SAGNIK BASU 113EC0199

QUESTION

Communication Channel Equalizer using Perceptron

Design a perceptron/ adaline with suitable activation function for communication channel equalization as detailed in the accompanying document.

MATLAB CODE

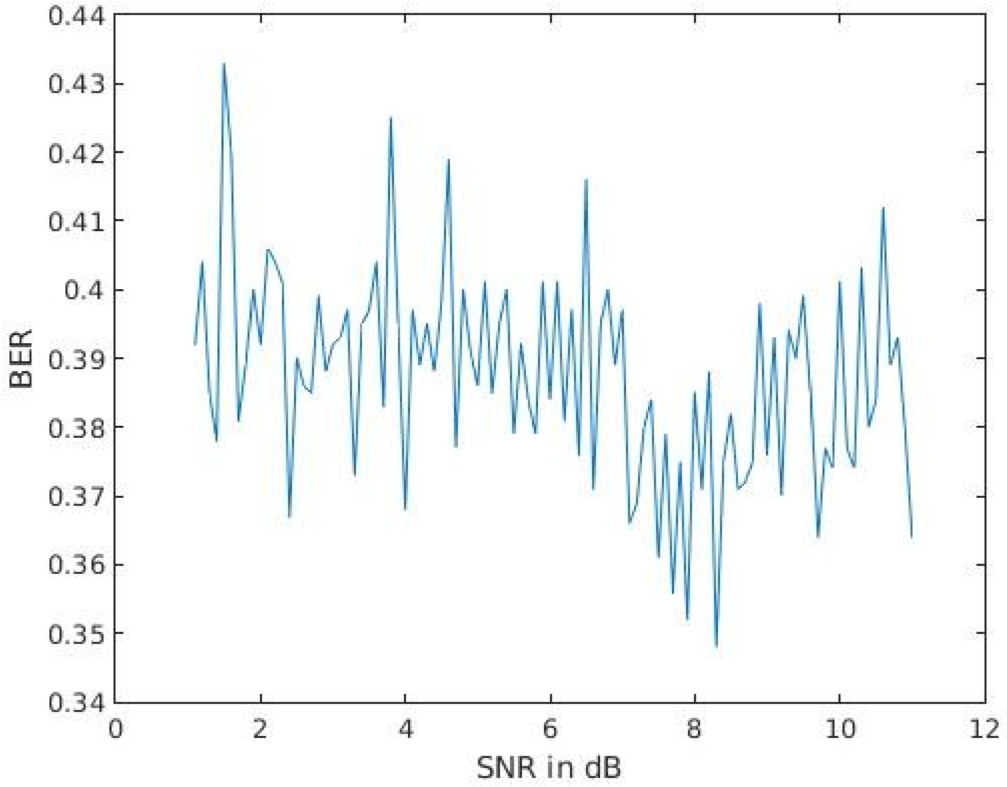
```
clc;
clear all;
close all;
c=input('Channel order');
experiments =50.0;

samples=1000;
x=2*rand(1,samples)-1;
inp=zeros(1,samples);
for i=1:length(x)    %%generation of inputs
    if(x(i)<0)
        inp(i)=-1;
    else if(x(i)>0)
            inp(i)=1;
        else
            inp(i)=0;
        end
    end
end


%noise=2*rand(1,samples)-1;  %%noise(bias)
%%channel 1
%y1=inp+noise;
SNR=20;
y1=awgn(inp,SNR);
weights=2*(rand(1,c))-1;
bias=2*rand(1,1)-1;   %%bias for the perceptron
y=zeros(1,samples);
output=zeros(1,samples);
error=zeros(1,samples);
```

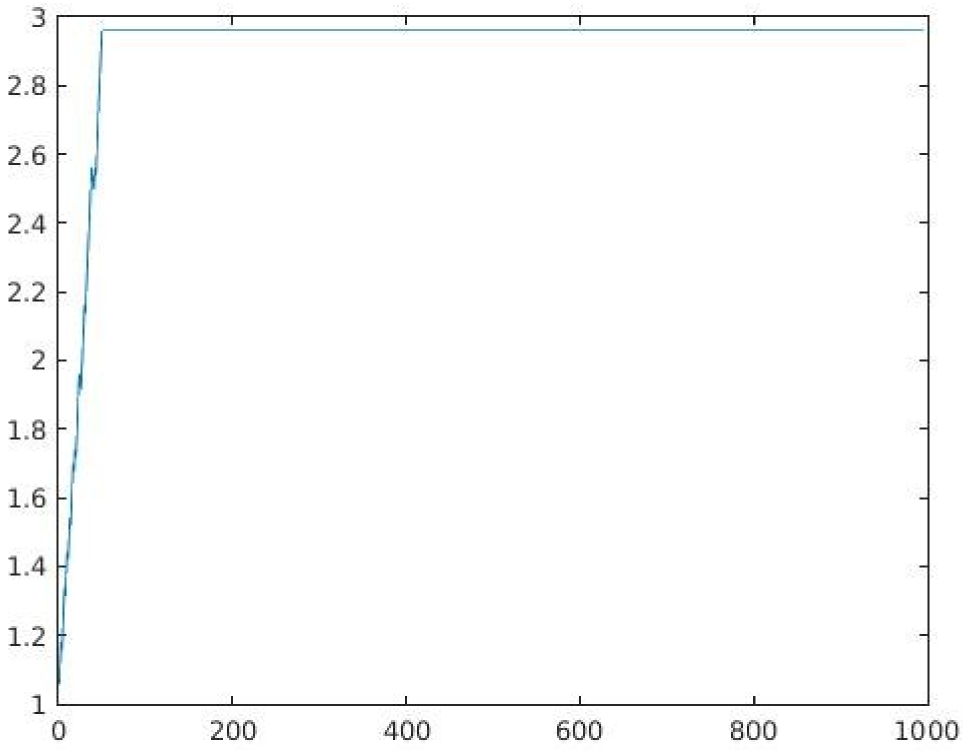
```
err_train=zeros(1,samples);
%%final_err_train=0;




for j=1:samples-c
                                %% input1(:,j)=input(:,r);
 y(1,j)=y1(1,j:j+c-1)*(transpose(weights))+bias;
                                %%out(1,j) = (1/(1+exp(-y(1,j))));
                                %%e=d_out(r)-out(j);
 output(1,j)=hardlims(y(1,j));

 %MSE Calculation for 50 experiments
 final_err_mse2=0;



 for k=1:experiments
     y_mse2(1,k)=y1(1,k:k+c-1)*(transpose(weights))+bias;
     output_mse2(1,k)=hardlims(y(1,k));
     error_mse2(1,k)=inp(1,k)-output(1,k);
     final_err_mse2=final_err_mse2+error_mse2(1,k)*error_mse2(1,k);
 end

 mse_2(j)=final_err_mse2/experiments;




 %%training
 error(1,j)=inp(1,j)-output(1,j);
 %%err_train(j)=error(1,j)*error(1,j);
 bias=bias+error(1,j);
 weights=weights+error(1,j)*y1(1,k:k+c-1);
 weights_array(j,:)=weights;
 %%weights_final(j,:,k)= weights;
 %%bias_final(j,1,k) = bias;
 %%error(j,1,k) = e;

end

%%Testing
testing_size=1000;
y_test=2*rand(1,testing_size)-1;
input=zeros(1,50);
final_err=0;
mse=zeros(1,50);
SNR=1;
for k=1:100
    y_test=2*rand(1,testing_size)-1;
    input=zeros(1,testing_size);
for i=1:length(y_test)    %%generation of inputs
    if(y_test(i)<0)
        input(i)=-1;
    else if(y_test(i)>0)
            input(i)=1;
        else
            input(i)=0;
        end
    end
end
final_err=0;
```

```
SNR_arr(k)=SNR+k/10;
y1=awgn(input,SNR_arr(k));
BER=0;
for i=1:testing_size-c+1

    y1_test(1,i)=y1(1,i:i+c-1)*(transpose(weights))+bias;
    percp_out(1,i)=hardlims(y1_test(1,i));
    error_test(i)=percp_out(1,i)-input(1,i);
    if(error_test(i)==0)
    else
        BER=BER+1;
    end

    %final_err=final_err+error_test(i)*error_test(i);
BER_arr(k)=BER/1000;
end
%mse(k)=final_err/1000.0;

end



%axis([-3 3 -3 3]);
%w1=-bias/weights(1,1);
%w2=-bias/weights(1,2);
%plot([w1,0],[0,w2]);
%hold on;




%plotpv(inp,inp);
%hold on;
%plotpc(weights,bias);
```

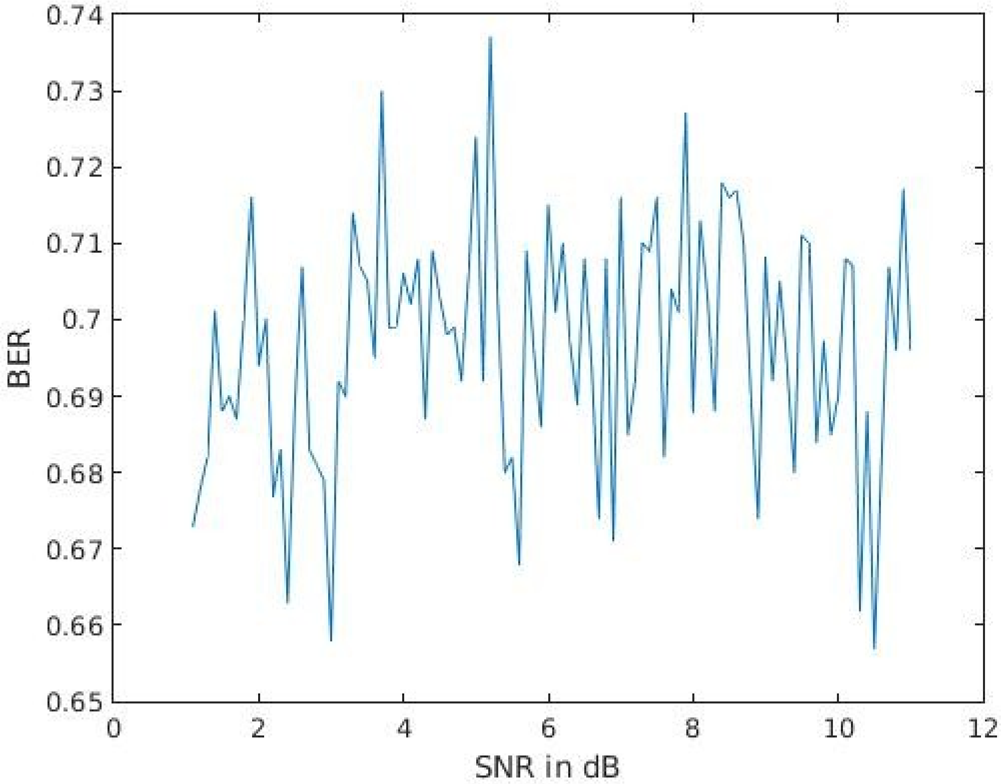
FIGURES (MSE Plot and BER Plot)

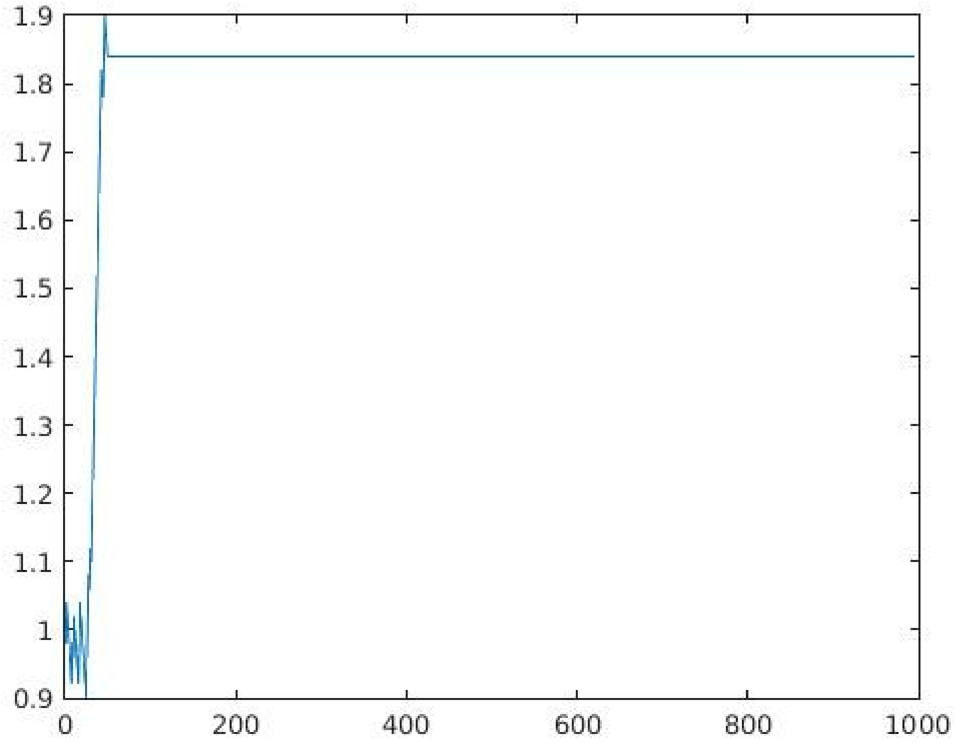
1) Channel Order =4



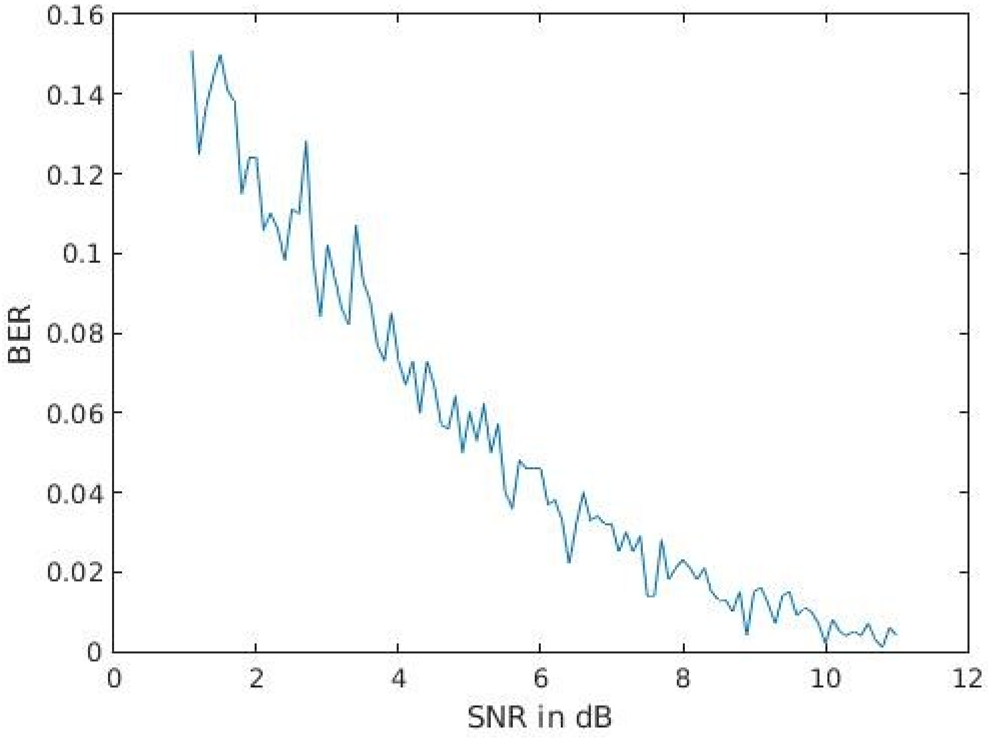


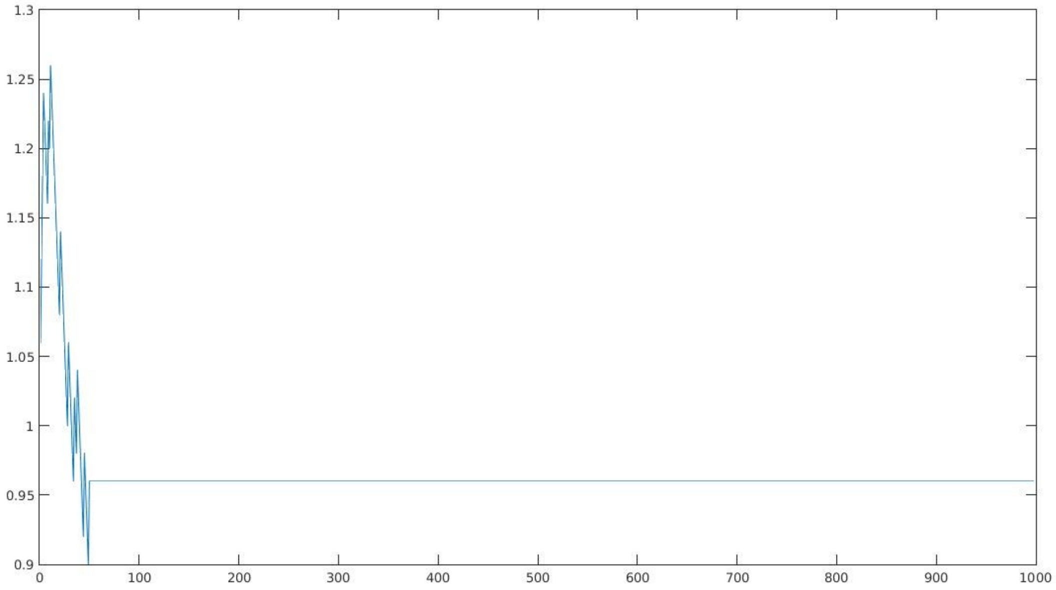
Channel Order =3





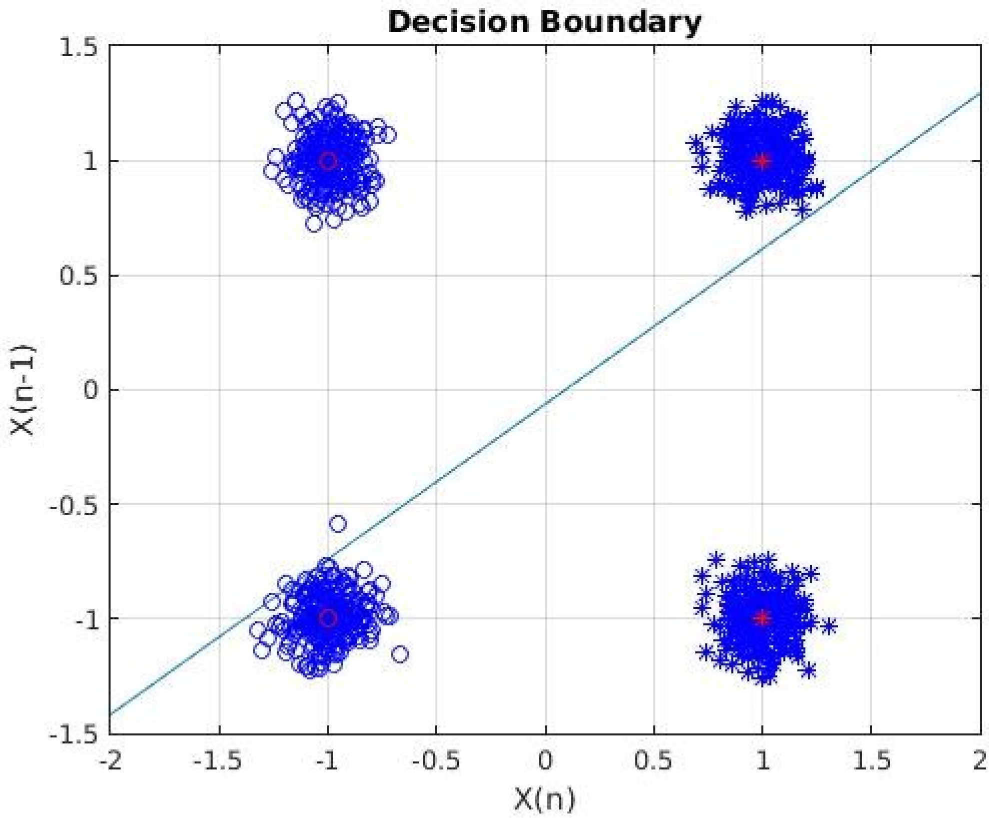
3)Channel Order =2



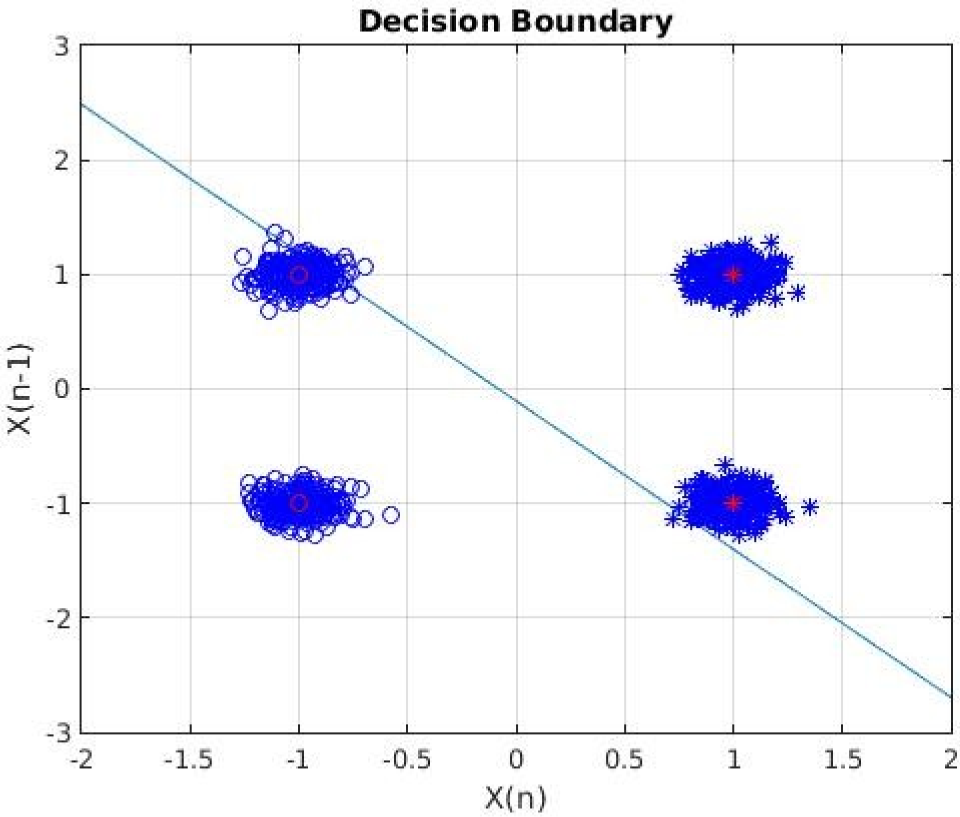


Decision Boundary for various SNR

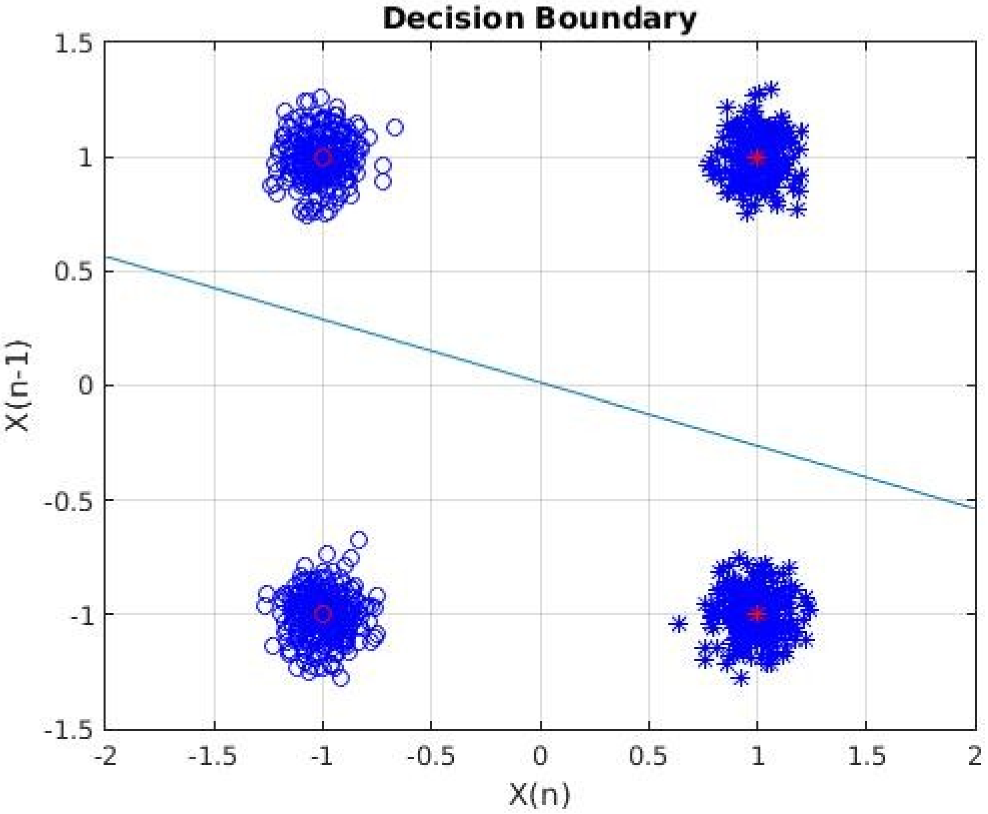
1) SNR = 5 Db


Decision Boundary
1.5
1
0.5
0
-0.5
-1
-1.5
X(n-1)
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
X(n)

2) SNR =10 Db


Decision Boundary
3
2
1
0
-1
-2
-3
X(n-1)
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2
X(n)

3)SNR = 15 Db


Decision Boundary
X(n-1)
X(n)