**Program 5: With a suitable example demonstrate the perceptron learning with its decision region using MATLAB. Give the output in graphical form.**

clear

p=5;

N=50;

X=2\*rand(p-1,2\*N)-1;

nn=round((2\*N-1)\*rand(N,1))+1;

X(:,nn)=sin(X(:,nn));

X=[X;ones(1,2\*N)];

wht=3\*rand(1,p)-1; wht=wht/norm(wht);

wht;

D=(wht\*X>=0);

Xv=X(:, N+1:2\*N);

Dv=D(:,N+1:2\*N);

X=X(:,1:N);

D=D(:,1:N);

%[X;D]

pr=[1,3];

Xp=X(pr,:);

wp=wht([pr p]);

c0=find(D==0); c1=find(D==1);

figure(1),clf reset

plot(Xp(1,c0),Xp(2,c0),'o',Xp(1,c1),Xp(2,c1),'X')

axis(axis),hold on

L=[-1 1];

S=-diag([1 1]./wp(1:2.))\*(wp([2,1])'\*L+wp(3));

plot([S(1,:) L],[L S(2,:)]), grid, drawnow

%PART 2:Learning

eta=0.5;%The training gain

wh=2\*rand(1,p)-1;

wp=wh([pr p]);

S=-diag([1 1]./wp(1:2))\*(wp([2,1])'\*L +wp(3));

plot([S(1,:) L],[L S(2,:)]),grid on, drawnow

C=50;

E=[C+1,zeros(1,C)];

WW=zeros(C\*N,p);

c=1;

cw=0;

while(E(c)>1)||(c==1)

c=c+1;

plot([S(1,:) L],[L S(2,:)],'w'),drawnow;

for n=1:N

eps=D(n)-((wh\*X(:,n))>=0);%eps(n)=d(n)-y(n)

wh=wh+eta\*eps\*X(:,n)';

cw=cw+1;

WW(cw,:)=wh/norm(wh);

E(c)=E(c)+abs(eps);

end;

wp=wh([pr p]);

S=-diag([1 1]./wp(1:2))\*(wp([2 1])'\*L+wp(3));

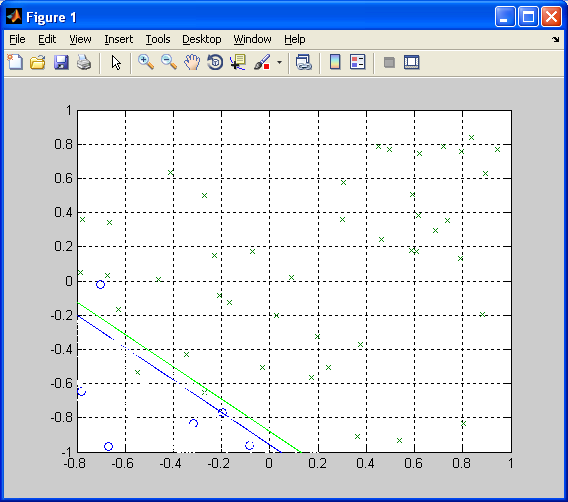
plot([S(1,:) L],[L S(2,:)],'g'),drawnow

end;

WW=WW(1:cw,pr);

E=E(2:c+1);

**Output:**



**Program 6: Write a suitable example simulate the perceptron learning network and separate the boundaries. Plot the points assume in respective quadrants using different symbols for identification.**

clear;

p1=[1 1]'; p2=[1 2]';

p3=[2 -1]'; p4=[2 -2]';

p5=[-1 2]'; p6=[-2 1]';

p7=[-1 1]'; p8=[-2 -2]';

hold on

plot(p1(1),p1(2),'ks',p2(1),p2(2),'ks',p3(1),p3(2),'ko',p4(1),p4(2),'ko')

plot(p5(1),p5(2),'k\*',p6(1),p6(2),'k\*',p7(1),p7(2),'kd',p8(1),p8(2),'kd')

grid

hold

axis([-3 3 -3 3])

t1=[0 0]'; t2=[0 0]';

t3=[0 1]'; t4=[0 1]';

t5=[1 0]'; t6=[1 0]';

t7=[1 1]'; t8=[1 1]';

R=[-2 2;-2 2];

netp=newp(R,2);

P=[p1 p2 p3 p4 p5 p6 p7 p8];

T=[t1 t2 t3 t4 t5 t6 t7 t8];

Y = sim(netp, P )

netp.trainParam.epochs = 20;

netp = train(netp,P,T);

Y1=sim(netp,P)

W=netp.IW{1,1}

B=netp.b{1}

x=[-3:0.01:3];

y=-W(1,1)/W(1,2)\*x-B(1)/W(1,2);

y1=-W(2,1)/W(2,2)\*x-B(2)/W(2,2);

figure

hold on

plot(p1(1),p1(2),'ks',p2(1),p2(2),'ks',p3(1),p3(2),'ko',p4(1),p4(2),'ko')

plot(p5(1),p5(2),'k\*',p6(1),p6(2),'k\*',p7(1),p7(2),'kd',p8(1),p8(2),'kd')

grid

axis([-3 3 -3 3])

plot(x,y,'r',x,y1,'b')

hold off

p9=[1 0.05]'; p10=[0.05 1]';

t9=t1; t10=t2;

p11=[1 -0.05]'; p12=[0.05 -1]';

t11=t3; t12=t4;

p13=[-1 0.05]'; p14=[-0.05 1]';

t13=t5; t14=t6;

p15=[-1 -0.05]'; p16=[-0.05 -1]';

t15=t7; t16=t8;

R=[-2 2;-2 2];

netp=newp(R,2,'hardlim','learnp');

P=[p1 p2 p3 p4 p5 p6 p7 p8 p9 p10 p11 p12 p13 p14 p15 p16];

T=[t1 t2 t3 t4 t5 t6 t7 t8 t9 t10 t11 t12 t13 t14 t15 t16];

Y=sim(netp,P);

netp.trainParam.epochs=5000;

netp=train(netp,P,T);

Y1=sim(netp,P);

C=norm(Y1-T)

W=netp.IW{1,1}

B=netp.b{1}

x=[-3:0.01:3];

y=-W(1,1)/W(1,2)\*x-B(1)/W(1,2);

y1=-W(2,1)/W(2,2)\*x-B(2)/W(2,2);

figure

hold on

plot(p1(1),p1(2),'ks',p2(1),p2(2),'ks',p3(1),p3(2),'ko',p4(1),p4(2),'ko')

plot(p5(1),p5(2),'k\*',p6(1),p6(2),'k\*',p7(1),p7(2),'kd',p8(1),p8(2),'kd')

plot(p9(1),p9(2),'ks',p10(1),p10(2),'ks',p11(1),p11(2),'ko',p12(1),p12(2),'ko')

plot(p13(1),p13(2),'k\*',p14(1),p14(2),'k\*',p15(1),p15(2),'kd',p16(1),p16(2),'kd')

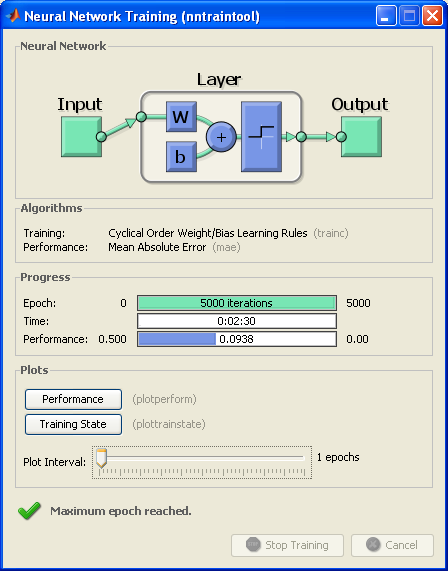
grid

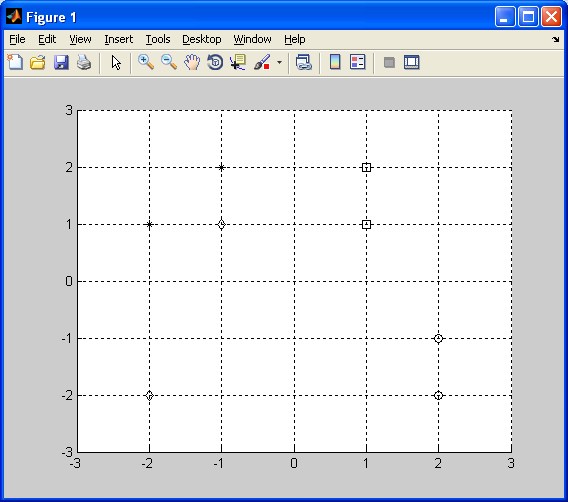
axis([-3 3 -3 3])

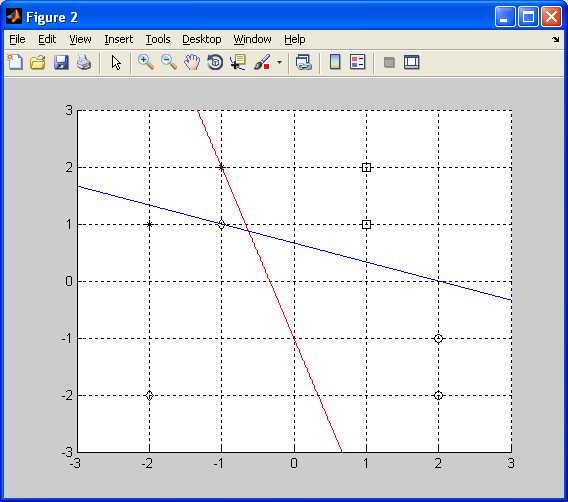
plot(x,y,'r',x,y1,'b')

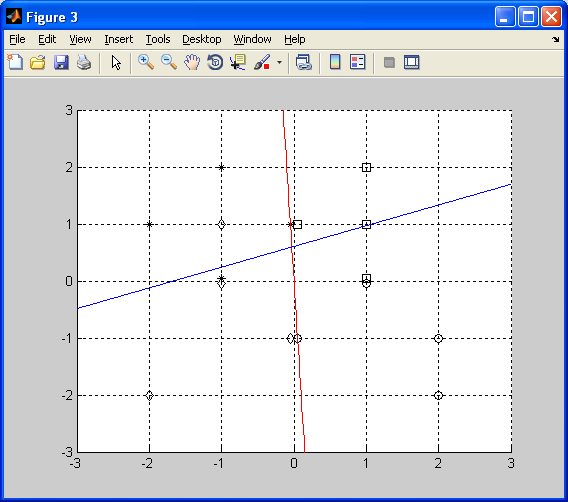
hold off

**Output:**









**Program 7: Write a MATLAB program to recognize the number 0, 1, 2, 3 ….9. A 5×3 matrix forms the numbers. For any valid point it is taken as 1 and invalid point it is taken as 0. The net has to be trained to recognize all the numbers and when the test data is given, the network has to recognize the particular numbers.**

clear;

clc;

input=xlsread('book2','input')

output=xlsread('book2','output')

test=xlsread('book2','test')

net=newp(input,output);

net.trainparam.epochs=1000;

net.trainparam.goal=0;

net=train(net,input,output);

y=sim(net,test);

x=y';

for i=1:5

k=0;

l=0;

for j=1:10

if x(i,j)==1

k=k+1;

l=j;

end

end

if k==1

s=sprintf('Test Pattern %d is Recognised as %d',i,l-1);

disp(s);

else

s=sprintf('Test Pattern %d is Not Recognised',i);

disp(s);

end

end

**Output:**

input =

1 0 1 1 1 1 1 1 1 1

1 1 1 1 0 1 1 1 1 1

1 0 1 1 1 1 1 1 1 1

1 1 0 0 1 1 1 0 1 1

0 1 0 0 0 0 0 0 0 0

1 0 1 1 1 0 0 1 1 1

1 0 1 1 1 1 1 0 1 1

0 1 1 1 1 1 1 0 1 1

1 0 1 1 1 1 1 1 1 1

1 0 1 0 0 0 1 0 1 0

0 1 0 0 0 0 0 0 0 0

1 0 0 1 1 1 1 1 1 1

1 1 1 1 0 1 1 0 1 1

1 1 1 1 0 1 1 0 1 1

1 1 1 1 1 1 1 1 1 1

output =

1 0 0 0 0 0 0 0 0 0

0 1 0 0 0 0 0 0 0 0

0 0 1 0 0 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0

0 0 0 0 0 1 0 0 0 0

0 0 0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 0 0 0 1

test =

1 0 1 1 1

1 1 1 1 0

1 1 1 1 1

1 1 0 0 1

0 1 0 0 1

1 1 1 1 1

1 0 1 1 1

0 1 1 1 1

1 0 1 1 1

1 1 1 0 0

0 1 0 1 0

1 0 0 1 1

1 1 1 1 1

1 1 1 1 0

1 1 1 1 1

\*\* Warning in INIT

\*\* Network "input{1}.processedRange" has a row with equal min and max values.

\*\* Constant inputs do not provide useful information.

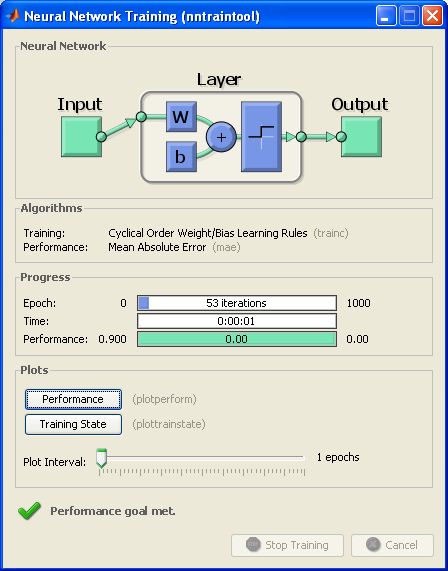
Test Pattern 1 is Recognised as 0

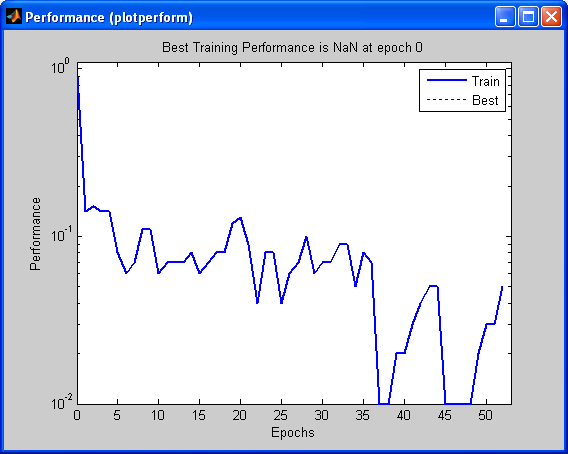
Test Pattern 2 is Not Recognised

Test Pattern 3 is Recognised as 2

Test Pattern 4 is Recognised as 3

Test Pattern 5 is Recognised as 4





**Program 8: Develop a MATLAB program for OR function with bipolar inputs and targets using Adaline Network.**

clear all;

clc;

disp('Adaline network for OR function Bipolar inputs and targets');

x1=[1 1 -1 -1];

x2=[1 -1 1 -1];

x3=[1 1 1 1];

t=[1 1 1 -1];

w1=0.1;w2=0.1;b=0.1;

alpha=0.1;

e=2;

delw1=0;delw2=0;delb=0;

epoch=0;

while(e>1.018)

epoch=epoch+1;

e=0;

for i=1:4

nety(i)=w1\*x1(i)+w2\*x2(i)+b;

nt=[nety(i) t(i)];

delw1=alpha\*(t(i)-nety(i)\*x1(i));

delw2=alpha\*(t(i)-nety(i)\*x2(i));

delb=alpha\*(t(i)-nety(i))\*x3(i);

wc=[delw1 delw2 delb]

w1=w1+delw1;

w2=w2+delw2;

b=b+delb;

w=[w1 w2 b]

x=[x1(i) x2(i) x3(i)];

pnt=[x nt wc w]

end

for i=1:4

nety(i)=w1\*x1(i)+w2\*x2(i)+b;

e=e+(t(i)-nety(i)^2);

end

end

**Output:**

Adaline network for OR function Bipolar inputs and targets

wc =

0.0700 0.0700 0.0700

w =

0.1700 0.1700 0.1700

pnt =

Columns 1 through 8

1.0000 1.0000 1.0000 0.3000 1.0000 0.0700 0.0700 0.0700

Columns 9 through 11

0.1700 0.1700 0.1700

wc =

0.0830 0.1170 0.0830

w =

0.2530 0.2870 0.2530

pnt =

Columns 1 through 8

1.0000 -1.0000 1.0000 0.1700 1.0000 0.0830 0.1170 0.0830

Columns 9 through 11

0.2530 0.2870 0.2530

wc =

0.1287 0.0713 0.0713

w =

0.3817 0.3583 0.3243

pnt =

Columns 1 through 8

-1.0000 1.0000 1.0000 0.2870 1.0000 0.1287 0.0713 0.0713

Columns 9 through 11

0.3817 0.3583 0.3243

wc =

-0.1416 -0.1416 -0.0584

w =

0.2401 0.2167 0.2659

pnt =

Columns 1 through 8

-1.0000 -1.0000 1.0000 -0.4157 -1.0000 -0.1416 -0.1416 -0.0584

Columns 9 through 11

0.2401 0.2167 0.2659

wc =

0.0277 0.0277 0.0277

w =

0.2679 0.2445 0.2936

pnt =

Columns 1 through 8

1.0000 1.0000 1.0000 0.7227 1.0000 0.0277 0.0277 0.0277

Columns 9 through 11

0.2679 0.2445 0.2936

wc =

0.0683 0.1317 0.0683

w =

0.3362 0.3762 0.3619

pnt =

Columns 1 through 8

1.0000 -1.0000 1.0000 0.3170 1.0000 0.0683 0.1317 0.0683

Columns 9 through 11

0.3362 0.3762 0.3619

wc =

0.1402 0.0598 0.0598

w =

0.4763 0.4360 0.4217

pnt =

Columns 1 through 8

-1.0000 1.0000 1.0000 0.4019 1.0000 0.1402 0.0598 0.0598

Columns 9 through 11

0.4763 0.4360 0.4217

wc =

-0.1491 -0.1491 -0.0509

w =

0.3273 0.2869 0.3708

pnt =

Columns 1 through 8

-1.0000 -1.0000 1.0000 -0.4906 -1.0000 -0.1491 -0.1491 -0.0509

Columns 9 through 11

0.3273 0.2869 0.3708