

Enter weights

weight w1=-1

weight w2=1

enter threshold value

theta=1

output of net

0 1 0 0

net is not learning enter another set of weights and threshold value

weight w1=-1

weight w2=-1

output of net

0 0 0 0

net is not learning enter another set of weights and threshold value

weight w1=1

weight w2=1

output of net

0 1 1 1

net is not learning enter another set of weights and threshold value

weight w1=1

weight w2=-1

output of net

0 0 1 0

mcculloh-pitts net for andnot function

weights of neuron

1

-1

threshold value

1

**6TH**

clear;

clc;

disp('1MS21EI035 NIDHI');

disp('Enter weights');

w11=input('Weight w11 = ');

w12=input('Weight w12 = ');

w21=input('Weight w21 = ');

w22=input('Weight w22 = ');

v1=input('Weight v1 = ');

v2=input('Weight v2 = ');

disp('Enter Threshold Value');

theta=input('Theta = ');

x1=[0 0 1 1];

x2=[0 1 0 1];

z=[0 1 1 0];

con=1;

while con

zin1=x1\*w11 + x2\*w21;

zin2=x1\*w21 + x2\*w22;

for i=1:4

if zin1(i)>=theta

y1(i)=1;

else

y1(i)=0;

end

if zin2(i)>=theta

y2(i)=1;

else

y2(i)=0;

end

end

yin=y1\*v1+y2\*v2;

for i=1:4

if yin(i)>=theta

y(i)=1;

else

y(i)=0;

end

end

disp('Output of Net');

disp(y);

if y==z

con=0;

else

disp('Net is not learning. Enter another set of weights and Threshold Value');

w11=input('Weight w11 = ');

w12=input('Weight w12 = ');

w21=input('Weight w21 = ');

w22=input('Weight w22 = ');

v1=input('Weight v1 = ');

v2=input('Weight v2 = ');

disp('Enter Threshold Value');

theta=input('Theta = ');

end

end

disp('McCulloch-Pitts Net for XOR function');

disp('Weights of Neuron Z1:');

disp(w11);

disp(w21);

disp('Weights of Neuron Z2:');

disp(w12);

disp(w22);

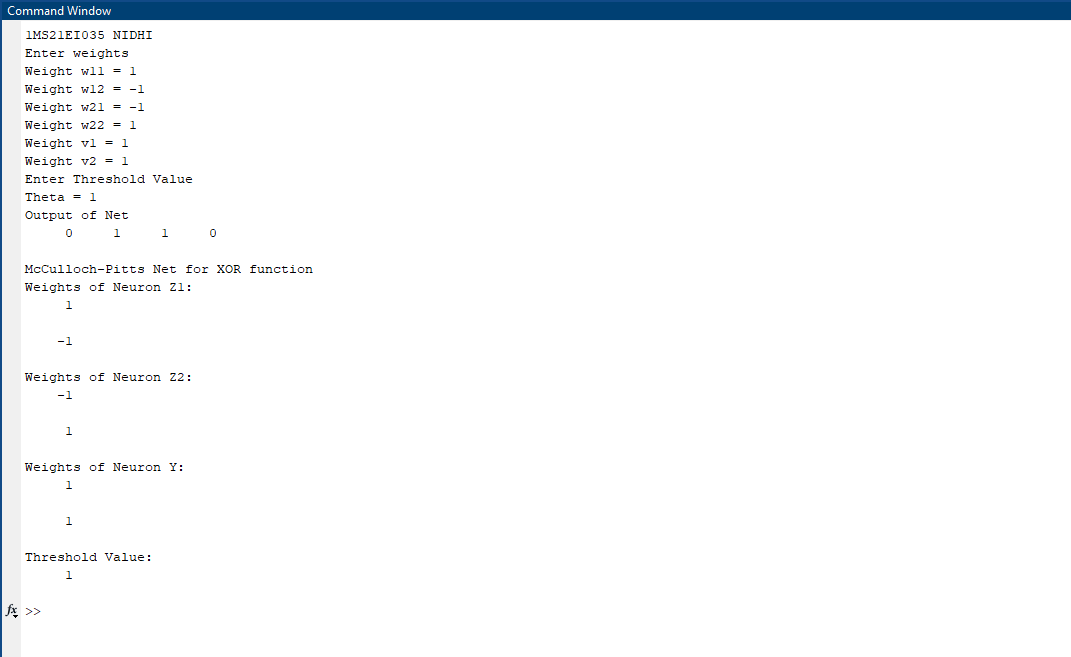
disp('Weights of Neuron Y:');

disp(v1);

disp(v2);

disp('Threshold Value:');

disp(theta);

clear;

clc;

disp('1MS21EI035 NIDHI')

x=[1 1 -1 -1; 1 -1 1 -1];

t=[1 -1 -1 -1];

w=[0 0];

b=0;

alpha=input('Enter the learning rate: ');

theta=input('Enter the threshold value: ');

con=1;

epoch=0;

while con

con=0;

disp("Epoch = " + (epoch+1));

for i=1:4

yin = b + (x(1,i)\*w(1))+(x(2,i)\*w(2));

disp("yin = " + yin);

if yin>theta

y=1;

end

if yin<=theta && yin>=-theta

y=0;

end

if yin<-theta

y=-1;

end

disp("y = " + y);

if y~=t(i)

con=1;

for j=1:2

w(j)=w(j)+alpha\*t(i)\*x(j,i);

end

b=b+alpha\*t(i);

end

end

epoch=epoch+1;

fprintf("\n");

end

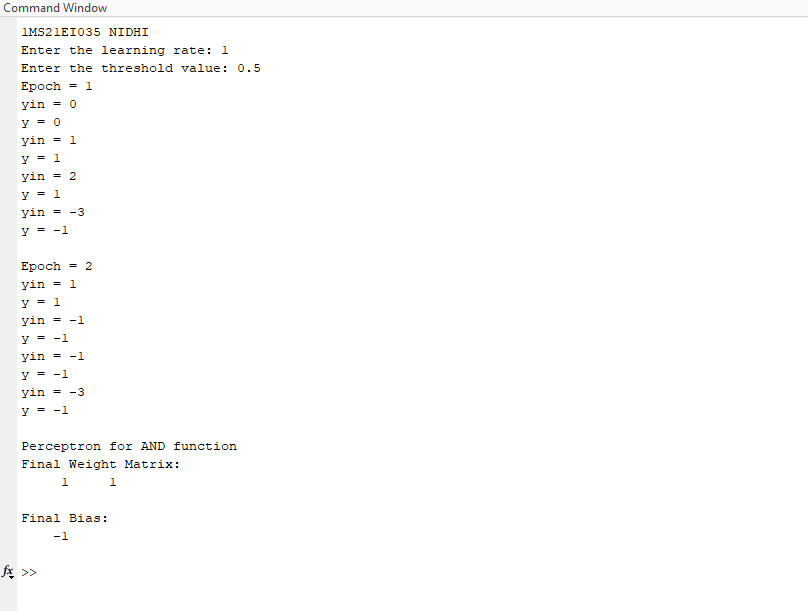
disp('Perceptron for AND function');

disp('Final Weight Matrix:')

disp(w);

disp('Final Bias:');

disp(b);

clear;

clc;

disp('1MS21EI035 NIDHI');

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

1MS21EI035 NIDHI

Adaline network for OR function Bipolar inputs and targets

wc =

0.0700 0.0700 0.0700

w =

0.1700 0.1700 0.1700

Pnt =

1.0000 1.0000 1.0000 0.3000 1.0000 0.0700 0.0700 0.0700 0.1700 0.1700 0.1700

wc =

0.0830 -0.0830 0.0830

w =

0.2530 0.0870 0.2530

Pnt =

1.0000 -1.0000 1.0000 0.1700 1.0000 0.0830 -0.0830 0.0830 0.2530 0.0870 0.2530

wc =

-0.0913 0.0913 0.0913

w =

0.1617 0.1783 0.3443

Pnt =

-1.0000 1.0000 1.0000 0.0870 1.0000 -0.0913 0.0913 0.0913 0.1617 0.1783 0.3443

wc =

0.1004 0.1004 -0.1004

w =

0.2621 0.2787 0.2439

Pnt =

-1.0000 -1.0000 1.0000 0.0043 -1.0000 0.1004 0.1004 -0.1004 0.2621 0.2787 0.2439

wc =

0.0215 0.0215 0.0215

w =

0.2837 0.3003 0.2654

Pnt =

1.0000 1.0000 1.0000 0.7847 1.0000 0.0215 0.0215 0.0215 0.2837 0.3003 0.2654

wc =

0.0751 -0.0751 0.0751

w =

0.3588 0.2251 0.3405

Pnt =

1.0000 -1.0000 1.0000 0.2488 1.0000 0.0751 -0.0751 0.0751 0.3588 0.2251 0.3405

**8TH**

clear;

clc;

%Input Signal; x(t);

f1 = 2; %KHz

ts = 1/(40\*f1) ; %12.5 usec -- sampling time

N = 100;

t1 = (0:N)\*4\*ts;

t2 = (0:2\*N)\*ts + 4\*(N+1)\*ts;

t = [t1 t2]; % 0 to 7.5 sec

N = size(t, 2); N = 302;

xt = [sin(2\*pi\*f1\*t1) sin(2\*pi\*2\*f1\*t2)];

plot(t, xt) ; grid; title('Signal to be predicted')

p = 4; %Number of synapses

% formation of the input matrix X of size p by N

% use the convolution matrix .Try convmt(1:8. 5)

x = convmtx(xt, p); X= x(:, 1:N);

d = xt ; % The target signal is equal to the input signal

y = zeros(size(d)); % memory allocation for y

eps = zeros(size(d)); % memory allocation for eps

eta = 0.4 ; % learning rate/gain

w = rand(1, p); % Initialisation of the weight vector

for n = 1:N %learning loop

y(n) = w\*X(:, n);

eps(n) = d(n) - y(n); % error signal

w = w+ eta\*eps(n)\*x(:,n)' ;

end

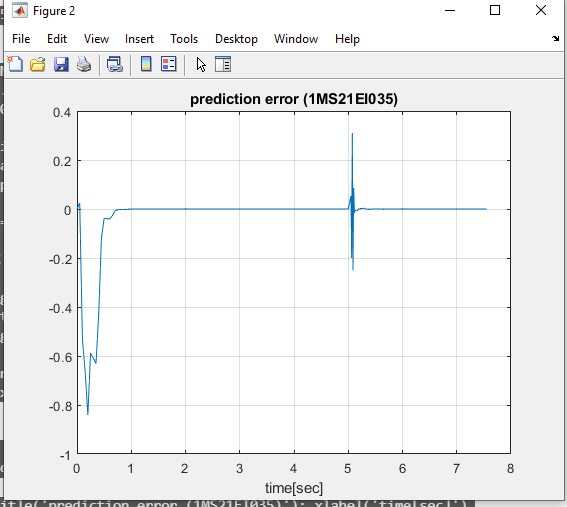
figure(1)

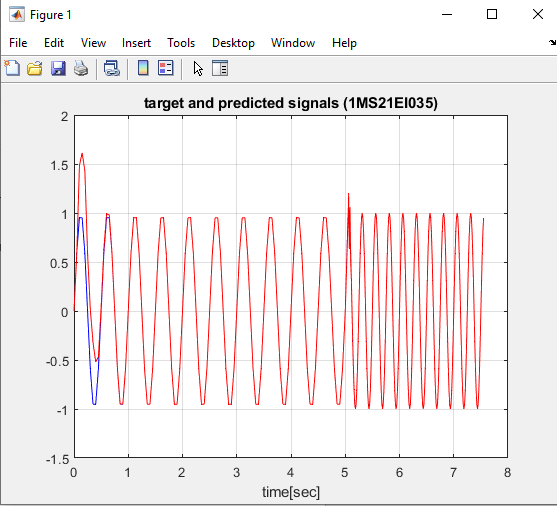
plot(t, d, 'b', t, y, '-r'); grid;

title('target and predicted signals (1MS21EI035)'); xlabel('time[sec]')

figure(2)

plot(t, eps); grid; title('prediction error (1MS21EI035)'); xlabel('time[sec]')





**9TH**

%Back propagation

clc;

clear;

disp('NIDHI 1MS21EI035');

%initialize weights

v=[0.197 0.3191 -0.1448 0.3394;0.3099 0.1904 -0.0347 -0.4861];

v1=zeros(2,4);

b1=[-0.3378 0.2771 0.2859 -0.3329];

b2=-0.1401;

w=[0.4919;-0.2913;-0.3979;0.3581];

w1=zeros(4,1);

x=[1 1 0 0;1 0 1 0];

t=[0 1 1 0];

alpha=0.02;

mf=0.9;

con=1;

epoch=0;

while con

e=0;

for I=1:4

%feed forward

for j=1:4

zin(j)=b1(j);

for i=1:2

zin(j)=zin(j)+x(i,I)\*v(i,j);

end

z(j)=binsig(zin(j));

end

yin=b2+z\*w;

y(I)=binsig(yin);

%Backprop

delk=(t(I)-y(I))\*binsig1(yin);

delw=alpha\*delk\*z'+mf\*(w-w1);

delb2=alpha\*delk;

delinj=delk\*w;

for j=1:4

delj(j,1)=delinj(j,1)\*binsig1(zin(j));

end

for j=1:4

for i=1:2

delv(i,j)=alpha\*delj(j,1)\*x(i,I)+mf\*(v(i,j)-v1(i,j));

end

end

delb1=alpha\*delj;

w1=w;

v1=v;

w=w+delw;

b2=b2+delb2;

v=v+delv;

b1=b1+delb1';

e=e+(t(I)-y(I))^2;

end

if e<0.005

con=0;

end

epoch=epoch+1;

end

disp('BPN for XOR function with Binary input and output');

disp('Total Epoch performed');

disp(epoch);

disp('error');

disp(e);

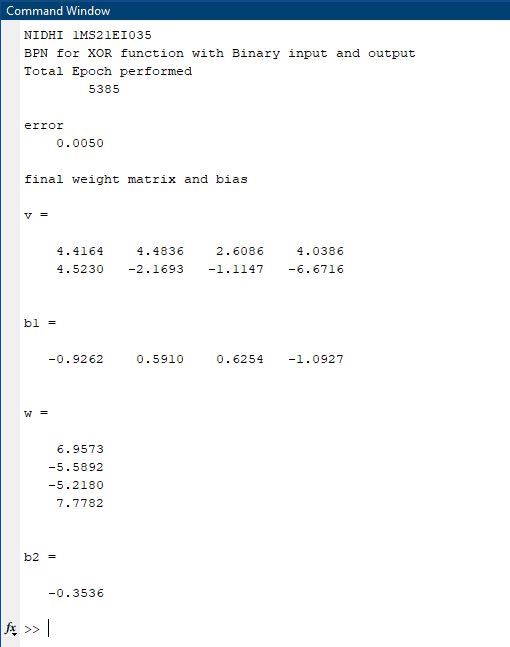
disp('final weight matrix and bias');

v

b1

w

b2



clc;

clear;

disp('NIDHI 1MS21EI035');

x=[1 1 1 0];

tx=[0 0 1 0];

w=(2\*x'-1)\*(2\*x-1);

for i=1:4

w(i,i)=0;

end

con=1;

y=[0 0 1 0];

while con

up=[4 2 1 3];

for i=1:4

yin(up(i))=tx(up(i))+y\*w(1:4,up(i));

if yin(up(i))>0

y(up(i))=1;

end

end

if y==x

disp('Convergence has been obtained');

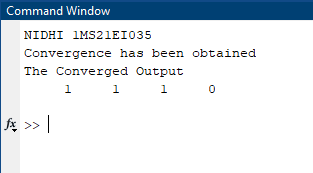
disp('The Converged Output');

disp(y);

con=0;

end

end



clc;

clear;

x=[1 1 -1 -1;1 -1 1 -1];

t=[-1 1 1 -1];

w=[0.05 0.1;0.2 0.2];

b1=[0.3 0.15];

v=[0.5 0.5];

b2=0.5;

con=1;

alpha=0.5;

epoch=0;

while con

con=0;

for i=1:4

for j=1:2

zin(j)=b1(j)+x(1,i)\*w(1,j)+x(2,i)\*w(2,j);

if zin(j)>=0

z(j)=1;

else

z(j)=-1;

end

end

yin=b2+z(1)\*v(1)+z(2)\*v(2);

if yin>=0

y=1;

else

y=-1;

end

if y~=t(i)

con=1;

if t(i)==1

if abs(zin(1))> abs(zin(2))

k=2;

else

k=1;

end

b1(k)=b1(k)+alpha\*(1-zin(k));

w(1:2,k)=w(1:2,k)+alpha\*(1-zin(k))\*x(1:2,i);

else

for k=1:2

if zin(k)>0

b1(k)=b1(k)+alpha\*(-1-zin(k));

w(1:2,k)=w(1:2,k)+alpha\*(-1-zin(k))\*x(1:2,i);

end

end

end

end

end

epoch=epoch+1;

end

disp("NIDHI 1MS21EI035");

disp('Weight matrix of hidden layer');

disp(w);

disp('Bias of hidden layer');

disp(b1);

disp('Total Epoch');

disp(epoch);

