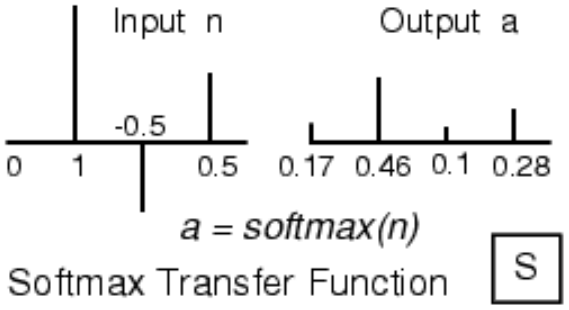
Activation Function-MATLAB Syntax

Question: Explain any four activation function with their MATLAB syntax, graph & example

1. Softmax

Graph and Symbol



Syntax

```
A = softmax(N)

info = softmax(code)
```

Examples

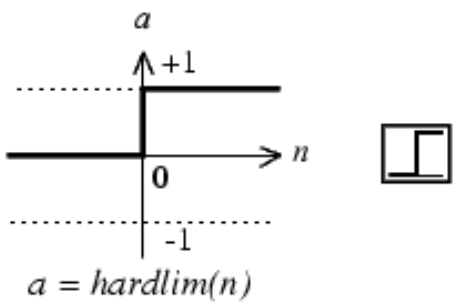
Here we define a net input vector N, calculate the output, and plot both with bar graphs.

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

Activation Function-MATLAB Syntax

2. Hard limit (Unit Step)

Graph and Symbol



Hard-Limit Transfer Function

Syntax

```
A = hardlim(N)

info = hardlim(code)
```

Description

The hard limit transfer function forces a neuron to output a 1 if its net input reaches a threshold, otherwise it outputs 0. This allows a neuron to make a decision or classification. It can say *yes* or *no*. This kind of neuron is often trained with the perceptron learning rule.

Activation Function-MATLAB Syntax

Examples

Here is the code to create a plot of the `hardlim` transfer function.

```
n = -5:0.1:5;
a = hardlim(n);
plot(n,a)
```

Algorithm

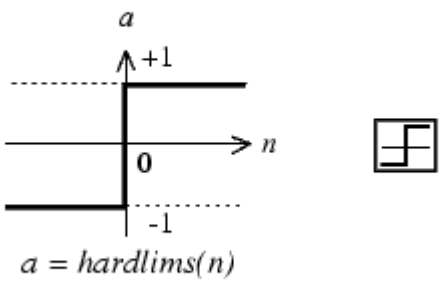
The transfer function output is one is n is less than or equal to 0 and zero if n is less than 0.

`hardlim(n)` = 1, if `n` >= 0; 0 otherwise.

Activation Function-MATLAB Syntax

3. Symmetric hard limit (Signum)

Graph and Symbol



Symmetric Hard-Limit Trans. Funct.

Syntax

```
A = hardlims(N)

info = hardlims(code)
```

Description

The symmetric hard limit transfer function forces a neuron to output a 1 if its net input reaches a threshold. Otherwise it outputs -1. Like the regular hard limit function, this allows a neuron to make a decision or classification. It can say yes or no.

Activation Function-MATLAB Syntax

Examples

Here is the code to create a plot of the `hardlims` transfer function.

```
n = -5:0.1:5;
a = hardlims(n);
plot(n,a)
```

Algorithm

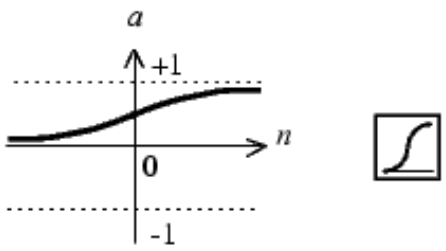
The transfer function output is one is n is greater than or equal to 0 and -1 otherwise.

`hardlim(n)` = 1, if `n` >= 0; -1 otherwise.

Activation Function-MATLAB Syntax

4. Log sigmoid

Graph and Symbol



a = logsig(n)

Log-Sigmoid Transfer Function

Syntax

```
A = logsig(N)

info = logsig(code)
```

Description

logsig is a transfer function. Transfer functions calculate a layer's output from its net input.

Activation Function-MATLAB Syntax

Examples

Here is the code to create a plot of the `logsig` transfer function.

```
n = -5:0.1:5;
a = logsig(n);
plot(n,a)
```

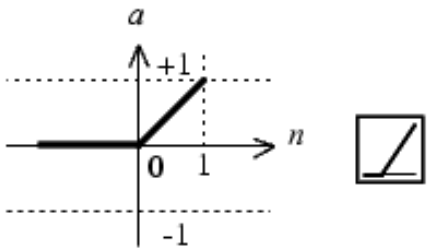
Algorithm

```
logsig(n) = 1 / (1 + exp(-n))
```

Activation Function-MATLAB Syntax

5. Poslin (RelU)

Graph and Symbol



$a = poslin(n)$

Positive Linear Transfer Funct.

Syntax

```
A = poslin(N)

info = poslin(code)
```

Description

poslin is a transfer function. Transfer functions calculate a layer's output from its net input.

Activation Function-MATLAB Syntax

Examples

Here is the code to create a plot of the `poslin` transfer function.

```
n = -5:0.1:5;
a = poslin(n);
plot(n,a)
```

Algorithm

The transfer function `poslin` returns the output `n` if `n` is greater than or equal to zero and 0 if `n` is less than or equal to zero.

`poslin(n) = n, if n >= 0; = 0, if n <= 0.`

Feed Forward Neural Network using MATLAB

Question: Write a program in MATLAB to train a feed-forward backpropagation network for $y = a + 2b + 3c$, where a,b,c are random numbers between 0 to 8, 2 layers containing 4 neurons in hidden layer and 1 neuron in output layer, use 'logsig' and 'purelin' activation function for hidden and output layer respectively and 1000 epochs.

Step 1: Let us produce 50 random numbers a, b, c between 0 & 5

```
a=randi([0,8],1,50)
b=randi([0,8],1,50)
c=randi([0,8],1,50)
```

Step 2: Consider input and output vector

```
x=[a; b; c]
y=a+2b+3c
```

Feed Forward Neural Network using MATLAB

Step 3: Use Matlab function newff

```
net=newff(minmax(x),[0,1],[4,1],{'logsig','purelin'});
net = init(net);
net.trainParam.epochs=1000
net.trainParam.goal=1e-25;
net.trainParam.lr=0.01;
net=train(net,x,y)
```

Prepare all the examples from MATLAB Assignment

Pimpri Chinchwad Education Trust's

Pimpri Chinchwad College of Engineering

Programming Assignment I

Course: Neural Network & Fuzzy Logic Control

(Open Elective)

Name of the student: Amay Chandravanshi

Roll Number: SXCOA32

Date of submission: 27 / 04 /2023

Max Marks: 60/60

Dinesh Kute

Course Teacher

Q.1 Write MATLAB code for Auto associative network for the input vector $x_1 = [1 \ -1 \ \ 1 \ -1]$ and $x_2 = [1 \ \ 1 \ \ -1 \ -1]$. Test the algorithm's performance for one missing entry, two missing entries, and one mistaken entry. [10 Marks]

Solution:

MATLAB Code:

```
clear all
x = [1,-1,1,-1 ; 1,+1,-1 -1]
t = x
w = x' * x
x2 = [0,-1, 1,-1]
x1 = [0,-1, 0,-1]
x3 = [1,1,1,-1]
z1 = x1 * w
z2 = x2 * w
z3 = x3 * w
activation_function(z1)
activation_function(z2)
activation_function(z3)
function activation_function(z1)
  for j=1:4
      if z1(1,j)>0
          y1(1,j) = 1;
      elseif z1(i,j)<0
          y1(i,j) = -1;
```

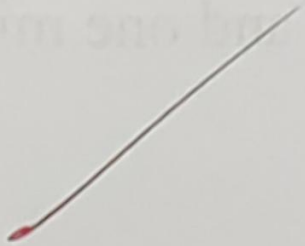
*Note: Attach MATLAB output

Marks

MATLAB CODE: ____/05

MATLAB OUTPUT: ____/05

```
    else
      H(i,j) = 0;
    end
  end
end
H
end;
```



```
%Name: Amay Chandravanshi
%Roll No: SYCOA32
clear all
%Input Vector
x=[1,-1,1,-1 ; 1,1,-1,-1]
```

```
x = 2×4
     1    -1     1    -1
     1     1    -1    -1
```

```
%Target Vector
t=x
```

```
t = 2×4
     1    -1     1    -1
     1     1    -1    -1
```

```
%Weight Matrix
W=x'*x
```

```
W = 4×4
     2     0     0    -2
     0     2    -2     0
     0    -2     2     0
    -2     0     0     2
```

```
%Testing vector: One missing entry
x2=[0,-1,1,-1]
```

```
x2 = 1×4
     0    -1     1    -1
```

```
%Testing vector: Two missing entries
x1=[0,-1,0,-1]
```

```
x1 = 1×4
     0    -1     0    -1
```

```
%Testing vector: One mistaken entry
x3=[1,1,1,-1]
```

```
x3 = 1×4
     1     1     1    -1
```

```
%Net Input for x1
z1=x1*W
```

```
z1 = 1×4
     2    -2     2    -2
```

```
%Net Input for x2
```

```
z2=x2*W

z2 = 1×4
     2    -4     4    -2


%Net Input for x3
z3=x3*W

z3 = 1×4
     4     0     0    -4


activation_function(z2) %One missing entry

y1 = 1×4
     1    -1     1    -1


activation_function(z1) %Two missing entries

y1 = 1×4
     1    -1     1    -1


activation_function(z3) %Two mistaken entries

y1 = 1×4
     1     0     0    -1


function activation_function(z1)
for j=1:4
    if z1(1,j)>0
        y1(1,j)=1;
    elseif z1(1,j)<0
            y1(1,j)=-1;
    else
        y1(1,j)=0;
    end
end
y1
end
```

Q.2 Write MATLAB code for Hetero associative network for the input vector $X = [x_1 \ x_2 \ x_3 \ x_4]$ and output vector $Y = [y_1 \ y_2]$.Test the algorithm's performance for one missing entry, two missing entries, and one mistaken entry. [10 Marks]

Input Vector				Target Output	
x_1	x_2	x_3	x_4	y_1	y_2
1	1	1	-1	1	-1
1	-1	1	-1	1	-1
-1	-1	1	1	-1	1
-1	-1	-1	1	-1	1

Solution:

MATLAB Algorithm:

```
clear all
x= [1,1,1,-1; 1 -1 -1 1; -1 -1 -1 1 ; -1 -1 -1 1];
t= [1,-1 ; 1,-1; -1, 1; -1, 1]
w =  x' * t
x1 = [0,1,0,-1]
x2 = [0,1,1,-1]
x3 = [-1,1,-1,-1]
z1 = x1 * w
```

*Note: Attach MATLAB output

Marks

MATLAB CODE: ____/05

MATLAB OUTPUT: ____/05

```
z2 = x2 * w

z3 = x3 * w

activation_function(z1)

activation_function(z2)

activation_function(z3)

function activation_function(z1)
 for j = 1:2
     if z1(1,j) > 0
          y1(1,j) = 1;
     elseif z1(i,j) < 0
          y1(1,j) = -1;
     else
          y1(1,j) = 0;
     end
 end
 y1
 end
```

```
%Name: Amay Chandravanshi
%Roll No: SYCOA32
clear all
%Input Vector
x=[1,1,1,-1 ; 1,-1,1,-1 ; -1,-1,-1,1 ; -1,-1,-1,1]
```

```
x = 4×4
     1     1     1    -1
     1    -1     1    -1
    -1    -1    -1     1
    -1    -1    -1     1
```

```
%Target Output Vector
t=[1,-1 ; 1,-1 ; -1,1 ; -1,1]
```

```
t = 4×2
     1    -1
     1    -1
    -1     1
    -1     1
```

```
%Weight Matrix
W=x'*t
```

```
W = 4×2
     4    -4
     2    -2
     4    -4
    -4     4
```

```
%Testing vector: Two missing entries
x1=[0,1,0,-1]
```

```
x1 = 1×4
     0     1     0    -1
```

```
%Testing vector: One missing entries
x2=[0,1,1,-1]
```

```
x2 = 1×4
     0     1     1    -1
```

```
%Testing vector: Two mistaken entries
x3=[-1,1,-1,-1]
```

```
x3 = 1×4
    -1     1    -1    -1
```

```
%Net Input for x1
z1=x1*W
```

```
z1 = 1×2
```

```
    6    -6
```

```
%Net Input for x2
z2=x2*W
```

```
z2 = 1×2
    10   -10
```

```
%Net Input for x3
z3=x3*W
```

```
z3 = 1×2
    -2     2
```

```
activation_function(z2) %One missing entry
```

```
y1 = 1×2
     1    -1
```

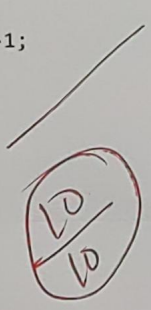
```
activation_function(z1) %Two missing entries
```

```
y1 = 1×2
     1    -1
```

```
activation_function(z3) %Two mistaken entries
```

```
y1 = 1×2
    -1     1
```

```
function activation_function(z1)
for j=1:2
    if z1(1,j)>0
        y1(1,j)=1;
    elseif z1(1,j)<0
            y1(1,j)=-1;
    else
        y1(1,j)=0;
    end
end
y1
end
```



Q.3 Write MATLAB code to update weight and bias using Hebb Learning Rule with implementation of bipolar AND function and attach the MATLAB output.

[Initialize the weights & bias with small random values $w = rand(1,3),\ b = rand()$*]*

Solution:

MATLAB Code:

```
clear all
x=[1 1 ; 1 -1; -1 1; -1 -1]
t=[1 -1 -1 -1]
w= [rand (1,3) rand (1,3)]
b=rand()
for i=1:4
    for j=1:2
        w(j)=w(j)+t(1,i)*x(i,j);
    end
    b=b+t(1,j);
end
w
b
```

*Note: Attach MATLAB output

Marks

MATLAB CODE: ____/05 MATLAB OUTPUT: ____/05

```
%Name: Amay Chandravanshi
%Roll No: SYCOA32
clear all
%Bipoloar AND Function with bias
x=[1 1;1 -1; -1 1; -1 -1]
```

```
x = 4×2
     1     1
     1    -1
    -1     1
    -1    -1
```

```
%Target Vector
t=[1 -1 -1 -1]
```

```
t = 1×4
     1    -1    -1    -1
```

```
w=[rand(1,3) rand(1,3)]
```

```
w = 1×6
    0.8147    0.9058    0.1270    0.9134    0.6324    0.0975
```

```
b=rand()
```

```
b = 0.2785
```

```
for i=1:4
    for j=1:2
        w(j)=w(j)+t(1,i)*x(i,j);
    end
    b=b+t(1,i);
end
w
```

```
w = 1×6
    2.8147    2.9058    0.1270    0.9134    0.6324    0.0975
```

```
b
```

```
b = -1.7215
```

10/10

Q.4 Write MATLAB code to update weight of the following ANN model using Delta Learning Rule and sigmoidal function $\left[\phi(z) = \frac{1}{1+e^{-z}}\right]$ and attach the MATLAB output.

[Initialize the weights with small random values $w = rand(1,3)$*]*

Input			Target Output
x_1	x_2	x_3	y
0	0	1	0
0	1	1	0
1	0	1	1
1	1	1	1

Solution:

MATLAB Code:

```
clear all;
x = [0 0 1; 0 1 1; 1 0 1; 1 1 1]
yt = [0; 0; 1; 1]
w = [rand(1,3) rand(1,3) rand(1,3)]
for i = 1:4
    z = x(i,1) * w(1) + x(i,2) * w(2) + x(i,3) * w(3);
    y = Sigmoid(z);
    for j = 1:3
    w(j) = w(j) + (yt(i) - y) * derivative_sigmoid
    (z) * x(i,j);
```

*Note: Attach MATLAB output

Marks

MATLAB CODE: ____/05 MATLAB OUTPUT: ____/05

```
    end
end
w
function [Ans] = sigmoid(z)
Ans = 1/(1+exp(-z));
end

function [a] = derivative_sigmoid(z)
a = Sigmoid(z) * (1 - sigmoid(z));
end
```

```
%Name: Amay Chandravanshi
%Roll No: SYCOA32
clear all;
```

```
%Input Vector
x=[0 0 1 ; 0 1 1 ; 1 0 1 ; 1 1 1]
```

```
x = 4×3
     0     0     1
     0     1     1
     1     0     1
     1     1     1
```

```
%Output Vector
yt=[0 ; 0 ; 1 ; 1]
```

```
yt = 4×1
     0
     0
     1
     1
```

```
%Weights Initialization
w=[rand(1,3) rand(1,3) rand(1,3)]
```

```
w = 1×9
    0.4733    0.3517    0.8308    0.5853    0.5497    0.9172    0.2858    0.7572 ···
```

```
%Weights Updation
for i=1:4
    z=x(i,1)*w(1)+x(i,2)*w(2)+x(i,3)*w(3);
    y=Sigmoid(z);
    for j=1:3
        w(j)=w(j)+(yt(i)-y)*derivative_sigmoid(z)*x(i,j);
    end
end
w
```

```
w = 1×9
    0.5600    0.2437    0.6276    0.5853    0.5497    0.9172    0.2858    0.7572 ···
```

```
%Utility Functions


function [Ans] = Sigmoid(z)
Ans=1/(1+exp(-z));
end

function [a] = derivative_sigmoid(z)
a=Sigmoid(z)*(1-Sigmoid(z));
end
```

10/10

Q.5 Describe the MATLAB syntax for any four activation (or transfer) functions, and explain each with one example (MATLAB output).

Solution:

1. Softmax:

```
A = softmax(N)
info = softmax('code')
```

2. Hard lim:

```
A = hardlim(N)
info = hardlim('code')
```

3. Purelin:

```
A = purelin(N)
info = purelin('code')
```

4. Poslin:

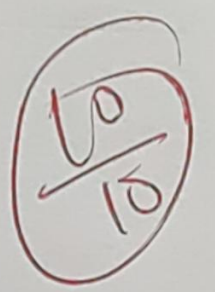
```
A = poslin(N)
info = poslin('code')
```

*Note: Attach MATLAB output

Marks

MATLAB CODE: ____/05 MATLAB OUTPUT: ____/05

```
%Name: Amay Chandravanshi
%Roll No: SYCOA32
clear all;

softmax(-1)

ans = 1

hardlim(-1)

ans = 0

purelin(100)

ans = 100

poslin(-1)

ans = 0
```



Q.6

Follow the following steps and complete feedforward backpropagation network.

Step 1: Generate random data for input x (min 100 observations)

Example: $x = a + (b - 1) * rand(m, n)$ generated random $m \times n$ values from the interval (a, b).

Step 2: Use any mathematical formula to generate output y

Example: $y = a.\hat{}2 + b.\hat{}2 + c.\hat{}2$ or $y = x.\hat{}2$

Step 3: Use syntax of ***newff*** and create feedforward backpropagation network.

Example: net=newff(minmax(x),[5,1],{'logsig','purelin'});

```
net = init(net);
net.trainParam.epochs=1000
net.trainParam.goal=1e-25;
net.trainParam.lr=0.01;
net=train(net,x,y)
```

Every student has to use their unique dataset, the number of hidden layers containing neurons, and the activation (transfer) function.

Attach the following MATLAB screenshot of the algorithm

Marks

MATLAB CODE: ____/05

MATLAB OUTPUT: ____/05

```
%Name: Amay Chandravanshi
%Roll No: SYCOA32
clear all
a=rand(1,1000)
```

```
a = 1×1000
    0.0886    0.2624    0.1515    0.3641    0.6810    0.1021    0.8871    0.4672 ···
```

```
%Input Matrix
x=(a.^2-5*a+6)
```

```
x = 1×1000
    5.5649    4.7567    5.2657    4.3119    3.0586    5.4998    2.3515    3.8821 ···
```

```
%Output Matrix
y=x.^2-5*x+6
```

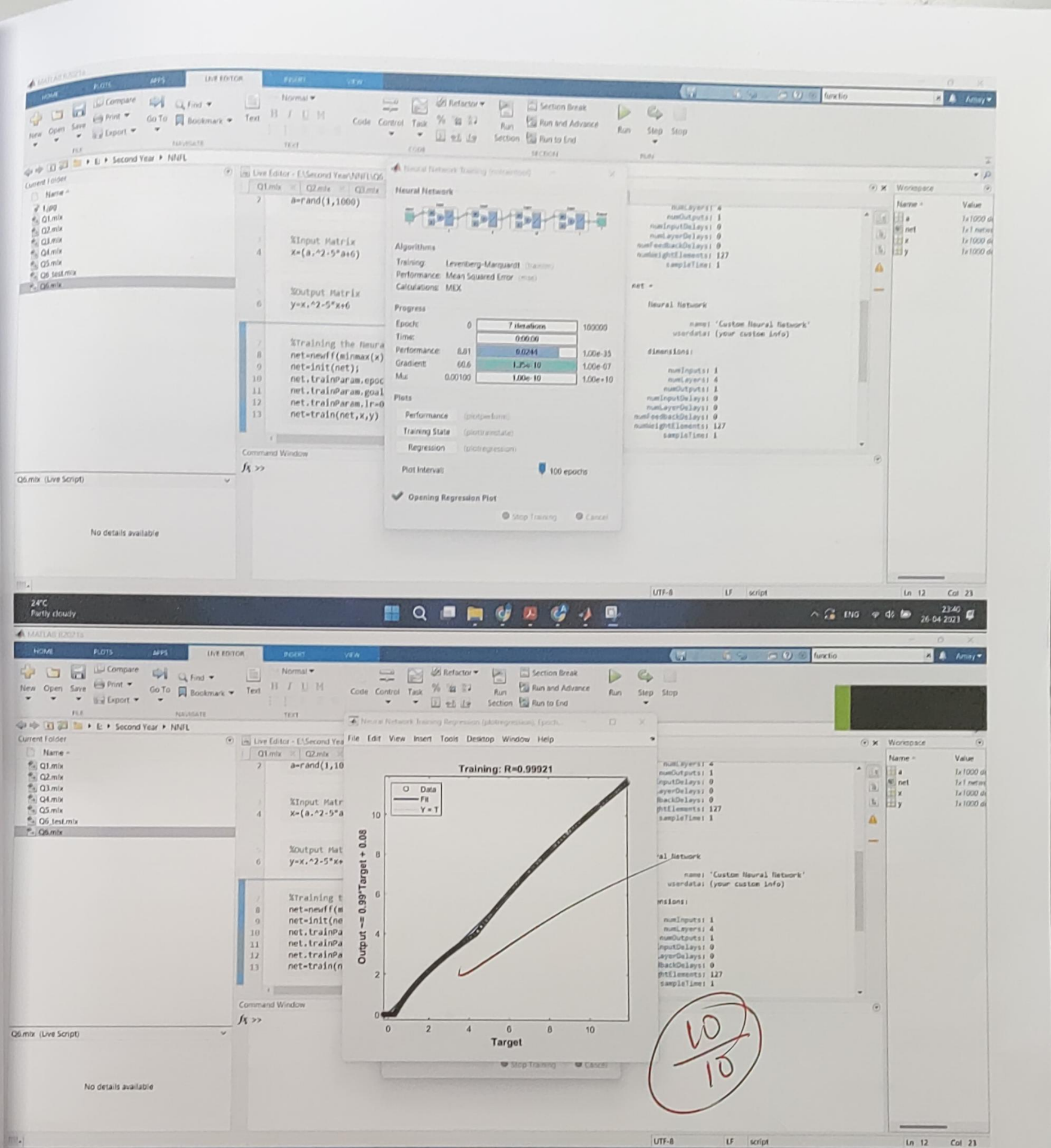
```
y = 1×1000
    9.1436    4.8428    7.3988    3.0330    0.0620    8.7487   -0.2280    1.6602 ···
```

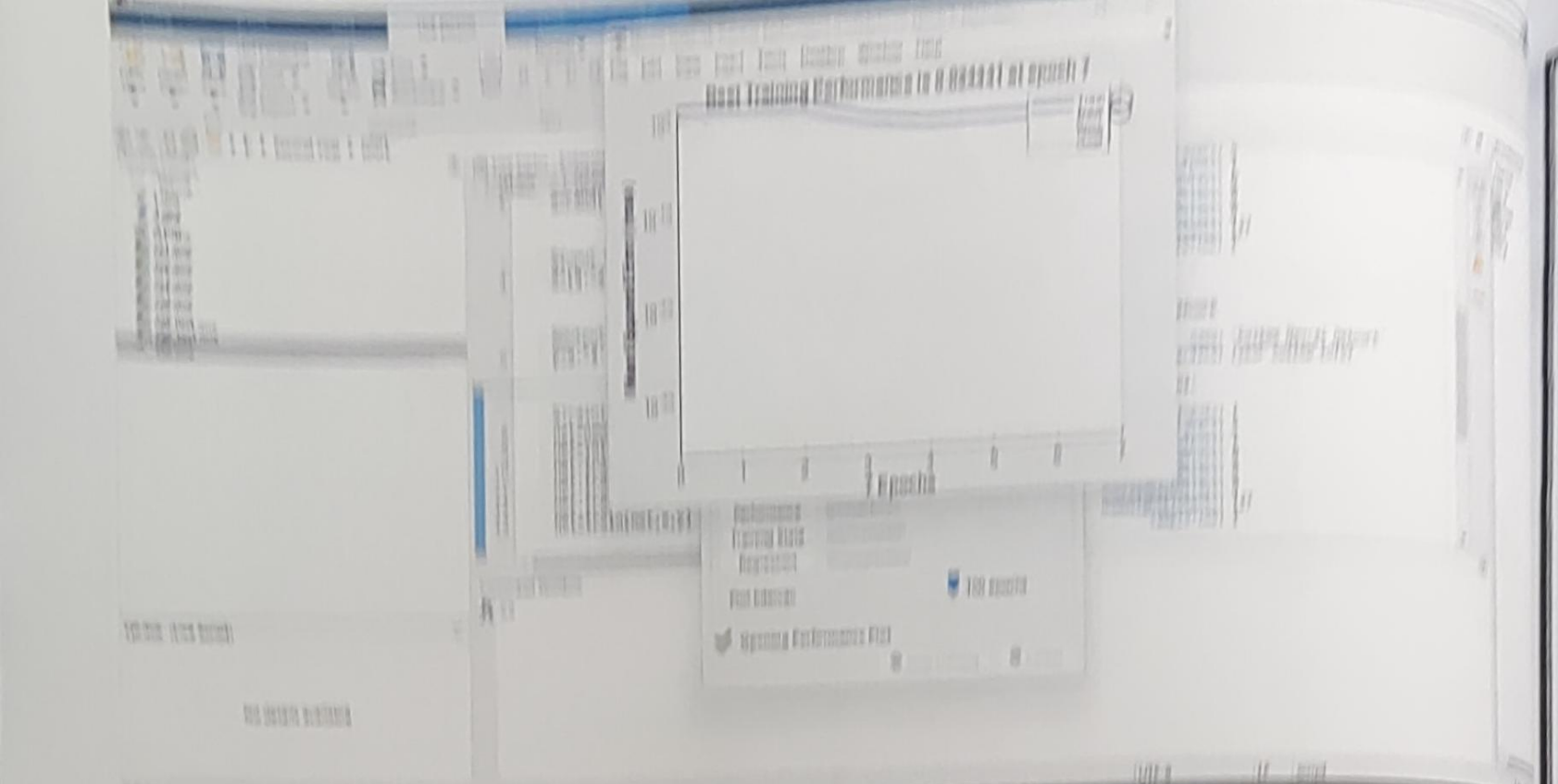
```
%Training the Neural Network
net=newff(minmax(x),[5,6,10,1],{'poslin','purelin','purelin','poslin'});
```

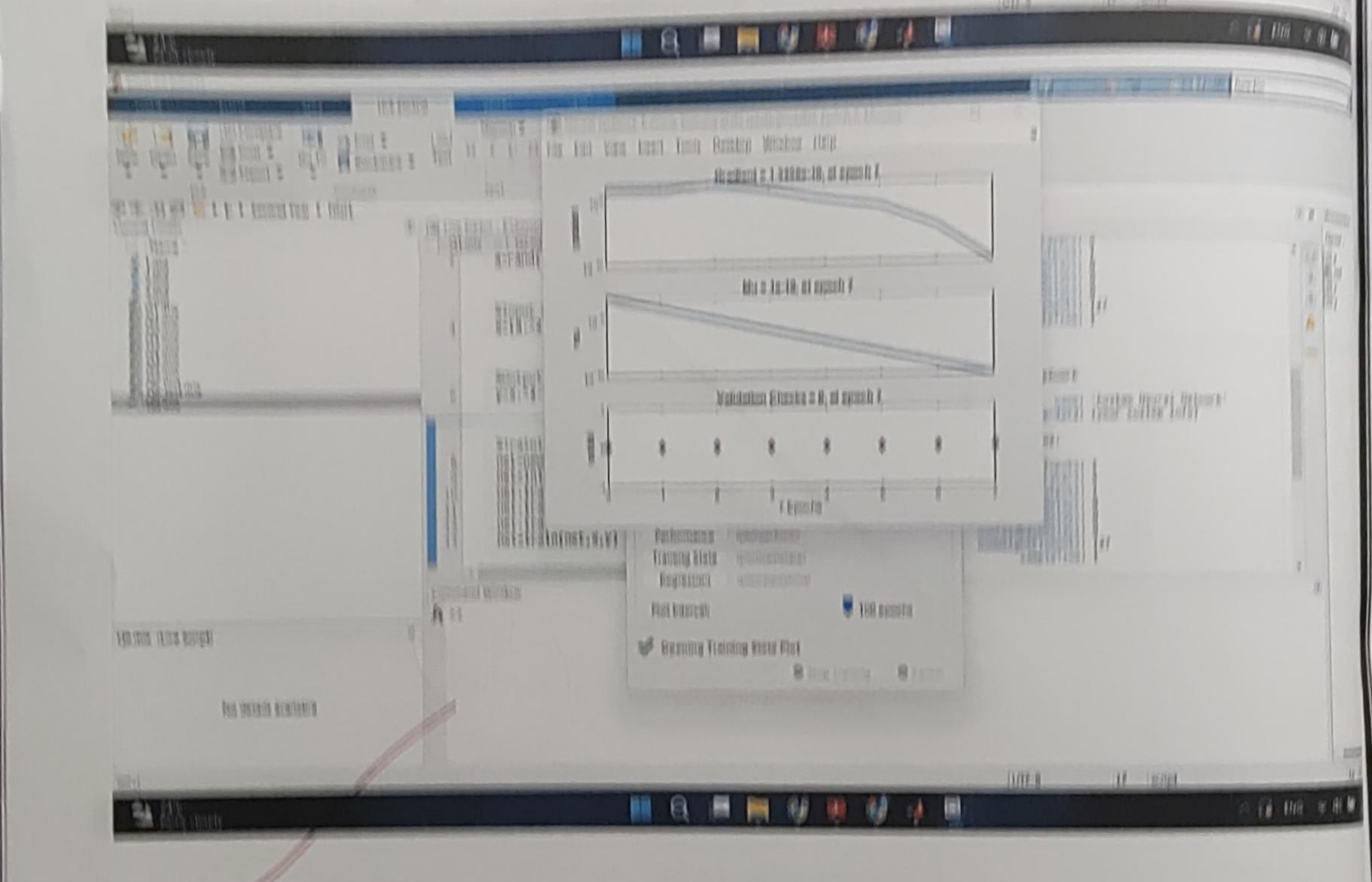
```
Warning: NEWFF used in an obsolete way.
         See help for NEWFF to update calls to the new argument list.
```

```
net=init(net);
net.trainParam.epochs=100000
```

```
net =

    Neural Network

              name: 'Custom Neural Network'
          userdata: (your custom info)

    dimensions:

         numInputs: 1
         numLayers: 4
        numOutputs: 1
    numInputDelays: 0
    numLayerDelays: 0
numFeedbackDelays: 0
numWeightElements: 127
       sampleTime: 1

    connections:

      biasConnect: [1; 1; 1; 1]
     inputConnect: [1; 0; 0; 0]
     layerConnect: [4x4 boolean]
    outputConnect: [0 0 0 1]

    subobjects:
```


Neural Network
Algorithms
Training: Levenberg-Marquardt
Performance: Mean Squared Error
Calculations: MEX
Progress
Epoch: 0 7 iterations 100000
Time: 0:00:00
Performance: 8.81 0.0244 1.00e-35
Gradient: 60.6 1.25e-10 1.00e-07
Mu: 0.00100 1.00e-10 1.00e+10
Plots
Performance
Training State
Regression
Plot Interval: 100 epochs
Opening Regression Plot
Stop Training
Cancel
Training: R=0.99921
Data
Fit
Y = T
Output ~= 0.99*Target + 0.08
Target
10/10





1) Upload screenshot of Neural Network Training(nntraintool)
2) Upload screenshot of Regression plot
3) Upload screenshot of Performance plot
4) Upload screenshot of Training State plot

MATLAB standard in build functions (refer this for activation function)

Item	Link
List of links	http://matlab.izmiran.ru/help/toolbox/nnet/nnetsloc.html
Transfer Function	http://matlab.izmiran.ru/help/toolbox/nnet/tabls113.html http://matlab.izmiran.ru/help/toolbox/nnet/tabls149.html
Soft max	http://matlab.izmiran.ru/help/toolbox/nnet/softmax.html#669443
Hard limit (Unit Step)	http://matlab.izmiran.ru/help/toolbox/nnet/hardlim.html#499618
Symmetric hard limit (Signum)	http://matlab.izmiran.ru/help/toolbox/nnet/hardlims.html#499657
Log sigmoid	http://matlab.izmiran.ru/help/toolbox/nnet/logsig.html#500748
Hyperbolic tangent sigmoid	http://matlab.izmiran.ru/help/toolbox/nnet/tansig.html#670934
Poslin (RelU)	http://matlab.izmiran.ru/help/toolbox/nnet/poslin.html#668876
Linear	http://matlab.izmiran.ru/help/toolbox/nnet/purelin.html#669092
Weight and Bias Initialization Functions	http://matlab.izmiran.ru/help/toolbox/nnet/tabls111.html
Weight & bias initialization function.	init
Conscience bias initialization function.	initcon
Zero weight and bias initialization function.	initzero
Midpoint weight initialization function.	midpoint
Normalized column weight initialization function.	randnc
Normalized row weight initialization function.	randnr

Marks

MATLAB CODE: ____/05

MATLAB OUTPUT: ____/05

Symmetric random weight/bias initialization function.	rands
Change ntwk wts. and biases to prev. initialization values.	revert
Layer-by-layer network initialization function	initlay
Nguyen-Widrow layer initialization function	initnw
Training Functions	http://matlab.izmiran.ru/help/toolbox/nnet/tabls147.html
Batch training with weight and bias learning rules.	trainb
BFGS quasi-Newton backpropagation.	trainbfg
Bayesian regularization.	trainbr
Cyclical order incremental update.	trainc
Powell-Beale conjugate gradient backpropagation.	traincgb
Fletcher-Powell conjugate gradient backpropagation.	traincgf
Polak-Ribiere conjugate gradient backpropagation.	traincgp
Gradient descent backpropagation.	traingd
Gradient descent with adaptive lr backpropagation.	traingda
Gradient descent with momentum backpropagation.	traingdm
Gradient descent with momentum & adaptive lr backprop.	traingdx
Levenberg-Marquardt backpropagation.	trainlm
One step secant backpropagation.	trainoss
Random order incremental update.	trainr
Resilient backpropagation (Rprop).	trainrp

Marks

MATLAB CODE: ____/05

MATLAB OUTPUT: ____/05

Sequential order incremental update.	trains
Scaled conjugate gradient backpropagation.	trainscg
Performance Functions	http://matlab.izmiran.ru/help/toolbox/nnet/tabls145.html
Mean absolute error performance function.	mae
Mean squared error performance function.	mse
Mean squared error w/reg performance function.	msereg
Sum squared error performance function.	sse
Network Use Functions	http://matlab.izmiran.ru/help/toolbox/nnet/tabls144.html
Allow a neural network to adapt.	adapt
Display a neural network's properties.	disp
Display a neural network variable's name and properties.	display
Initialize a neural network.	init
Simulate a neural network.	sim
Train a neural network.	train
New Networks Functions	http://matlab.izmiran.ru/help/toolbox/nnet/tabls144.html
Create a custom neural network.	network
Create a competitive layer.	newc
Create a cascade-forward backpropagation network.	newcf
Create an Elman backpropagation network.	newelm
Create a feed-forward backpropagation network.	newff

Marks

MATLAB CODE: ____/05

MATLAB OUTPUT: ____/05

Create a feed-forward input-delay backprop network.	newfftd
Design a generalized regression neural network.	newgrnn

Solution:

*Note: Attach MATLAB output only

Marks

MATLAB CODE: ____/05

MATLAB OUTPUT: ____/05