**Computer Engineering Department**

**Bilkent University**

**CS551 21701423**

**Fall 2017 Sevil Çalışkan**

**Homework 2 26/11/2017**

**Question 1:**

Initializations are done by clustering the data points to the number of mixtures to get a mean value to start. Clustering is done by k-means algorithm of matlab. Mean values are calculated as their maximum likelihood estimates as a start. Covariance matrices calculated as if they should be same spherical covariance matrices for each model as a start by calculating the variances for each feature then averaging them. Mixture weights taken as equal for each component at the beginning. Stopping condition is chosen as the loglikelihood change between iterations being less than 10, since after that value, means are not changing significantly.

Means, covariance matrices and weights of the mixtures are calculated as in the lecture notes, maximizing the loglikelihood function.

For Class – 1, assuming same spherical covariance matrix for each component of the mixture, I have tried different number of mixtures. For 2, 3, 4, 5, 6 and 7 components, mean vectors were calculated as below:

Mean vectors of 2 components mixture =

18.0187 34.1944

27.2218 25.2769

Mean vectors of 3 components mixture =

17.1238 34.7906

27.7126 23.6877

24.4113 30.2596

Mean vectors of 4 components mixture =

15.9831 36.1863

26.8229 21.6739

20.8952 31.5735

28.6235 27.8784

Mean vectors of 5 components mixture =

18.8556 32.2168

24.0356 30.7792

15.5332 37.0410

26.1258 21.0293

29.5496 26.6496

Mean vectors of 6 components mixture =

17.8312 32.9543

25.7426 20.8032

20.8985 31.5553

26.9583 29.6223

15.3539 37.3870

29.8410 25.2360

Mean vectors of 7 components mixture =

18.9091 32.0126

25.2025 20.6000

29.1372 27.8672

16.2161 35.4559

29.4187 23.0914

23.5711 30.9761

15.2247 37.8100

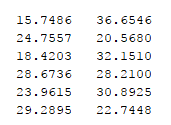
As can be seen, mean vectors of 7 components mixture are getting closer. Also looking the shape of the data, 6 seems to be a good estimation. So, I decided to use a mixture with 6 components for the first class. Number of mixtures for Class – 2 and Class – 3 is decided in the same manner. Parameters calculated by EM algorithm are as below:

Class – 1

A Gaussian mixture with 6 components is decided to be fit.

Same spherical covariance matrices:

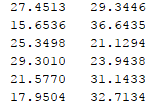
Means =



Loglikelihood = - 2759.6

Different diagonal covariance matrix for each component:

Means =

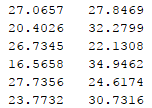




Loglikelihood = -2840.9

Different arbitrary covariance matrix for each component:

Means =





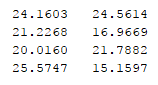
Loglikelihood = -2786.1

Class – 2

A Gaussian mixture with 4 components is decided to be fit.

Same spherical covariance matrices:

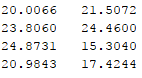
Means =



Loglikelihood = -2553.9

Different diagonal covariance matrix for each component:

Means =

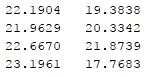




Loglikelihood = -2629.6

Different arbitrary covariance matrix for each component.

Means =





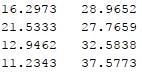
Loglikelihood = -2620.0

Class – 3

A Gaussian mixture with 4 components is decided to be fit.

Same spherical covariance matrices:

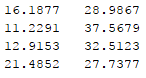
Means =



Loglikelihood = -2553.9

Different diagonal covariance matrix for each component:

Means =

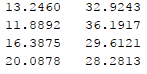




Loglikelihood = -2755.6

Different arbitrary covariance matrix for each component:

Means =





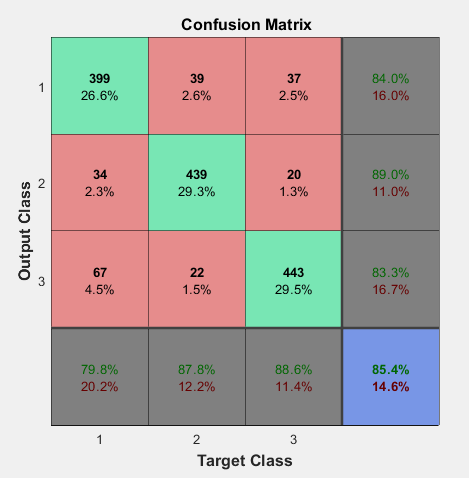
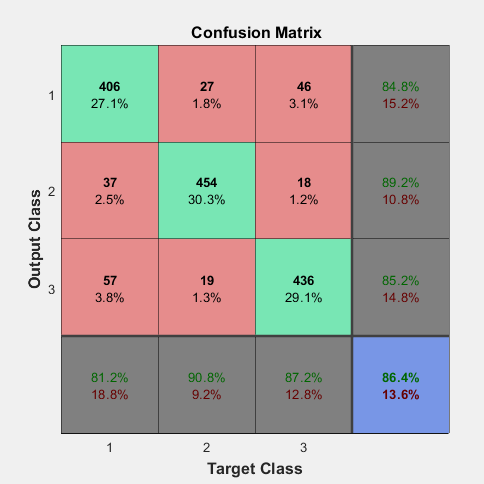
Loglikelihood = -2581.4

Now that we have the density function parameters for each class, we can test the classifiers. For classifying new coming data, we know that we should classify it as the same class that it is more likely to be a member of, i.e. it should be labelled as the same class with the highest probability of belonging to that class. Probability of belonging to one of the three classes is:

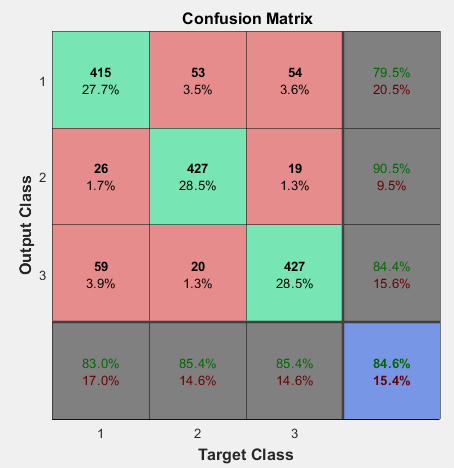
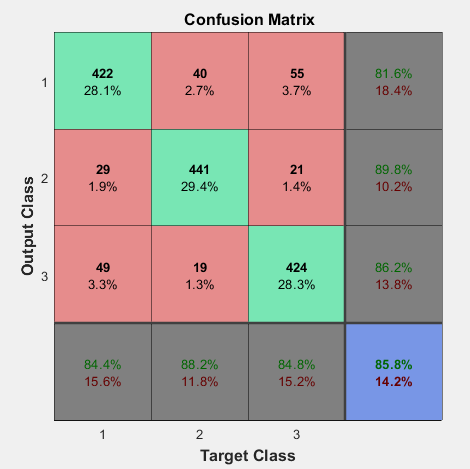
So, we know that denominator will be same for the each class-conditional probability calculation. As a result, we can use the class-conditional density function alone to compare the probabilities. It is done by so in this homework.

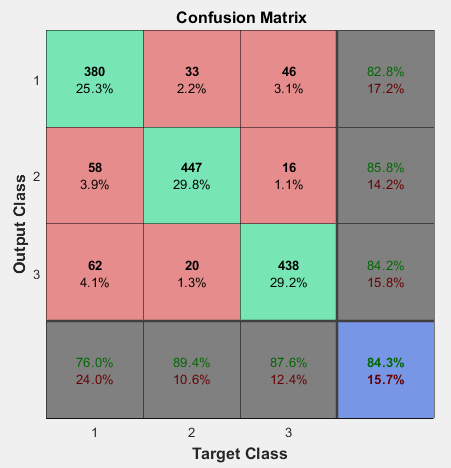
New data classified as the class whose class-conditional density gives the higher result for the new data. Then confusion matrices has been formed by the help of matlab function plotconfusion. Resulted confusion matrices are as below.

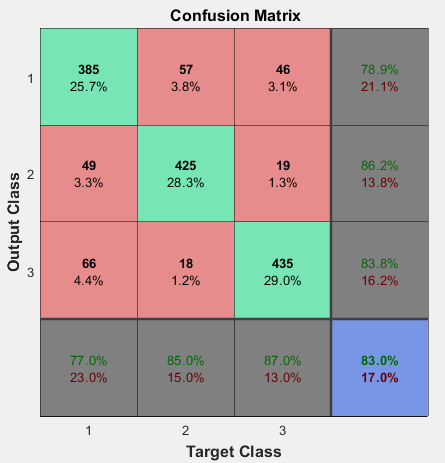
Confusion matrix of train and test sets respectively, for classifier with same spherical covariance matrices:



Confusion matrix of train and test sets respectively, for classifier with different diagonal covariance matrices:



Confusion matrix of train and test sets respectively, for classifier with different covariance matrices:



As can be seen, train set results are better than test set results as expected since the densities of the classes are modeled with the train set. Classifier with same spherical covariance matrices is giving the best results in terms of correctly classifying train and test data, then classifier with different diagonal covariance matrices and the worst results belong to the classifier with arbitrary covariance matrices. Actually, I would expect the classifier with arbitrary covariance matrices to give the best results since it could adapt itself to the different shapes of data where some are denser with less variability and some are separate with higher variability. However, when we look at the data, it looks like the variabilities of the different datasets are not really changing much and since datasets are overlapping, spherical covariance matrices can give the better results in terms of modelling variability. Also, it would be a safer choice if we do not want to over fit the model to the training data.

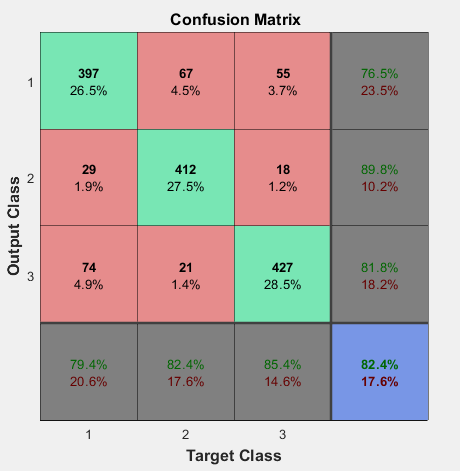
**Question 2:**

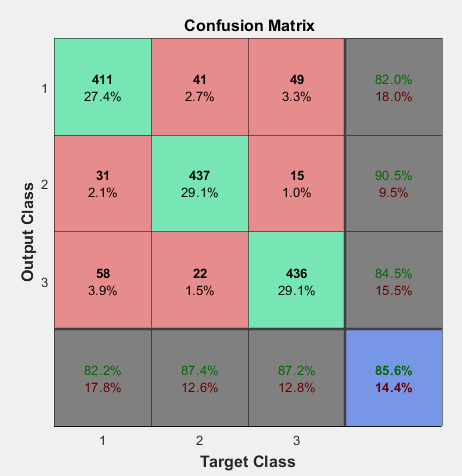
In this part, we will use histograms to estimate the densities of the different class data. Data has two features so bin size will be vectors and the volume will be the area of each bin. Density will be calculated as below:

where n is the total number of samples, k is the number of samples in the cell that includes x, and V is the volume of that cell. To separate two dimensional data into bins, I have used matlab function histcounts2. Its input are feature vectors and bin numbers vector (one for each feature). Then it returns edges for each feature and a matrix that gives the number of sample inside the bin N(i,j). Then the job is to find the bins that new coming data is in between, then calculate the density as below. Different confusion matrices for different bin sizes are as below for train and test data sets.

Nbin = [10,10]

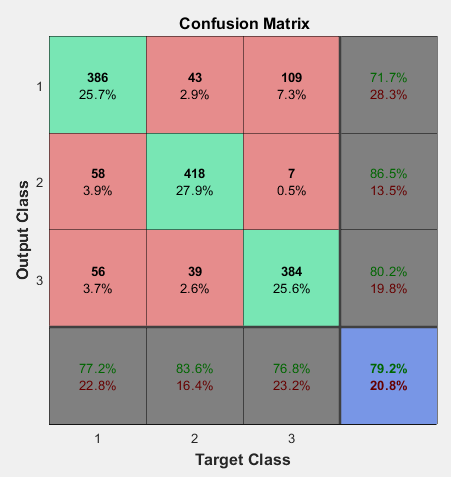
Train: Test:

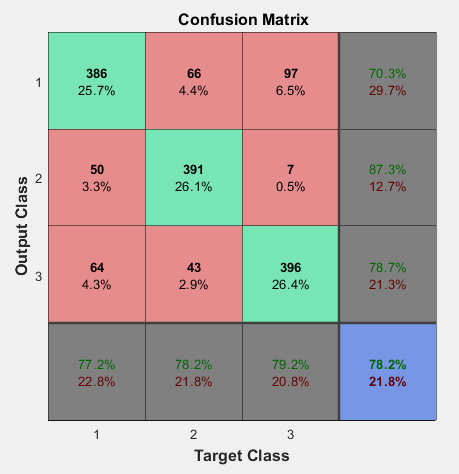




Nbin = [5,5]

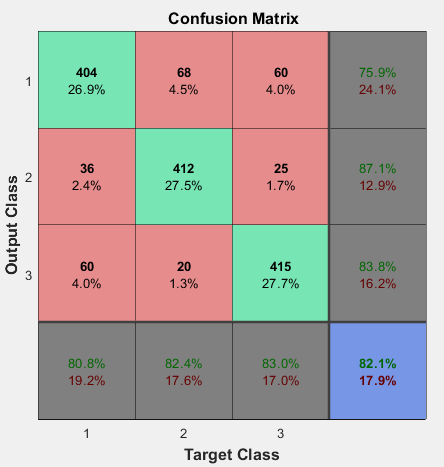
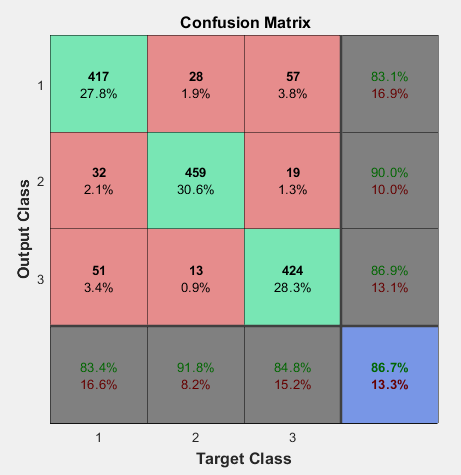
Train: Test:





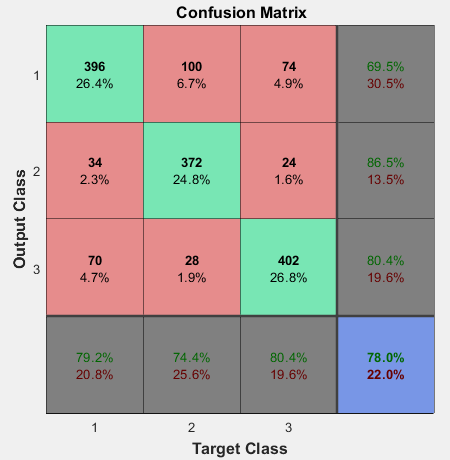
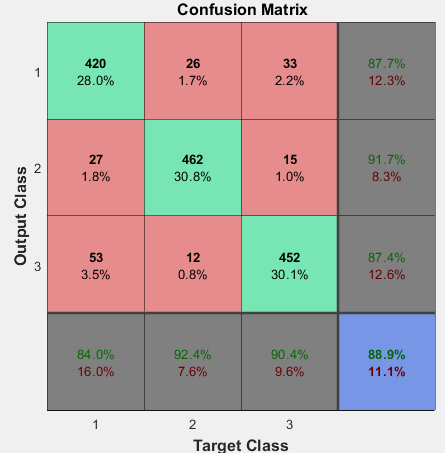
Nbin = [15,15]

Train: Test:



Nbin = [20,20]

Train: Test:



As can be seen from the confusion matrices, when we choose number of bins as 5, results are worse than the other choices. This is expected since large bin sizes are not good enough to model the density, i.e. a small number for the number of bins is not sufficient to model the shape of the density. Also when number of bins chosen as 20, success for train data is better than the others while it is not better for test data. This indicates that small bin sizes over fits the train data and model does not give same results on test data. This is also expected since small bin size or larger number of bins means dividing data into more cells and memorizing accordingly with those cells. Number of bins 10 and 15 seems to give similar results while 10 is slightly better.

Appendix:

Question – 1:

function [prob] = Gauss(data,mu,sigma)

[r,~]= size(sigma);

prob = 1/((2\*pi)^(r/2)\* det(sigma)^(1/2))\*exp((-1/2)\*(data - mu)\*sigma^(-1)\*transpose(data - mu)) ;

end

function [condprob] = CondProbj(alpha,data,mu,sigma,c,j)

prob = 0;

for i=1:c

prob = prob + alpha(1,i)\* Gauss( data , mu(i,:) , sigma(:,:,i));

end

condprob = alpha(1,j) \* Gauss( data , mu(j,:) , sigma(:,:,j))/prob ;

end

function [alpha, means, sigmasq] = Starting(data, c)

[n,~]=size(data);

means = zeros(c,2);

count = zeros(c,1);

clust=kmeans(data,c);

for i=1:n

for k=1:c

if clust(i,1)==k

means(k,:)= means(k,:)+data(i,:);

count(k,1)= count(k,1)+1;

end

end

end

means = means./count;

sigmasq = zeros(2,2,c);

sigma=0;

for i=1:n %same sphrecal covarience starting value

sigma = sigma + (data( n, 1) - means(1,1))^2 + (data( n, 2) - means(1,2))^2;

end

sigma = sigma/(2\*n);

for i=1:c

sigmasq(:,:,i) = sigma\*eye(2);

end

alpha = zeros(1,c);

for i=1:c % starting alpha value

alpha (1,i) = 1 / c;

end

end

function [alp,means,covarience] = Parametersj(alpha,data,mu,sigma,c,j,s)

[n,d]=size(data);

nominatorm = zeros(1,2);

denominatorm = 0;

denominators = 0;

nominators = 0;

covarience = 0;

alp = 0;

if s==1

for k=1:n

prob = CondProbj(alpha,data(k,:),mu,sigma,c,j);

nominatorm = nominatorm +prob \*data(k,:);

denominatorm = denominatorm + prob;

alp = alp + prob;

for i=1:c

prob = CondProbj(alpha,data(k,:),mu,sigma,c,i)\*norm(data(k,:)- mu(i,:))^2;

covarience = covarience + prob;

end

end

alp = alp/n;

means = nominatorm / denominatorm;

covarience = eye(d)\*covarience / (2\*n);

elseif s==2

nominators = zeros(2,2);

for k=1:n

prob = CondProbj(alpha,data(k,:),mu,sigma,c,j);

nominatorm = nominatorm +prob \*data(k,:);

denominatorm = denominatorm + prob;

alp = alp + prob;

for i=1:2

nominators(i,i) = nominators(i,i) + CondProbj(alpha,data(k,:),mu,sigma,c,j)\*norm(data(k,i)- mu(j,i))^2;

denominators = denominators + CondProbj(alpha,data(k,:),mu,sigma,c,j);

end

end

covarience = nominators/(denominators);

alp = alp/n;

means = nominatorm / denominatorm;

elseif s==3

for k=1:n

prob = CondProbj(alpha,data(k,:),mu,sigma,c,j);

nominatorm = nominatorm +prob \*data(k,:);

denominatorm = denominatorm + prob;

alp = alp + prob;

nominators = nominators + CondProbj(alpha,data(k,:),mu,sigma,c,j)\*transpose(data(k,:)- mu(j,:))\*(data(k,:)- mu(j,:));

denominators = denominators + CondProbj(alpha,data(k,:),mu,sigma,c,j);

end

covarience = nominators/(denominators);

alp = alp/n;

means = nominatorm / denominatorm;

end

end

function [density] = Density(alpha,data,mu,sigma,c)

density =0;

for i = 1:c

density = density + alpha(1,i)\*Gauss(data, mu(i,:),sigma(:,:,i));

end

end

% Main code

%Read data

Data = xlsread('hw2data');

%separate different class data

[r ,~]= size(Data);

firstc = zeros(0,3);

secondc = zeros(0,3);

thirdc = zeros(0,3);

for n=1:r

if Data(n,3) == 1

firstc = [firstc; Data(n,:)];

elseif Data(n,3) == 2

secondc = [secondc; Data(n,:)];

else

thirdc = [thirdc; Data(n,:)];

end

end

%For shuffling data

[n ,~]= size(secondc);

firstc = firstc(randperm(n),:);

secondc = secondc(randperm(n),:);

thirdc = thirdc(randperm(n),:);

%separate different class data to test and train sets

n = round(n/2);

firstctrain = firstc(1:n, 1:2);

firstctest = firstc((n+1):2\*n, 1:2);

secondctrain = secondc(1:n, 1:2);

secondctest = secondc((n+1):2\*n, 1:2);

thirdctrain = thirdc(1:n, 1:2);

thirdctest = thirdc((n+1):2\*n, 1:2);

clear firstc secondc thirdc Data r;

for m=1:3

[alpha1,mu1,sigma1,loglikelihood1] = Em(firstctrain,6,m);

[alpha2,mu2,sigma2,loglikelihood2] = Em(secondctrain,4,m);

[alpha3,mu3,sigma3,loglikelihood3] = Em(thirdctrain,4,m);

Density1 = @(x) Density(alpha1, x, mu1, sigma1, 6);

Density2 = @(x) Density(alpha2, x, mu2, sigma2, 4);

Density3 = @(x) Density(alpha3, x, mu3, sigma3, 4);

train = [firstctrain;secondctrain;thirdctrain];

test = [firstctest;secondctest;thirdctest];

targetsvector = [ones(1,500), 2\*ones(1,500), 3\*ones(1,500)];

outputsvector = zeros(1,1500);

for i = 1:1500

dens1 = Density1(train(i,:));

dens2 = Density2(train(i,:));

dens3 = Density3(train(i,:));

if dens1>=dens2 && dens1>=dens3

outputsvector(1,i) = 1;

else

if dens2>dens3

outputsvector(1,i)=2;

else

outputsvector(1,i)=3;

end

end

end

% Convert this data to a [numClasses x 1500] matrix

targets = zeros(3,1500);

outputs = zeros(3,1500);

targetsIdx = sub2ind(size(targets), targetsvector, 1:1500);

outputsIdx = sub2ind(size(outputs), outputsvector, 1:1500);

targets(targetsIdx) = 1;

outputs(outputsIdx) = 1;

% Plot the confusion matrix for a 3-class problem

figure

plotconfusion(targets,outputs)

for i = 1:1500

dens1 = Density1(test(i,:));

dens2 = Density2(test(i,:));

dens3 = Density3(test(i,:));

if (dens1>=dens2) && (dens1>=dens3)

outputsvector(1,i) = 1;

else

if dens2>dens3

outputsvector(1,i)=2;

else

outputsvector(1,i)=3;

end

end

end

% Convert this data to a [numClasses x 1500] matrix

targets = zeros(3,1500);

outputs = zeros(3,1500);

targetsIdx = sub2ind(size(targets), targetsvector, 1:1500);

outputsIdx = sub2ind(size(outputs), outputsvector, 1:1500);

targets(targetsIdx) = 1;

outputs(outputsIdx) = 1;

% Plot the confusion matrix for a 3-class problem

figure

plotconfusion(targets,outputs)

end

Question – 2:

function [density] = Hist(data,traindata,nbins)

[n,~] = size(traindata);

x = traindata(:,1);

y = traindata(:,2);

[N,Xedges,Yedges] = histcounts2(x,y,nbins);

[~,r1] = size(Xedges);

[~,r2] = size(Yedges);

b1=0;

b2=0;

v1=1;

v2=1;

for i=1:(r1-1)

if Xedges(i) <= data(1,1) && data(1,1) < Xedges(i+1)

b1 = i;

v1 = Xedges(i+1)- Xedges(i);

end

end

for j=1:(r2-1)

if Yedges(j) <= data(1,2) && data(1,2) < Yedges(j+1)

b2 = j;

v1 = Yedges(j+1)-Yedges(j);

end

end

if b1>0 && b2>0

density = N(b1,b2)/(n\*v1\*v2);

else

density = 0;

end

end

% Main code

train = [firstctrain;secondctrain;thirdctrain];

test = [firstctest;secondctest;thirdctest];

targetsvector = [ones(1,500), 2\*ones(1,500), 3\*ones(1,500)];

outputsvector = zeros(1,1500);

nbins = [10,10];

for i = 1:1500

dens1 = Hist(train(i,:),firstctrain,nbins);

dens2 = Hist(train(i,:),secondctrain,nbins);

dens3 = Hist(train(i,:),thirdctrain,nbins);

if dens1>=dens2 && dens1>=dens3

outputsvector(1,i) = 1;

else

if dens2>dens3

outputsvector(1,i)=2;

else

outputsvector(1,i)=3;

end

end

end

% Convert this data to a [numClasses x 1500] matrix

targets = zeros(3,1500);

outputs = zeros(3,1500);

targetsIdx = sub2ind(size(targets), targetsvector, 1:1500);

outputsIdx = sub2ind(size(outputs), outputsvector, 1:1500);

targets(targetsIdx) = 1;

outputs(outputsIdx) = 1;

% Plot the confusion matrix for a 3-class problem

figure

plotconfusion(targets,outputs)

for i = 1:1500

dens1 = Hist(test(i,:),firstctrain,nbins);

dens2 = Hist(test(i,:),secondctrain,nbins);

dens3 = Hist(test(i,:),thirdctrain,nbins);

if dens1>=dens2 && dens1>=dens3

outputsvector(1,i) = 1;

else

if dens2>dens3

outputsvector(1,i)=2;

else

outputsvector(1,i)=3;

end

end

end

% Convert this data to a [numClasses x 1500] matrix

targets = zeros(3,1500);

outputs = zeros(3,1500);

targetsIdx = sub2ind(size(targets), targetsvector, 1:1500);

outputsIdx = sub2ind(size(outputs), outputsvector, 1:1500);

targets(targetsIdx) = 1;

outputs(outputsIdx) = 1;

% Plot the confusion matrix for a 3-class problem

figure

plotconfusion(targets,outputs)