainstate(tr);              % training state plot
y=net(x);                                % Test the network
```

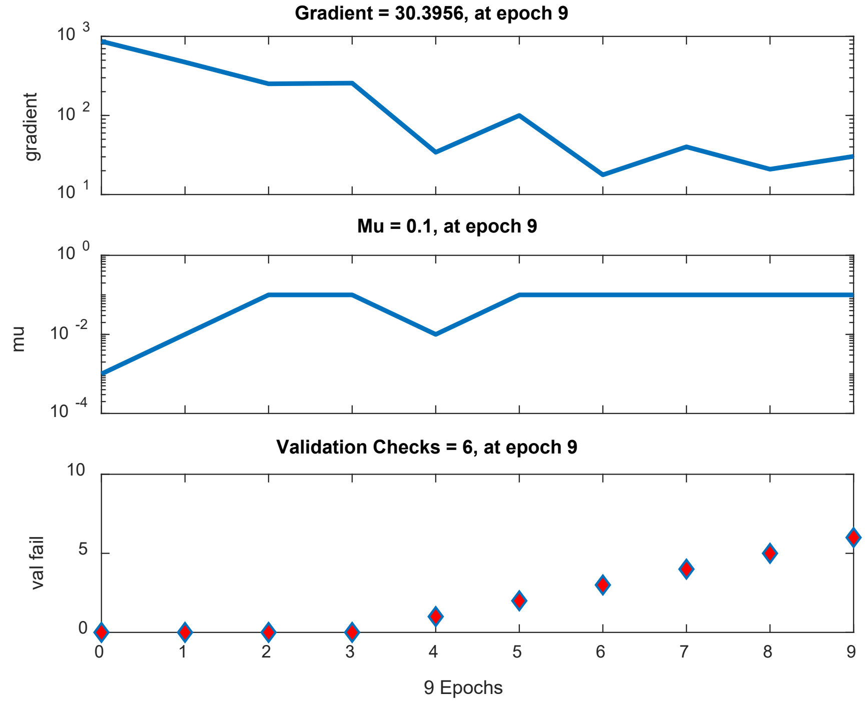


Neural Network Training



Plotperform

Plottrainstate



5. Validation and performance (ANN) (cont.)

- MSE (Mean of squared error)= $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- RMSE (Root Mean squared error) = $\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$

  y_i: Actual outputs

  $\hat{y}_i$: Predicted outputs

  n: Number of observations

- **Calculate network performance**

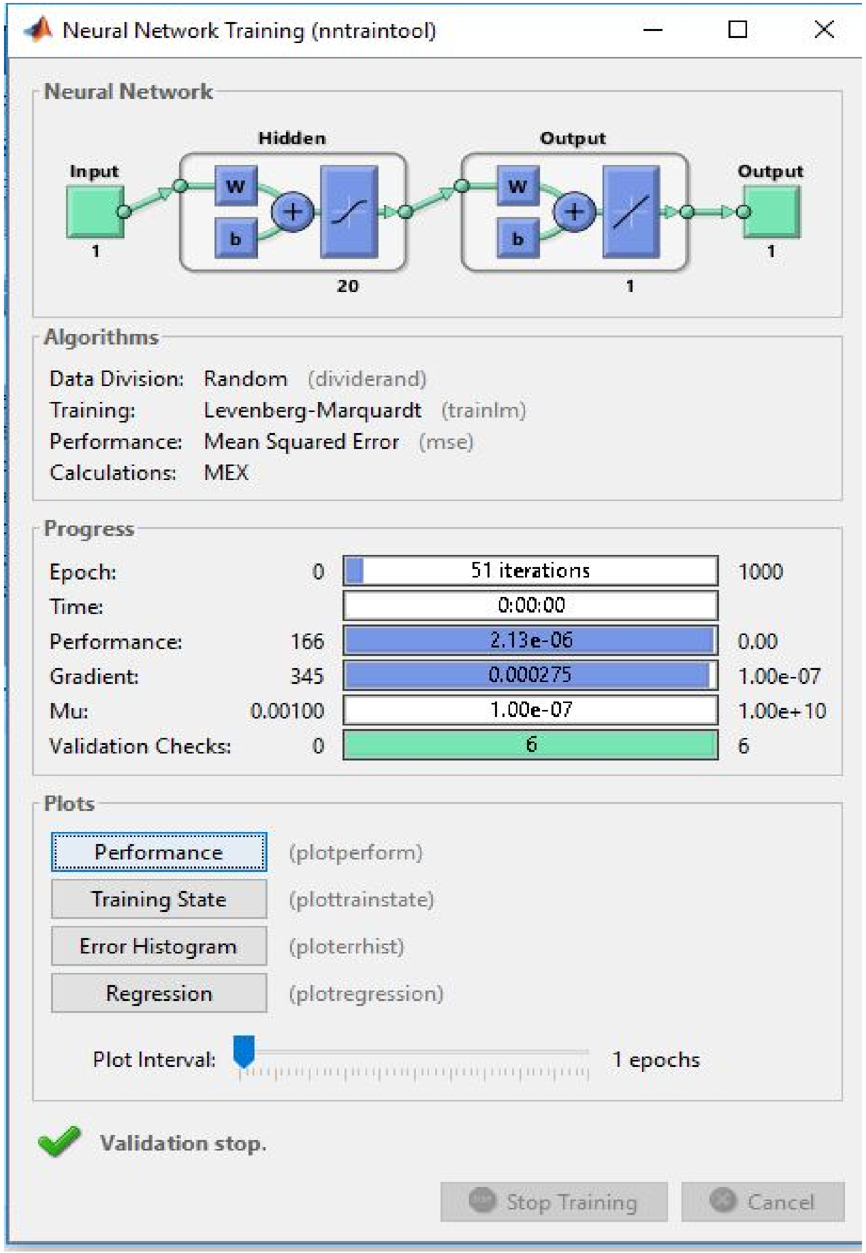
Syntax:

```
perf = perform(net,t,y);          % t is the output y is the predicted output.
perf1 = mse(net, t, y);           % Same result as perform
```

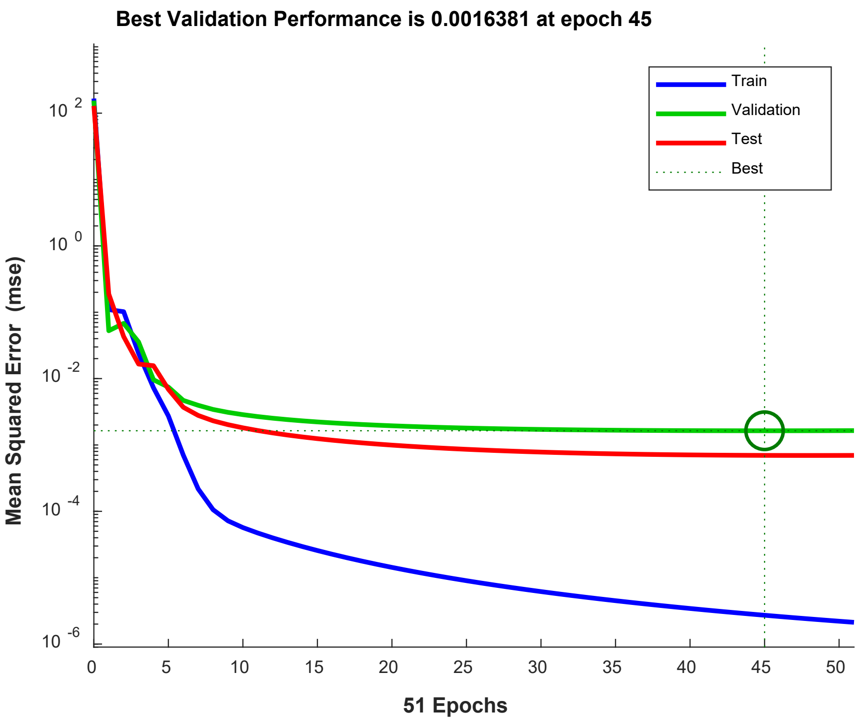
5. Validation and performance (ANN) (cont.)

Example2: % Example2.xlsx is imported %

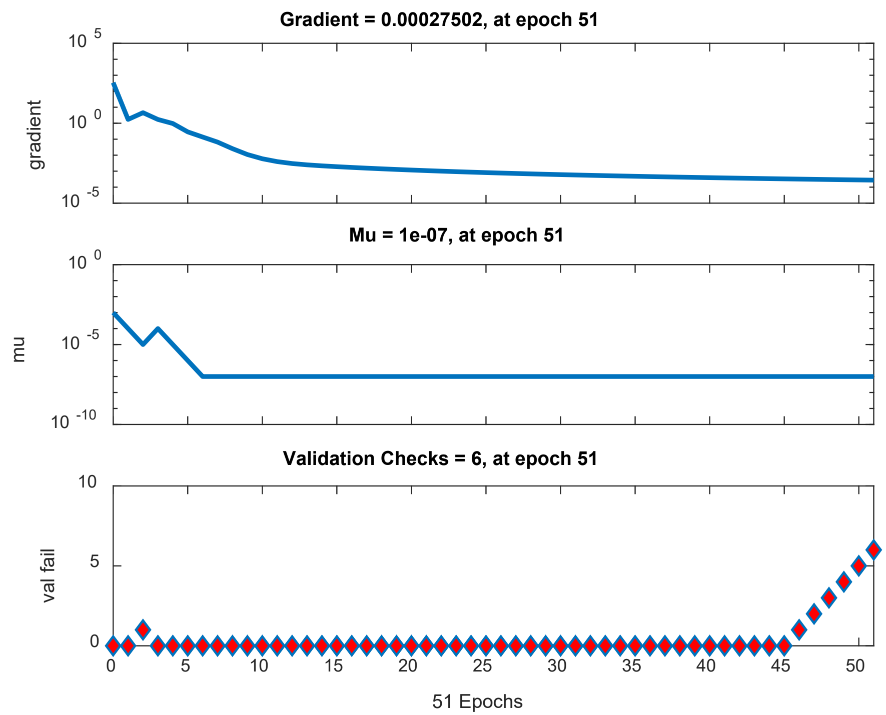
```
clear;
clc;
load x.mat;
load t.mat;
x=x';                        % Transpose Input
t=t';                        % Transpose Output
plot(x,t,'o');
net = feedforwardnet(20); % Build the network with 20 neurons
net.performFcn;
% Shows that MSE is the default network performance function
RandStream.setGlobalStream (RandStream ('mrg32k3a'));
net = train(net,x,t);        % Train the network
y = net(x);                  % Validation
perf = perform(net,t,y);   % perf1 = mse(net, t, y)  % performance
plot(x,t,'o',x,y, '*');    % o for x versus t , * for x versus y
```



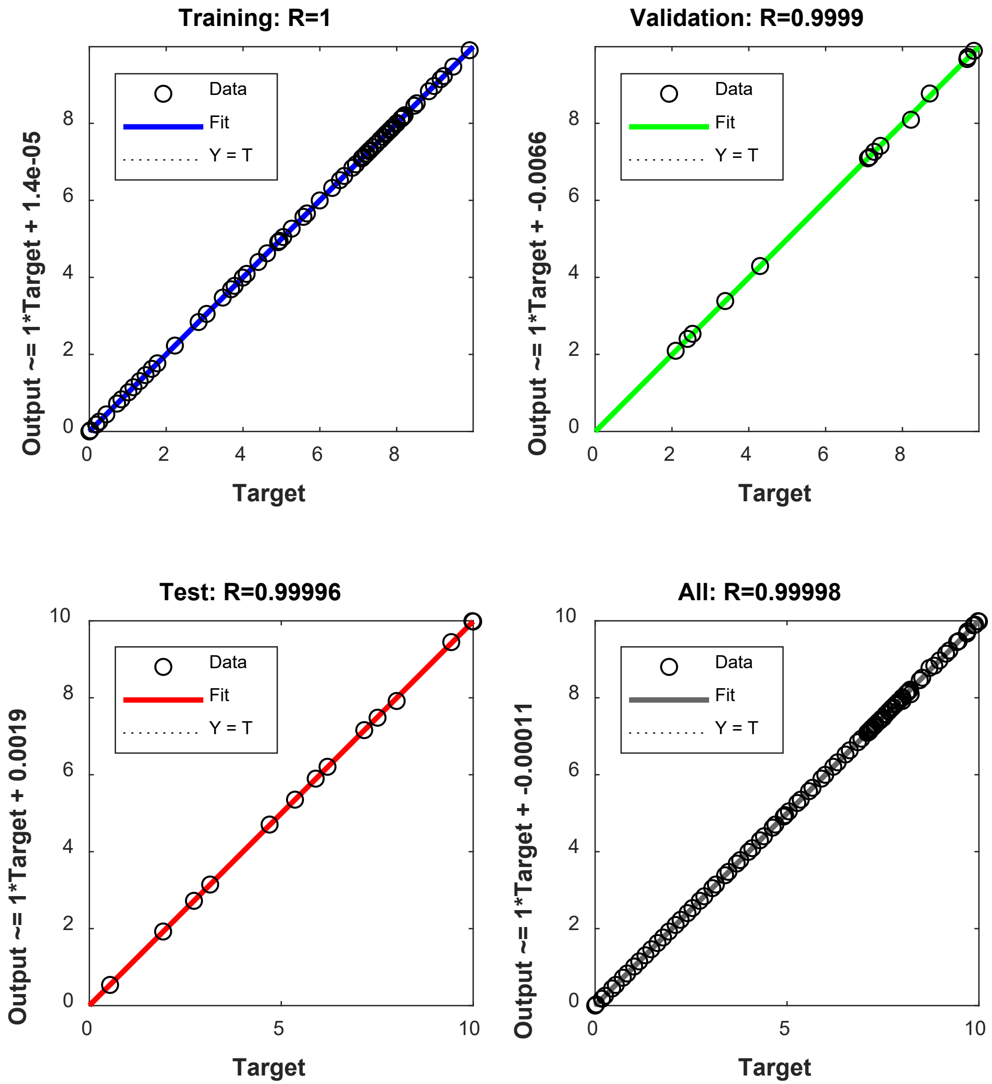
Neural Network Training



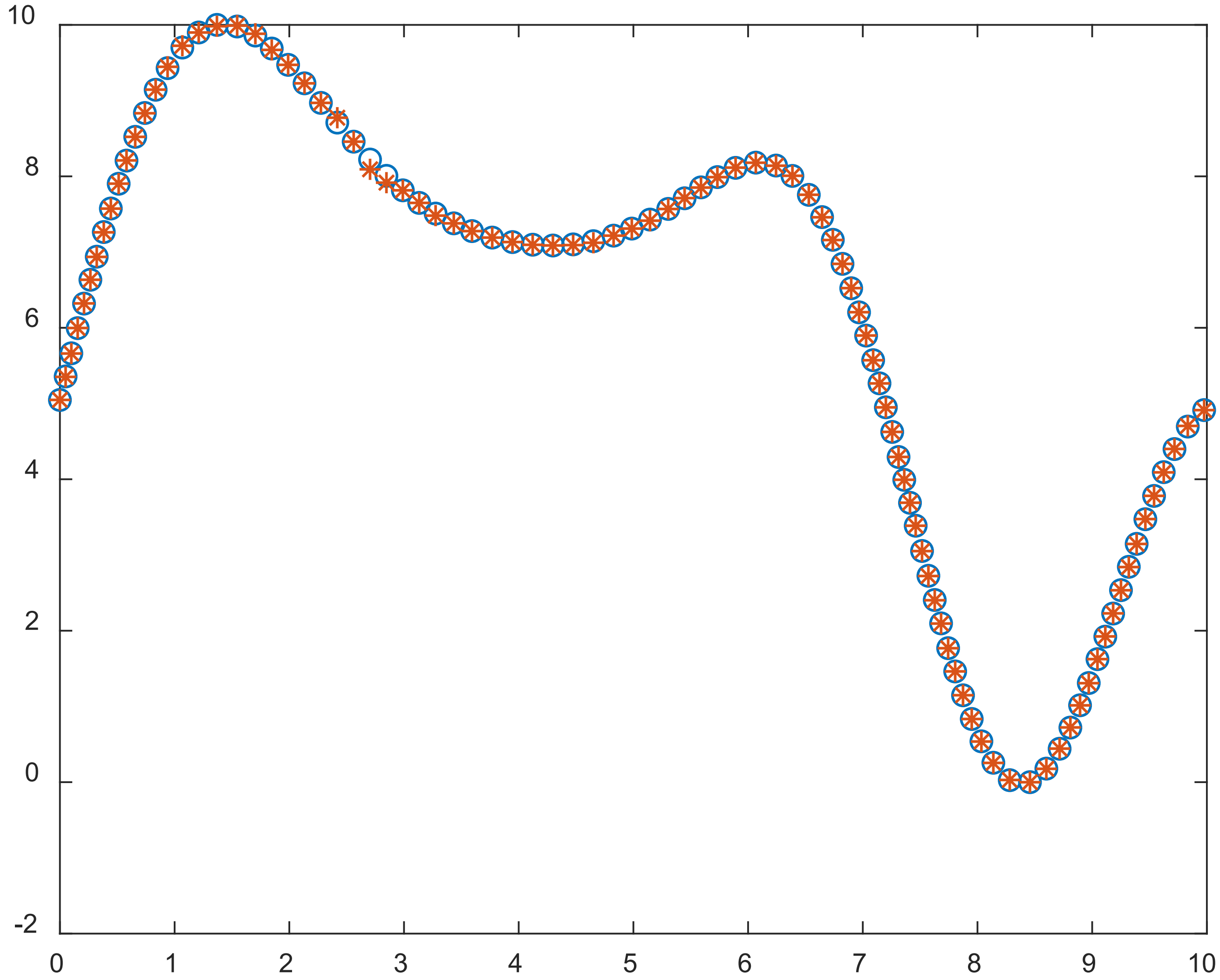
Plotperform



Plottrainstate



Plotregression



t (actual output shown with o) versus y (predicted output shown with *)