1. **Feedforward Neural Network**

```
clc
clear all
close all
% Feedforward Neural Network
% input
x=[-1 -1 2 2; 0 5 0 5]
% Unknown input
x2=[1 1 2 2; 1 1 0 5]
% target
t=[-1 -1 1 1]
net=feedforwardnet(10);
net=train(net,x,t);
view(net)
y= net(x);
perf=perform(net,y,t)
y1=sim(net,x2)
```

2. Batch gradient Descent

```
clc
clear all
close all
% Batch gradient Descent
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'traingd');
net.trainParam.show = 50;
net.trainParam.lr = 0.05;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
y=sim(net,p)
% error
e=sum(y-t)
```

3. Backpropagation training with an adaptive learning rate

```
clc
clear all
close all
% Backpropagation training with an adaptive learning rate
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'traingda');
net.trainParam.show = 50;
net.trainParam.lr = 0.05;
net.trainParam.lr_inc = 1.05;
```

```
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
% TRAINGDA, Epoch 0/300, MSE 1.71149/1e-05, Gradient 2.6397/1e-06
% TRAINGDA, Epoch 44/300, MSE 7.47952e-06/1e-05, Gradient 0.00251265/1e-06
% TRAINGDA, Performance goal met
a=sim(net,p)
```

4. Resilient Backpropagation

```
clc
clear all
close all
% Resilient Backpropagation
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'trainrp');
net.trainParam.show = 10;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
a=sim(net,p)
```

5. conjugate gradient backpropagation training algorithms

```
clc
clear all
close all
% conjugate gradient backpropagation training algorithms
% Input
p=[-1 -1 2 2;0 5 0 5];
% Target
t=[-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1;0 4 0 4];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'traincgf');
net.trainParam.show = 5;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
a=sim(net,p)
a1=sim(net,ip)
```

6. conjugate gradient backpropagation training algorithms with Polak-Ribiére

```
clc
clear all
close all
% conjugate gradient backpropagation training algorithms with Polak-Ribiére
% Input
p=[-1 -1 2 2;0 5 0 5];
```

```
% Target
t=[-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1;0 4 0 4];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'traincgp');
net.trainParam.show = 5;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
a=sim(net,p)
a1=sim(net,ip)
```

7. conjugate gradient backpropagation training algorithms with Scaled Conjugate Gradient

```
clc
clear all
close all
% conjugate gradient backpropagation training algorithms with Scaled Conjugate Gradient
% Input
p=[-1 -1 2 2;0 5 0 5];
% Target
t=[-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1;0 4 0 4];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'trainscg');
net.trainParam.show = 10;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
a=sim(net,p)
a1=sim(net,ip)
```

8. Fletcher, Goldfarb, and Shanno algorithm (BFG)Quasi Newton

```
clc
clear all
close all
% conjugate gradient backpropagation training algorithms with Broyden,
% Fletcher, Goldfarb, and Shanno algorithm (BFG)Quasi Newton
% Input
p=[-1 -1 2 2;0 5 0 5];
% Target
t=[-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1;0 4 0 4];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'trainbfg');
net.trainParam.show = 5;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
```

```
a=sim(net,p)
a1=sim(net,ip)
```

9. **conjugate gradient backpropagation training algorithms with Levenberg-Marquardt algorithm Algorithm**

```
clc
clear all
close all
% conjugate gradient backpropagation training algorithms with Levenberg-Marquardt algorithm
Algorithm
% Input
p=[-1 -1 2 2;0 5 0 5];
% Target
t=[-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1;0 4 0 4];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'trainlm');
net.trainParam.show = 5;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t);
a=sim(net,p)
a1=sim(net,ip)
```

10. BFGS algorithm with the regularized performance function

```
clc
clear all
close all
% BFGS algorithm with the regularized performance function
% Input
p = [-1 -1 2 2;0 5 0 5];
% Target
t = [-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1;0 4 0 4];
net=newff(minmax(p),[3,1],{'tansig','purelin'},'trainbfg');
net.performFcn = 'msereg';
net.performParam.ratio = 0.5;
net.trainParam.show = 5;
net.trainParam.epochs = 300;
net.trainParam.goal = 1e-5;
[net,tr]=train(net,p,t)
a=sim(net,p)
e=t-a
ee=sum(e.^2)
a1=sim(net,ip)
```

11 Radial Basis Function Neural Network

```
clc
clear all
close all
% Radial Basis Function Neural Network
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1;0 4 0 4];
net=newrbe(p,t)
view(net)
a=sim(net,p)
%error
e=(a-t)
% absolute error
ee=sum(e)
a1=sim(net,ip)
```

12. More Efficient Design Radial Basis Function Neural Network (newrb)

```
clc
clear all
close all
% More Efficient Design Radial Basis Function Neural Network (newrb)
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1;0 4 0 4];
net=newrbe(p,t)
view(net)
a=sim(net,p)
%error
e=(a-t)
% absolute error
ee=sum(e)
a1=sim(net,ip)
```

13. Generalized Regression Neural Network

```
clc
clear all
close all
% Generalized Regression Neural Network
p = [-1 -1 2 2;0 5 0 5];
t = [-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1;0 4 0 4];
net=newgrnn(p,t)
view(net)
a=sim(net,p)
```

```
% Root Mean Square Error
rmse=((sum((a-t).^2))/4)^0.5
% Mean Bias Error
mbe=(sum(a-t))/4
% Mean Absolute Percentage Error
mape=(sum(abs((a-t))./t))*100/4
a1=sim(net,ip)
```

14. Elman Neural Network

```
clc
clear all
close all
% Elman Neural Network
% Input
p = [-1 -1 2 2];
% Target
t = [-1 -1 1 1];
% Unknown input
ip=[-1 -1 1 1];
% Syntax
net=newelm([0 1],[5 1],{'tansig','purelin'});
view(net)
a=sim(net,p)
% Root Mean Square Error
rmse=((sum((a-t).^2))/4)^0.5
% Mean Bias Error
mbe=(sum(a-t))/4
% Mean Absolute Percentage Error
mape=(sum(abs((a-t))./t))*100/4
a1=sim(net,ip)
```

15. **Elman Neural Network**

```
clc
clear all
close all
% Elman Neural Network for wind speed prediction
% Inputs
% Solar Radiation
pp=xlsread('WS.xls',6);
p=pp(1:800,:)
% Target Wind Speed
tt=xlsread('WS.xls',1);
t=tt(1:800,:)
t1=tt(801:852,:)
% Unknown input
ip=pp(801:852,:)
net=newelm([0 1],[5 1],{'tansig','logsig'});
view(net)
```

```
a=sim(net,p')
% Root Mean Square Error
rmse=((sum((a-t').^2))/800)^0.5
% Mean Bias Error
mbe=(sum(a-t'))/800
% Mean Absolute Percentage Error
mape=(sum(((a-t'))./t'))*100/800
% Error Analysis for Testing
a1=sim(net,ip')
% Root Mean Square Error
rmse1=((sum((a1-t1').^2))/52)^0.5
% Mean Bias Error
mbe1=(sum(a1-t1'))/52
% Mean Absolute Percentage Error
mape1=(sum(((a1-t1'))./t1'))*100/52
plot(a1,'r')
hold all
plot(t1','b')
```