Data mining: Assignment 2

1.Training the given data set in a Neural Network(Perceptron training and testing):

Code Snippet:

data=load('data.tsv');

n\_in = data';

t = label';

accuracy=zeros(1,10);

index=zeros(10000,10);

label(1:5000,1)=1;

label(5001:10000,1)=0;

label(1:5000,2)=0;

label(5001:10000,2)=1;

cv=cvpartition(10000,'kfold',10);

for i=1:1:10

index(:,i)=test(cv,i);

end

trainid=zeros(10,9000);

testid=zeros(10,1000);

for n=1:1:10

a=1;

b=1;

for j=1:1:10000

if(index(j,n)==0)

trainid(n,a)=j;

a=a+1;

else

testid(n,b)=j;

b=b+1;

end

end

end

% Create a Pattern Recognition Network

hiddensize = 10;

nn = patternnet(hiddensize);

% Choose Input and Output Pre/Post-Processing Functions

nn.input.processFcns = {'removeconstantrows','mapminmax'};

nn.output.processFcns = {'removeconstantrows','mapminmax'};

for i=1:1:10

% Setup Division of Data for Training, Validation, Testing

nn.divideFcn = 'divideind'; % Divide data by K-Fold cross validation

nn.divideMode = 'sample'; % Divide up every sample

nn.divideParam.trainInd=trainid(i,1:8000);

nn.divideParam.valInd=trainid(i,8001:9000);

nn.divideParam.testInd=testid(i,:);

% For a list of all training functions type: help nntrain

nn.trainFcn = 'trainscg'; % Scaled conjugate gradient

% Choose a Performance Function

nn.performFcn = 'crossentropy'; % Cross-entropy

% Choose Plot Functions

nn.plotFcns = {'plotperform','plottrainstate','ploterrhist','plotregression', 'plotfit'};

% Train the Network

[nn,tr] = train(nn,n\_in,t);

% Test the Network

y = nn(n\_in);

e = gsubtract(t,y);

tind = vec2ind(t);

yind = vec2ind(y);

percentErrors = sum(tind ~= yind)/numel(tind);

performance = perform(nn,t,y);

% Recalculate Training, Validation and Test Performance

trainTargets = t .\* tr.trainMask{1};

valTargets = t .\* tr.valMask{1};

testTargets = t .\* tr.testMask{1};

trainPerformance = perform(nn,trainTargets,y);

valPerformance = perform(nn,valTargets,y);

testPerformance = perform(nn,testTargets,y);

[c,cm,ind,per] = confusion(t,y);

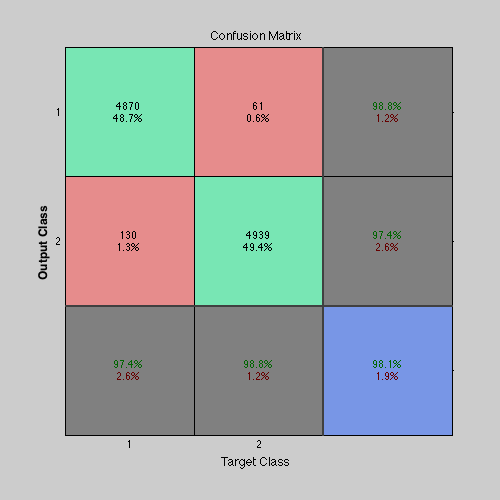
accuracy(1,i)=((1-c)\*100);

% Plots

figure, plotconfusion(t,y)

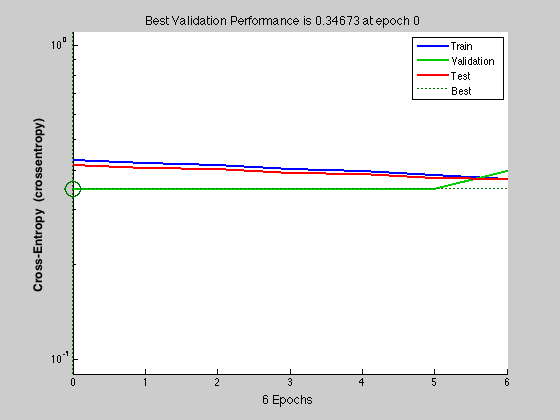
end

Confusion plot:

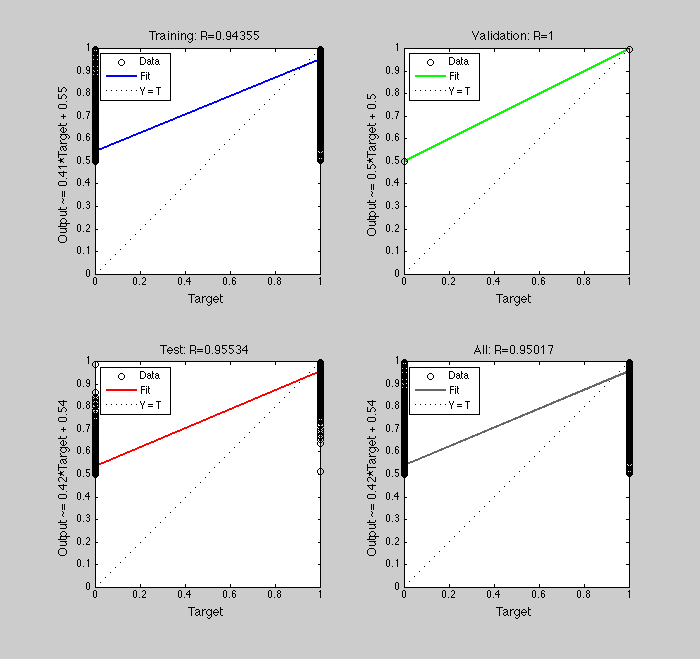


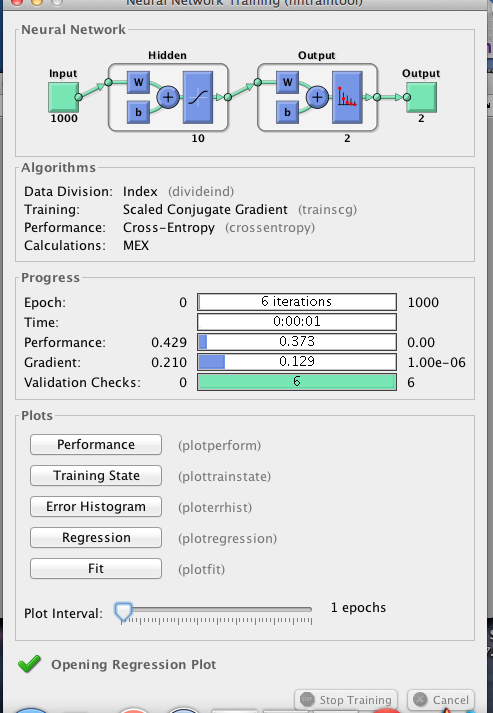
Accuracy = 98.0900%

Performance plot: Epoch 6



Regression plot:





2.Training the data set with Neural Network for the rows [1:500,1001:1500,2001:2500,3001:3500,4001:4500,5001:5500,6001:6500,7001:7500,8001:8500,9001:9500] as class 1 and the remaining rows as class 2 of the given data. Perceptron training and testing.

Code Snippet:

data=load('data.tsv');

n\_in = data';

t = label';

accuracy=zeros(1,10);

index=zeros(10000,10);

label=zeros(10000,1);

label=zeros(10000,2);

label( [1:500,1001:1500,2001:2500,3001:3500,4001:4500,5001:5500,6001:6500,7001:7500,8001:8500,9001:9500],1)=1;

label( [501:1000,1501:2000,2501:3000,3501:4000,4501:5000,5501:6000,6501:7000,7501:8000,8501:9000,9501:10000],2)=1;

cv=cvpartition(10000,'kfold',10);

for i=1:1:10

index(:,i)=test(cv,i);

end

trainid=zeros(10,9000);

testid=zeros(10,1000);

for n=1:1:10

a=1;

b=1;

for j=1:1:10000

if(index(j,n)==0)

trainid(n,a)=j;

a=a+1;

else

testid(n,b)=j;

b=b+1;

end

end

end

% Create a Pattern Recognition Network

hiddensize = 10;

nn = patternnet(hiddensize);

% Choose Input and Output Pre/Post-Processing Functions

nn.input.processFcns = {'removeconstantrows','mapminmax'};

nn.output.processFcns = {'removeconstantrows','mapminmax'};

for i=1:1:10

% Setup Division of Data for Training, Validation, Testing

nn.divideFcn = 'divideind'; % Divide data by K-Fold cross validation

nn.divideMode = 'sample'; % Divide up every sample

nn.divideParam.trainInd=trainid(i,1:8000);

nn.divideParam.valInd=trainid(i,8001:9000);

nn.divideParam.testInd=testid(i,:);

% For a list of all training functions type: help nntrain

nn.trainFcn = 'trainscg'; % Scaled conjugate gradient

% Choose a Performance Function

nn.performFcn = 'crossentropy'; % Cross-entropy

% Choose Plot Functions

nn.plotFcns = {'plotperform','plottrainstate','ploterrhist','plotregression', 'plotfit'};

% Train the Network

[nn,tr] = train(nn,n\_in,t);

% Test the Network

y = nn(n\_in);

e = gsubtract(t,y);

tind = vec2ind(t);

yind = vec2ind(y);

percentErrors = sum(tind ~= yind)/numel(tind);

performance = perform(nn,t,y);

% Recalculate Training, Validation and Test Performance

trainTargets = t .\* tr.trainMask{1};

valTargets = t .\* tr.valMask{1};

testTargets = t .\* tr.testMask{1};

trainPerformance = perform(nn,trainTargets,y);

valPerformance = perform(nn,valTargets,y);

testPerformance = perform(nn,testTargets,y);

[c,cm,ind,per] = confusion(t,y);

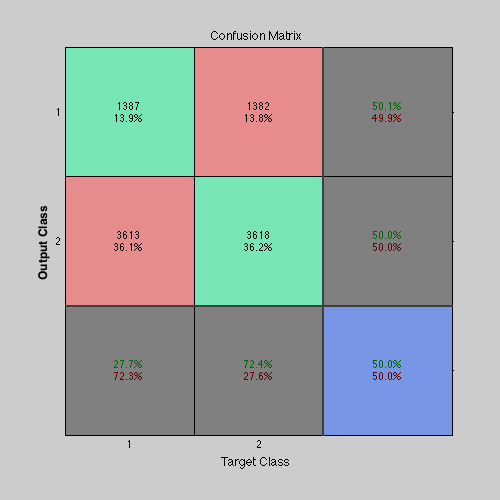
accuracy(1,i)=((1-c)\*100);

% Plots

figure, plotconfusion(t,y)

end

Confusion Plot:



Accuracy = 50.3200%

Performance plot:

