Name: Joshi Patanjalikumar S.

UTA Id: 1000858838

SUB: EE5322

Date: 4/5/2014

**Homework 4**

**1. Classification:-**

**a) Using a Perceptron:**

* MATLAB Code:

clc;clear all;close all;

P=[5 4 6 5 2 1 2 1;0 -1 0 -1 -1 -2 -2 -3]; %Input

T=[0 0 0 0 1 1 1 1]; %Target

perceptNN=newp(P,T); %Build a perceptron NN

perceptNN=train(perceptNN,P,T); %Train the NN

W=perceptNN.IW{1,1}; %NN weights

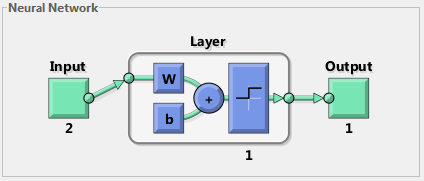
B=perceptNN.b{1}; %NN bias

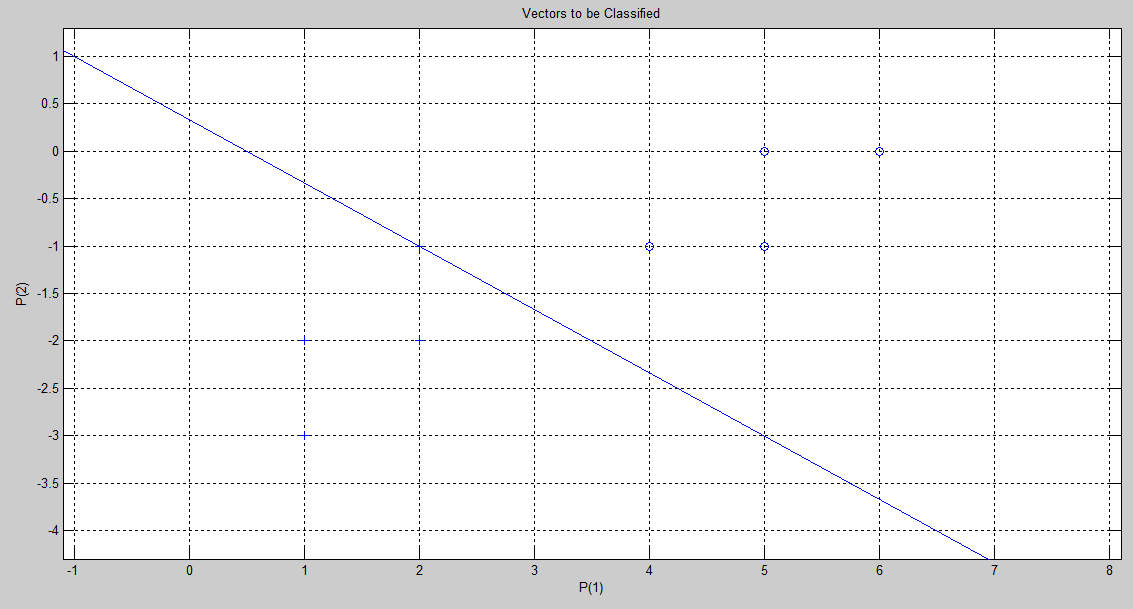
plotpv(P,T) %Plot perceptron input & output vectors

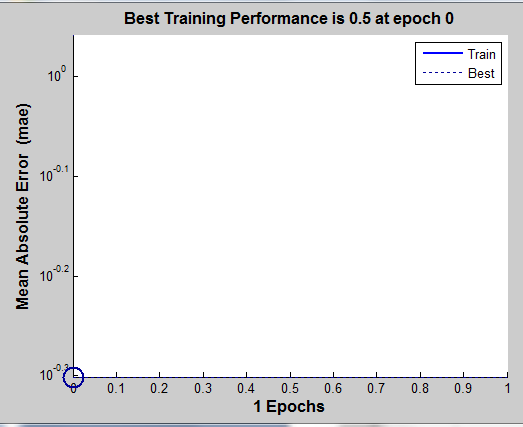
plotpc(W,B) %Plot classification line on perceptron vector plot

grid on;

* Output:

****

****

****

**b) Using Sigmoid activation function:-**

* MATLAB Code:

%Single neuron with sigmoid activation function

clc;clear all;close all;

P=[5 4 6 5 2 1 2 1;0 -1 0 -1 -1 -2 -2 -3]; %Input

T=[0 0 0 0 1 1 1 1]; %Target

sigNN=newp(P,T); %Build a perceptron NN

sigNN.layers{1}.transferFcn = 'logsig'; %Select sigmoid activation function

sigNN=train(sigNN,P,T); %Train the NN

W=sigNN.IW{1,1}; %NN weights

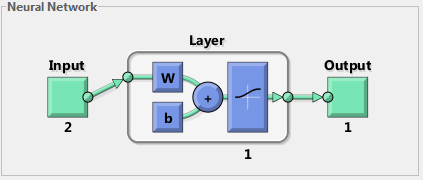
B=sigNN.b{1}; %NN bias

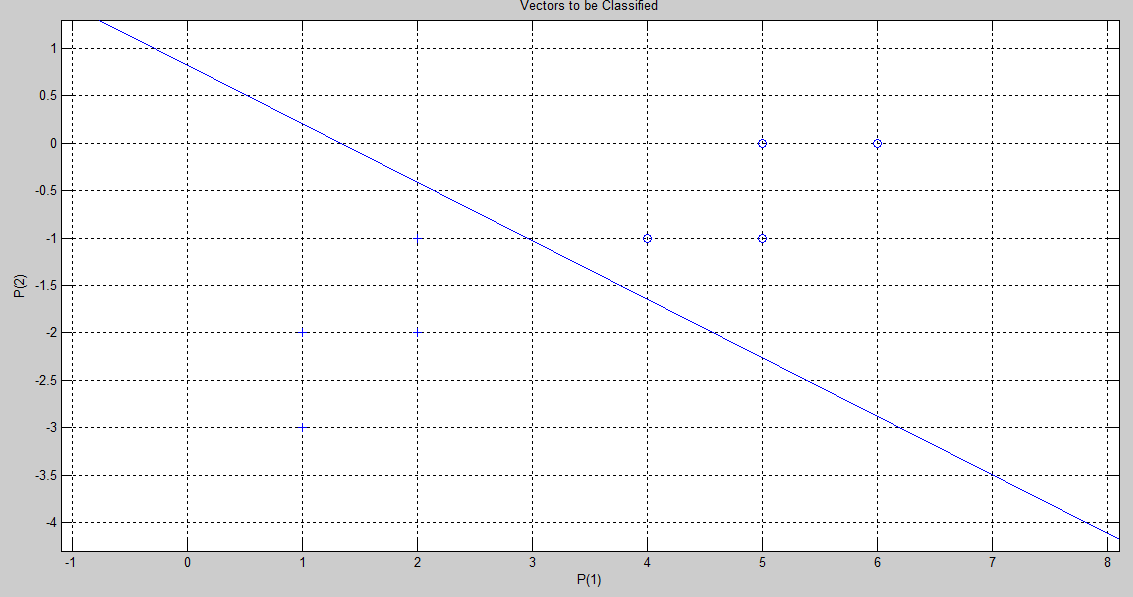
plotpv(P,T) %Plot perceptron input & output vectors

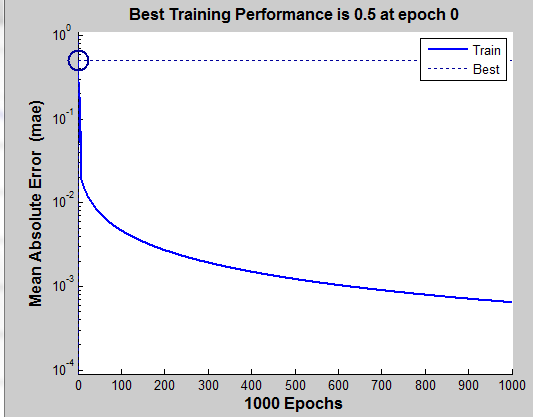
plotpc(W,B) %Plot classification line on perceptron vector plot

grid on;

* Output:

****

****

****

->When compared, looks like the single neuron with sigmoid activation function classifies better than the a perceptron-an input data (2,-1) lies on the classifier line.

**2. Function Approximation:-**

**a)**

****

* MATLAB Code:

%Function approximation using MLP

clc;clear all;close all;

t=-6\*pi:.05:6\*pi;

P=[rand(1,length(t));t]; %Input vector

T=t.^2+sin(t); %Target vector

MLPNN=feedforwardnet(10); %Building MLP with 10 hidden neurons

MLPNN.trainParam.goal= 0.0;

MLPNN=train(MLPNN,P,T); %Train the NN

Y=sim(MLPNN,P); %Verify the output

plot(t,T,t,Y)

title('Function Approximation using MLP NN')

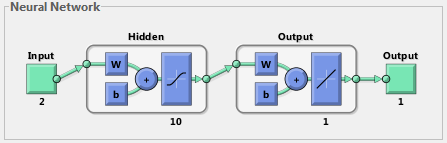
xlabel('Input x');

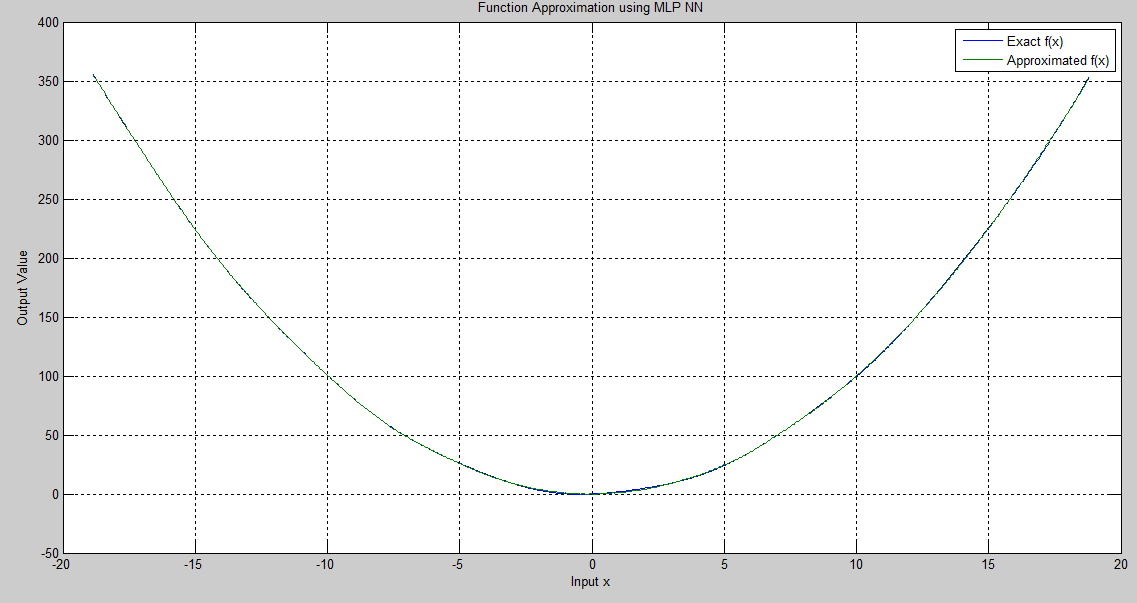
ylabel('Output Value');

grid on;

legend('Exact f(x)','Approximated f(x)');

* Output:-





**b) Response of system with exact f(x) and estimated f(x):**

* MATLAB Code:

clc;clear all;close all;

%System response for the exact f(x)

z0=[0;1];

[t,z]=ode23('funcap',[0 5],z0);

%subplot 311

plot(t,z(:,1))

title('System Response for the Exact f(x)');

xlabel('Time');

ylabel('System Response');

grid on;

%System response for the estimated f(x)

a=MLPApprox(1);

function a=MLPApprox(q)

t1=-2\*pi:.05:2\*pi;

P1=[rand(1,length(t1));t1]; %Input vector

T=t1.^2+sin(t1); %Target vector

MLPNN=feedforwardnet(10); %Building MLP with 10 hidden neurons

MLPNN.trainParam.goal= 0.0;

MLPNN=train(MLPNN,P1,T); %Train the NN

z10=[0;1];

function zdot=funcap3(t,z)

zdot=zeros(2,1);

P=[z(2);t];

%Y=MLPNN(z(2)); %Verify the output

zdot=[-.1\*(z(2)^3)+sim(MLPNN,P); z(1)];

end

[t,z1]=ode23(@funcap3,[0 5],z10);

figure;

plot(t,z1(:,1))

title('System Response for the Estimated f(x)');

xlabel('Time');

ylabel('System Response');

grid on;

a=q;

end

* Output:

****

****

**-**>It can be seen from the graphs above that the function f(x) has been exactly approximated by the multilayer perceptron; but the system response couldn’t be followed.

**3. Breast Cancer Data Classification using MLP:-**

* MATLAB Code:

clc;clear all;close all;

%Breast-cance data classification using MLP

data=open('breastcancerDataforHW4.mat');

P=data.data(:,2:10)'; %Input matrix

T=data.data(:,11)'; %Target values

total\_benign=0;

total\_mali=0;

for i=1:length(T)

if T(i)==2

T(i)=1; %Benign cell

total\_benign=total\_benign+1;

else

T(i)=-1; %Malignant cell

total\_mali=total\_mali+1;

end

end

cancernet=feedforwardnet(10); %Building MLP with 10 hidden neurons

cancernet.divideParam.trainRatio = 70/100;

cancernet.divideParam.valRatio = 15/100;

cancernet.divideParam.testRatio = 15/100;

cancernet=train(cancernet,P,T); %Train the NN

output=hardlims(sim(cancernet,P));

incorrectMalign=0; %No of benign cells incorrectly classified as “malignant”

incorrectBenign=0; %No of malignant cells incorrectly classied as “benign”

for i=1:length(T)

if T(i)==1 && output(i)==-1

incorrectMalign=incorrectMalign+1;

end

if T(i)==-1 && output(i)==1

incorrectBenign=incorrectBenign+1;

end

end

FalsePositiveRate=incorrectMalign/total\_benign

FalseNegativeRate=incorrectBenign/total\_mali

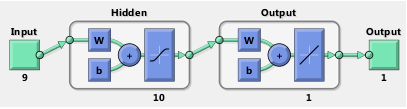
* **Output:**

FalsePositiveRate =

0.0225

FalseNegativeRate =

0.0293



**4. Breast Cancer Data Classification using Self-Organizing Map:**

* MATLAB Code:

clc;clear all;close all;

%Breast-cance data classification using Self-Organizing Map

data=open('breastcancerDataforHW4.mat');

P=data.data(:,2:10)'; %Input matrix

T=data.data(:,11)'; %Target values

total\_benign=0;

total\_mali=0;

for i=1:length(T)

if T(i)==2

%T(i)=1; %Benign cell

total\_benign=total\_benign+1;

else

%T(i)=0; %Malignant cell

total\_mali=total\_mali+1;

end

end

cancerSOM=selforgmap([8 8],478); %Building SOM

cancerSOM=train(cancerSOM,P); %Train the NN

%output=sim(cancerSOM,P);

output=cancerSOM(P);

classes=vec2ind(output);

* Output:

