About the course

We are always trying to improve the course so would be very happy to hear any suggestions. The course has been run by using new method Project Based Learning (PBL). We remind you that:

(1) The course is run in its current configuration for the first time; the number of contact face-to-face hours has been dramatically increased. With the minimum required 12x3=36 hrs, we actually delivered more hours than this minimum: 4x3 + 8x4 = 12+32 = 44 hrs.

(2) Our qualified team has designed a modern and relevant context for this Course.

About the course

(3) We have presented universal techniques to solve variety of problems; at the same time representative examples were provided from a range of engineering applications (mechanical, automotive, electrical, environmental engineering, etc.).

(4) We have introduced computer-based hand-on tutorials and involved the team of best expert tutors.

(5) We worked hard to take into account student's feedback and listen to the students suggestions.

(6) We value their feedback and CES is important to us.

(7) Please if you have any questions and issues contact your tutorial directly and cc Hamid (hamid.khayyam@rmit.edu.au) as well.

CES

10 Minutes to complete your feedback and CES please!!

Machine Learning Practical Revision (ANN, SVM, NLR)

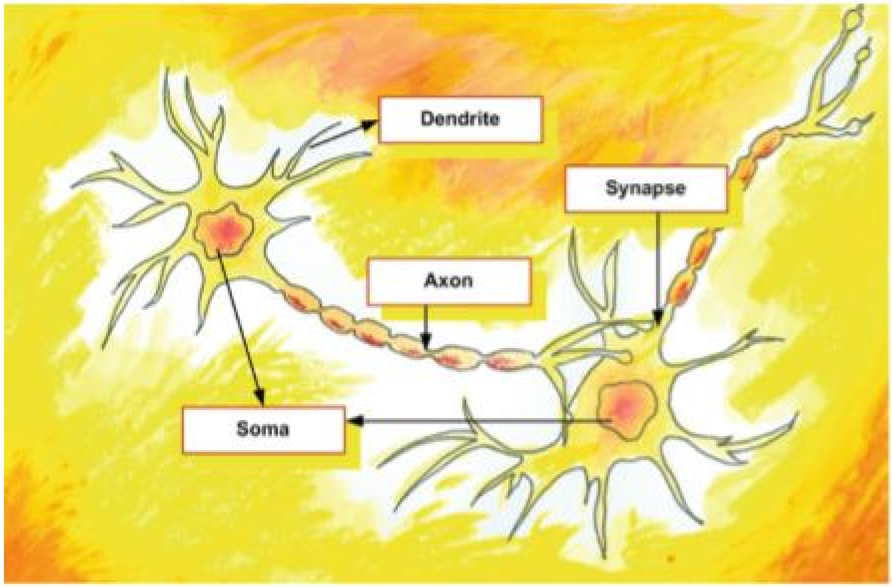
Lecturer:

Dr Hamid Khayyam (Australia)

Email: hamid.khayyam@rmit.edu.au

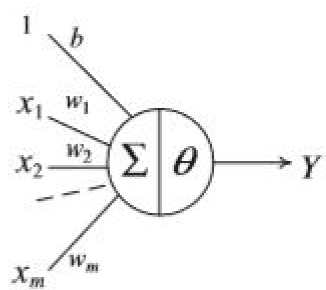
RMIT UNIVERSITY

Artificial Neural Network (ANN)



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

The Steps of an application of ANN

1. **Data pre-processing (check for missing data ,standardization)**

2. **Selecting network architecture**

   net = feedforwardnet();

3. **Network training**

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

4. **Simulation (validation)**

   a =net (inputs);

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

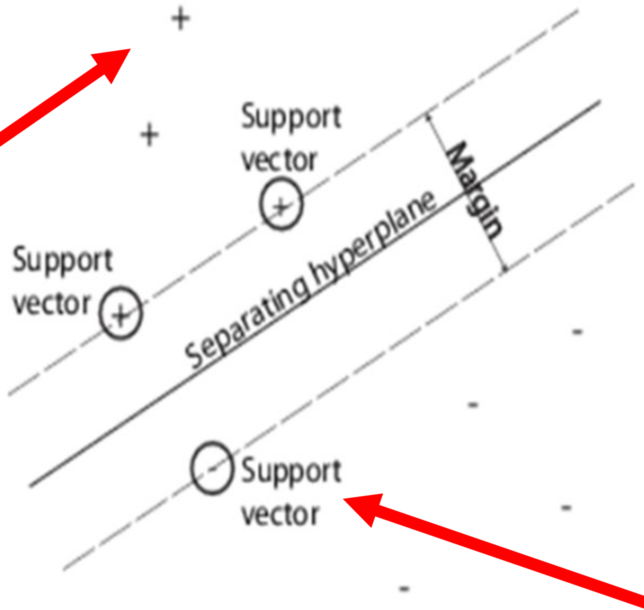
   MSE: Mean squared error performance function.

   RMSE: Root Mean squared error performance function

   R: Coefficient of correlation.

Support Vector Machine (SVM):

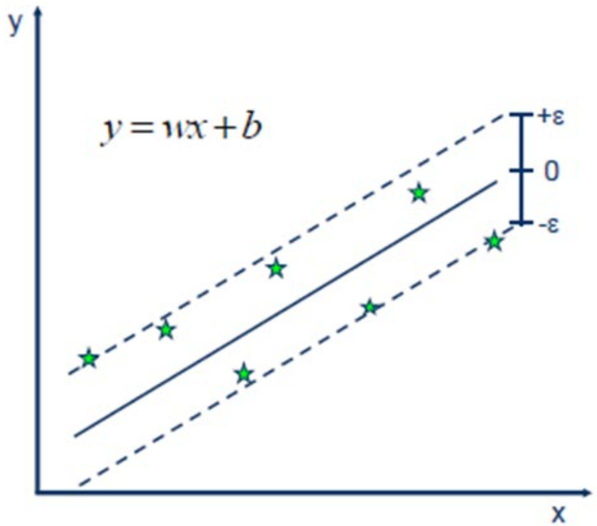
- **Classification in SVM**



+ : data points of type 1
– : data points of type –1

The data points nearest to the hyperplane

- **Regression in SVM (SVR)**



- Solution:

$$\min \frac{1}{2}\|w\|^2$$

- Constraints:

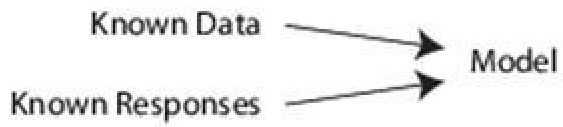
$$y_i - wx_i - b \le \varepsilon$$
$$wx_i + b - y_i \le \varepsilon$$

ε: Margin of tolerance

Steps of An Application of SVR

1. Data pre-processing (check for missing data ,standardization)

2. Model development and training



Mdl = fitrsvm(X, Y) (Regression)

3. Simulation (prediction)

Y_predicted= predict(Mdl,X) (Regression) %Y_predicted is the predicted responses

4. Post-processing

- MSE,RMSE,R (Regression)

Nonlinear Regression (NLR):

A term used for all a wide range of regression models which all present a nonlinear relationship between input and output .

Model	Equation
Exponential	$y = ae^{bx} + ce^{dx}$
Fourier	$y = a_0 + a_1 \cos(x * w) + b_1 \sin(x * w) + \cdots a_n \cos(x * w) + b_n \sin(x * w)$
Gaussian	$y = a_1 e^{[(x-b_1)/c_1]^2} + \cdots + a_n e^{[(x-b_n)/c_n]^2}$
Polynomial	$y = a_1 x^n + a_2 x^{n-1} + \cdots + a_n x + a_0$
Power	$y = ax^b + c$
Sin function	$y = a_1 \sin(b_1 x + c_1) + \cdots + a_n \sin(b_n x + c_n)$
Weibull	$y = abx^{b-1} e^{\left(-a * x^b\right)}$

Conventional nonlinear regression functions

Steps of An Application of NLR

I. Data pre-processing (check for missing data ,standardization)

II. Define the model function and coefficients initiation

III. Model development and training

```
mdl = fitnlm(x,t,fun,beta);
```

IV. Simulation (prediction)

```
Y_predicted= predict(Mdl,X) ;
```

V. Post-processing

MSE, RMSE, R

Example of ANN (1) :

```
%prediction of Torque based on fuel rate and speed
clear;clc;
Filename='DataEngine.xlsx';
Sheetread='Training';
Input1='A1:B1194';
output1='C1:C1194';
Input=xlsread(Filename,Sheetread,Input1);
Target=xlsread(Filename,Sheetread,output1 );
x=Input;
t=Target;
```

- `Sheetread1='Newdata';`
- `Input2='A1:B3';`
- `Target2 ='C1:C3';`

```
Inputnew=xlsread(Filename,Sheetread1,Input2);
Targetnew=xlsread(Filename,Sheetread1,Target2 );
```

Example of ANN (1): (cont.)

```
xnew=Inputnew;

tnew=Targetnew;

x=Input';

t=Target';

xnew=Inputnew';

tnew=Targetnew';

trainFcn = 'trainlm'; hiddenLayerSize = 10;

net = fitnet(hiddenLayerSize,trainFcn);

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

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

RandStream.setGlobalStream (RandStream ('mrg32k3a'));% Just to get the same results; Set random number stream;

net.divideMode = 'sample';
```

Example of ANN (1): (cont.)

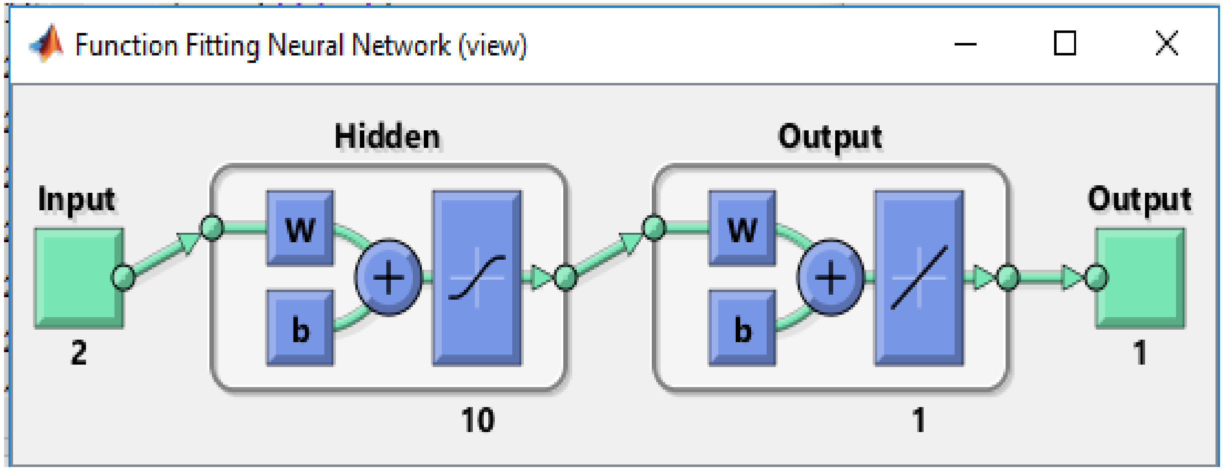
```
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 20/100;
net.divideParam.testRatio = 10/100;
net.performFcn = 'mse';  % Choose MSE for performance
[net,tr] = train(net,x,t);
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y);
figure;plotperform(tr)
figure;plottrainstate(tr)
figure, plotregression(t,y)
trainTargets = t .* tr.trainMask{1};
```

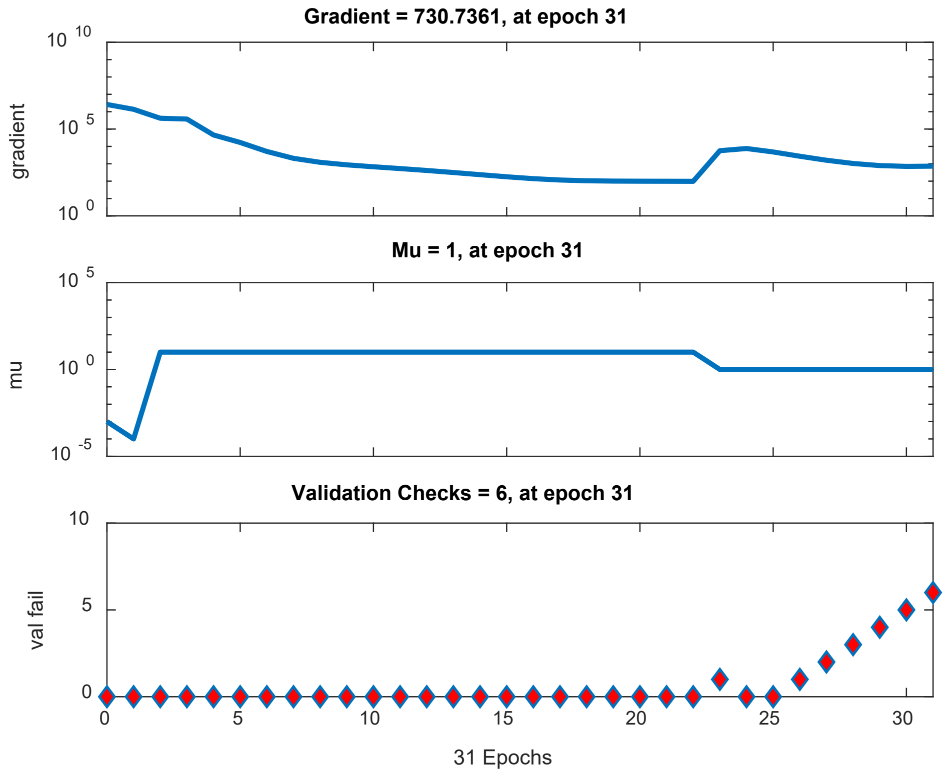
Example of ANN(1): (cont.)

```
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
Ynew=net(xnew); %Test the net with new data to calculate the performance and make predictions.
table( tnew( 1: 3)',Ynew( 1: 3)', 'VariableNames',...
{'Actual_Torque',' Predicted_Torque'})
```

Example of ANN(1): (cont.)

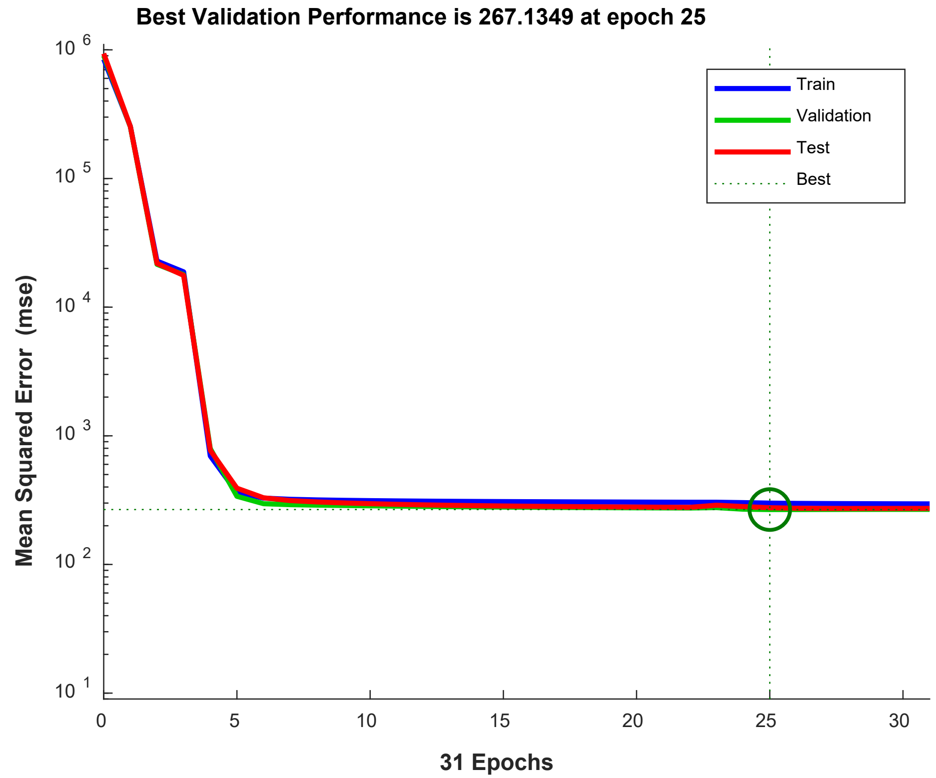
```
MSE_testing=sum((tnew-Ynew).^2)/numel(tnew); % Calculate MSE for new data
RMSE_testing=sqrt(sum((tnew-Ynew).^2)/numel(tnew));
% Calculate RMSE for new data
```

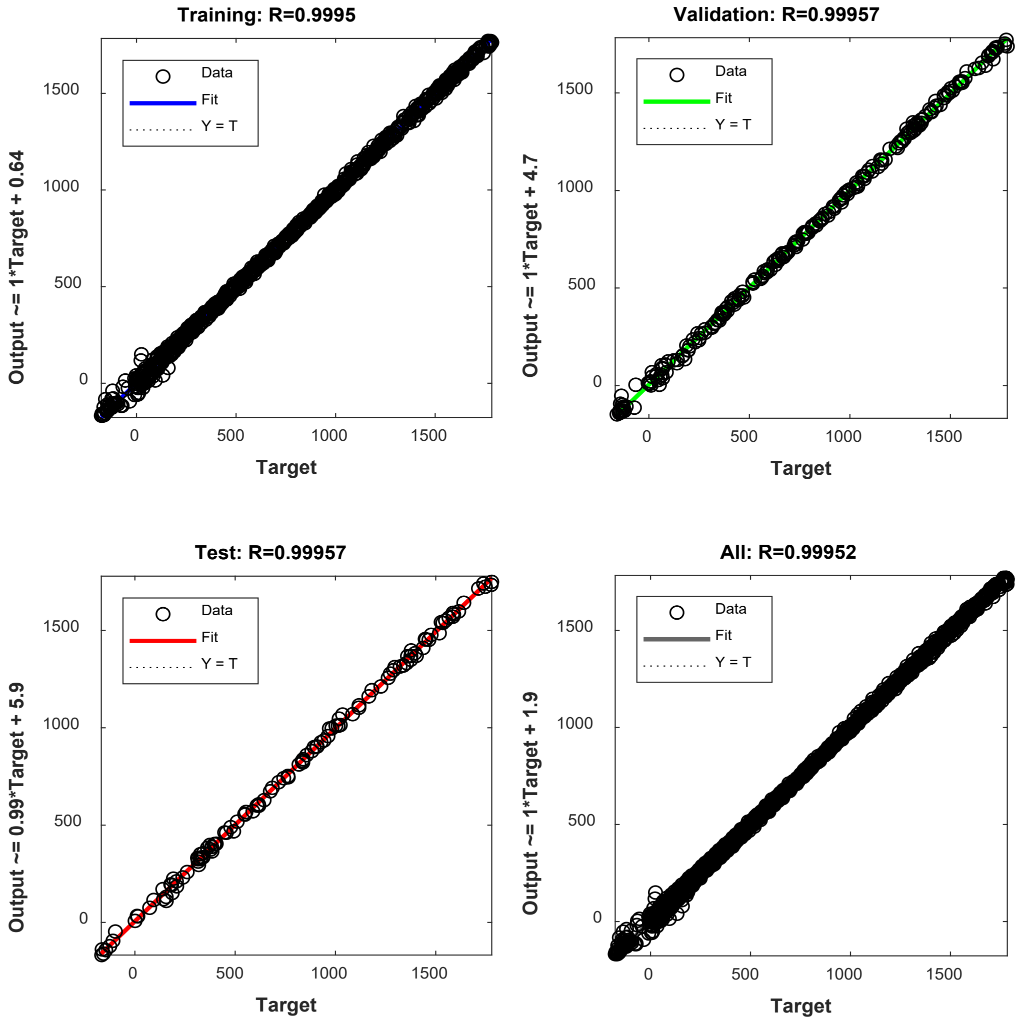




Plottrainstate

Plotperform





Plotregression

```
trainPerformance =

  299.7241


valPerformance =

  267.1349


testPerformance =

  275.9776
```

Actual_data_tnew	Predicted_data_Ynew
58	50.023
45.5	55.111
57.6	50.739

MSE_newdata=67.6933

RMSE_newdata=8.2278

Example of ANN, SVM, and NLR: (ANN)

```
clear;clc;
Filename='Datachemical.xlsx';
Sheetread='Training';
Input1='A1:C72';
output1='D1:D72';
Input=xlsread(Filename,Sheetread,Input1);
Target=xlsread(Filename,Sheetread,output1 );
x=Input;
t=Target;
Sheetread1='Newdata';
Input2='A1:C3';
Target2 ='D1:D3';
Inputnew=xlsread(Filename,Sheetread1,Input2);
Targetnew=xlsread(Filename,Sheetread1,Target2);
```

Example of ANN, SVM, and NLR :(ANN) (cont.)

```
xnew=Inputnew;

tnew=Targetnew;

x=Input';

t=Target';

xnew=Inputnew';

tnew=Targetnew';

trainFcn = 'trainlm'; hiddenLayerSize = 10;

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

net = fitnet(hiddenLayerSize,trainFcn);

net.input.processFcns = {'mapminmax'}; % To standardize
the input between -1 to 1.

net.output.processFcns = {'mapminmax'}; % To standardize
the output -1 to 1

RandStream.setGlobalStream (RandStream ('mrg32k3a'));%
Just to get the same results create random number streams;

net.divideMode = 'sample';
```

Example of ANN, SVM, and NLR: (ANN) (cont.)

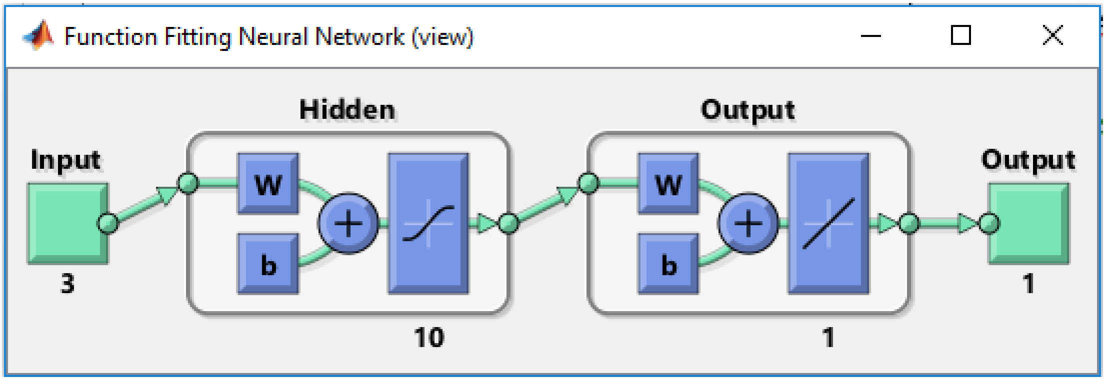
```
net.divideParam.trainRatio = 70/100;

net.divideParam.valRatio = 20/100;

net.divideParam.testRatio = 10/100;

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

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

y = net(x);

e = gsubtract(t,y); %Generalized subtraction

performance = perform(net,t,y);

figure;plotperform(tr)

figure;plottrainstate(tr) %Plot training state
valuesfigure, 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
```

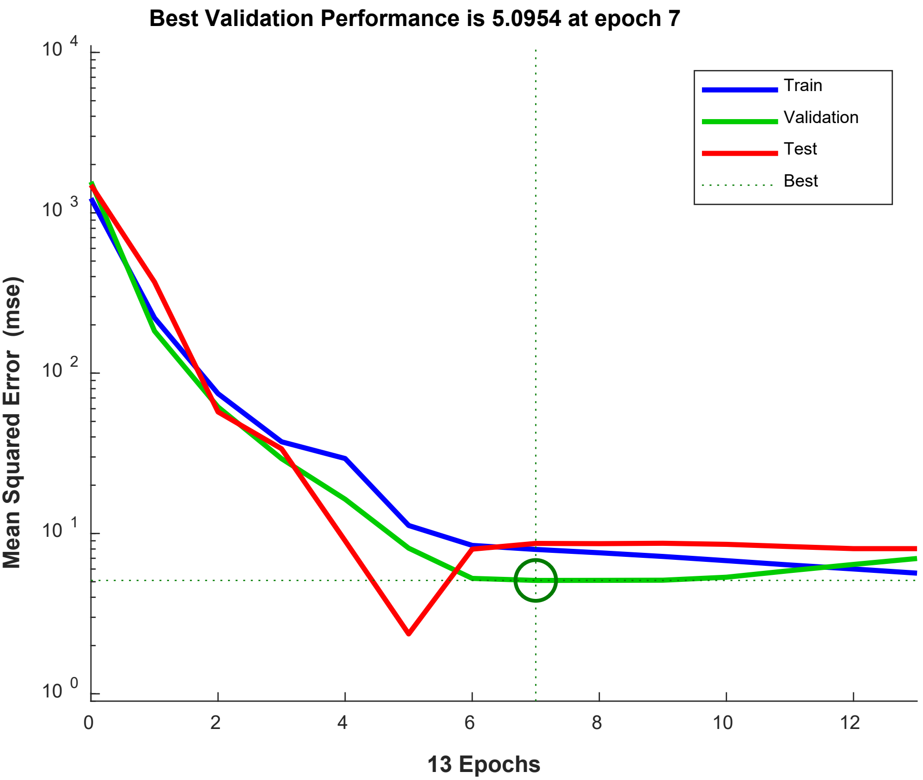
Example of ANN, SVM, and NLR: (ANN) (cont.)

```
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

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
```

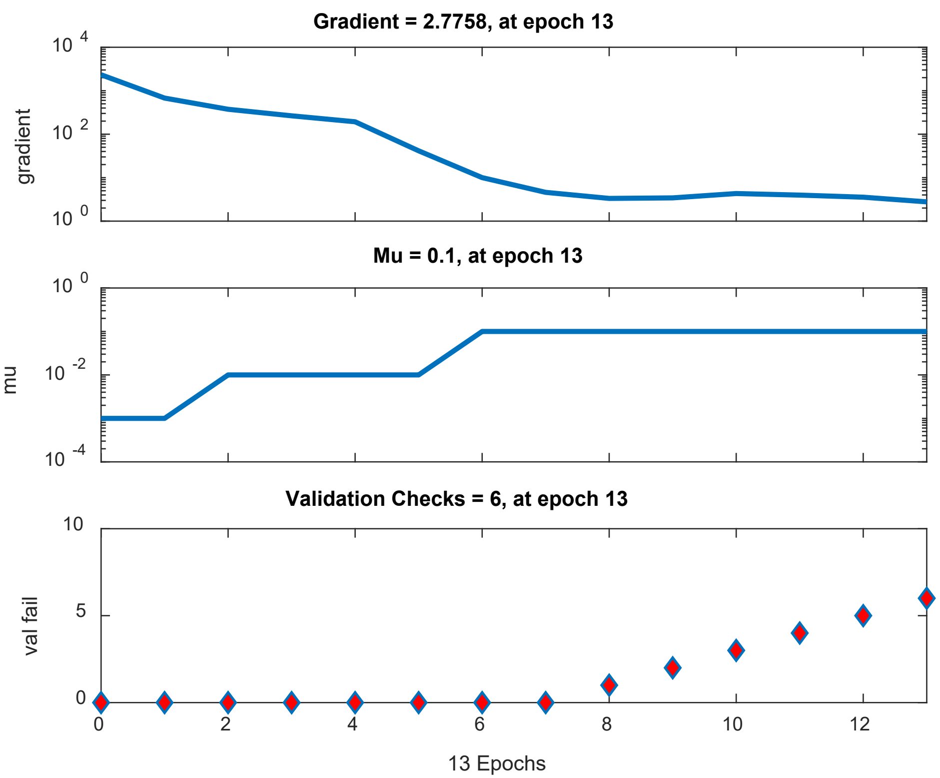
Example of ANN, SVM, and NLR: (ANN) (cont.)

```
MSE_testing=sum((tnew-Ynew).^2)/numel(tnew); % Calculate MSE for new data

RMSE_testing=sqrt(sum((tnew-Ynew).^2)/numel(tnew)); % Calculate RMSE for new data

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

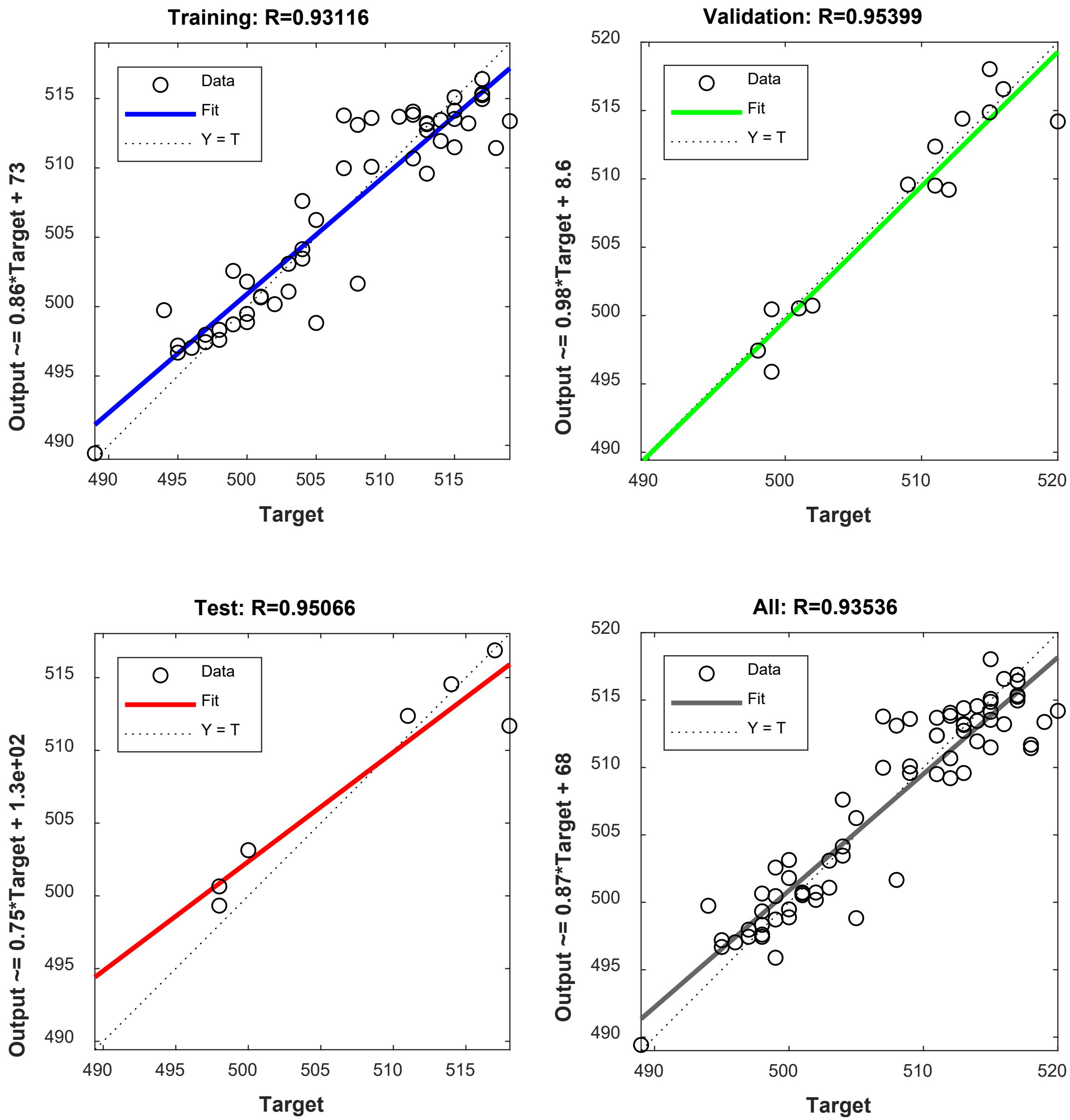




Plotperform



Plottrainstate



Plotregression

```
Command Window

trainPerformance =

    7.9503


valPerformance =

    5.0954


testPerformance =

    8.6680
```

3×2 table

Actual_data_tnew	Predicted_data_Ynew
514	515.79
516	512.83
512	514.58

MSE_newdata=6.6460

RMSE_newdata=2.5780

Errorpercentage

1x3 double

	1	2	3	4
1	0.3478	-0.6150	0.5045	
2				
3				
4				
5				

Example of ANN, SVM, and NLR: (SVM) (cont.)

```
clear;clc;
Filename='Datachemical.xlsx';
Sheetread='Training';
Input1='A1:C72';
output1='D1:D72';
Input=xlsread(Filename,Sheetread,Input1);
Target=xlsread(Filename,Sheetread,output1 );
x=Input;
t=Target;
Sheetread1='Newdata';
Input2='A1:C3';
Target2 ='D1:D3';
```

Example of ANN, SVM, and NLR: (SVM) (cont.)

```
Inputnew=xlsread(Filename,Sheetread1,Input2);

Targetnew=xlsread(Filename,Sheetread1,Target2 );

xnew=Inputnew;

tnew=Targetnew;

mdl = fitrsvm(x,t,'KernelFunction', 'gaussian', ...
     'Standardize', true); %standardize the data
%standardize the data and

conv = mdl.ConvergenceInfo.Converged; % Shows whether the
program reach an answer or not .

% smetimes it cannot find a solution and doesnot converge

iter = mdl.NumIterations; % number of iteration to reach
the answer
```

Example of ANN, SVM, and NLR: (SVM) (cont.)

```
yfit=predict(mdl,x); % prediction based on the developed SVR model and x as the input

table(t(1:10,:),yfit(1:10,:),'VariableNames',{'ObservedValue',' PredictedValue'}) % show 20th to 30th data in output and predicted output

MSE_training=sum((yfit-t).^2)/numel(t); % Calculate MSE for data

% MSE_training1=mse(yfit,t);

RMSE_training=sqrt(sum((yfit-t).^2)/numel(t)); % Calculate RMSE for data

ynew=predict(mdl,xnew);

table(tnew(:),ynew(:),'VariableNames',{'ObservedValue_Newdata',' PredictedValue_newdata'}) % show data in output and predicted output

% MSE_testing1=mse(tnew,ynew);
```

Example of ANN, SVM, and NLR: (SVM) (cont.)

```
MSE_testing=sum((tnew-ynew).^2)/numel(tnew); % Calculate MSE for new data

RMSE_testing=sqrt(sum((tnew-ynew).^2)/numel(tnew)); % Calculate RMSE for new data

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

ObservedValue	PredictedValue
511	511.9
504	505.17
512	514.62
505	506.05
507	510.81
501	502.04
500	500.05
505	498.23
502	500.96
508	502.33
502	500.95

Example of ANN, SVM, and NLR: (SVM) (cont.)

3×2 table

ObservedValue_Newdata	PredictedValue_newdata
514	514.23
516	513.71
512	513.61

Errorpercentage

3x1 double

	1	2	3	4
1	0.0446			
2	-0.4444			
3	0.3150			
4				

MSE_training =5.7400

MSE_testing =2.6371

RMSE_training =2.3958

RMSE_testing =1.6239

Example of ANN, SVM, and NLR: (NLR)

```
clear;clc;
Filename='Datachemical.xlsx';
Sheetread='Training';
Input1='A1:C72';
output1='D1:D72';
Input=xlsread(Filename,Sheetread,Input1);
Target=xlsread(Filename,Sheetread,output1 );
x=Input;
t=Target;
Sheetread1='Newdata';
Input2='A1:C3';
Target2 ='D1:D3';
Inputnew=xlsread(Filename,Sheetread1,Input2);
Targetnew=xlsread(Filename,Sheetread1,Target2 );
```

Example of ANN, SVM, and NLR: NLR (cont.)

```
xnew=Inputnew;

tnew=Targetnew;

[xn,sxn] = mapminmax(x');% Standardize x

[tn,stn]= mapminmax(t');% Standardize t

xnewn = mapminmax('apply',xnew',sxn); % The same Process
setting of

%standardization for x should  be applied for xnew as well
.%xnewn is the

%standardized xnew

xn=xn';% standardized x

tn=tn';% standardized t
```

Example of ANN, SVM, and NLR: NLR (cont.)

```
xnewn=xnewn';%standardized xnew

beta = [1 1 1 1 1 1 1 1]; % coefficient initiation

fun=@(b,xn)b(1)+b(2)*xn(:,1)+b(3)*xn(:,2)+b(4)*xn(:,3)+b(5
)*(xn(:,1).^2)+b(6)*((xn(:,1).*xn(:,2).*xn(:,3))+b(7).*exp
(b(8)*xn(:,3)));% nonlinear model with standardized x

mdl = fitnlm(xn,tn,fun,beta);% find coeffcients(beta) of
model(fun )using normalized x and t

yfitn = predict(mdl,xn);% make prediction based on
normalized x

yfit = mapminmax('reverse', yfitn,stn); % To reverse the
prediction to original state using the same process
setting of t

table( t( 40:50 ),yfit( 40:50 ), 'VariableNames',...

    {' TrueLabel',' PredictedLabel'}) %Show the results of
40th  to 50th
```

Example of ANN, SVM, and NLR: NLR (cont.)

```
MSE_training=sum((yfit-t).^2)/numel(t); % Calculate MSE
for data

RMSE_training=sqrt(sum((yfit-t).^2)/numel(t)); % Calculate
RMSE for data

ynewn=predict(mdl,xnewn);% make prediction based on
normalized new data

ynew = mapminmax('reverse', ynewn,stn); % To reverse the
normalized ynew and use the processing setting of t

table(tnew(:),ynew(:),'VariableNames',{'ObservedValue_Newd
ata',' PredictedValue_newdata'}) % show data in output and
predicted output

MSE_testing=sum((tnew-ynew).^2)/numel(tnew); % Calculate
MSE for new data

RMSE_testing=sqrt(sum((tnew-ynew).^2)/numel(tnew)); %
Calculate RMSE for new data

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

TrueLabel	PredictedLabel
511	512.76
504	509.69
512	513.41
505	505.57
507	511.84
501	504.31
500	499.39
505	496.6
502	497.6
508	499.4
502	498.5

MSE_training =10.1974

RMSE_training =3.1933

MSE_testing =5.6746

RMSE_testing =2.3821

3×2 table

ObservedValue_Newdata	PredictedValue_newdata
514	517.44
516	514.06
512	513.19

mdl.Coefficients

	1 Estimate
1 b1	-3.1592
2 b2	0.5237
3 b3	-0.0042
4 b4	-1.7069
5 b5	-0.0952
6 b6	0.0679
7 b7	44.2546
8 b8	0.4503

Errorpercentage

3x1 double

	1	2
1	0.6695	
2	-0.3755	
3	0.2334	
4		

ANN

MSE_training =7.9503

MSE_testing =8.6680

SVR

MSE_training =5.7400

MSE_testing =2.6371

NLR

MSE_training =10.1974

MSE_testing =5.6746