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 =

15.7486	36.6546
24.7557	20.5680
18.4203	32.1510
28.6736	28.2100
23.9615	30.8925
29.2895	22.7448

$$\sigma^2 = 3.6936$$

$$\propto = [\ 0.2240 \quad 0.1303 \quad 0.2295 \quad 0.1479 \quad 0.1464 \quad 0.1218\ ]$$

Loglikelihood = - 2759.6

Different diagonal covariance matrix for each component:

Means =

27.4513	29.3446
15.6536	36.6435
25.3498	21.1294
29.3010	23.9438
21.5770	31.1433
17.9504	32.7134

$$\sigma_1^2 = \begin{matrix} 2.6615 & 0 \\ 0 & 2.1601 \end{matrix} \quad \sigma_2^2 = \begin{matrix} 1.2142 & 0 \\ 0 & 2.4347 \end{matrix} \quad \sigma_3^2 = \begin{matrix} 3.3358 & 0 \\ 0 & 3.3308 \end{matrix}$$

$$\sigma_4^2 = \begin{matrix} 2.2893 & 0 \\ 0 & 3.4281 \end{matrix} \quad \sigma_5^2 = \begin{matrix} 2.8165 & 0 \\ 0 & 1.0949 \end{matrix} \quad \sigma_6^2 = \begin{matrix} 1.4579 & 0 \\ 0 & 1.6108 \end{matrix}$$

$$\propto = [\ 0.1519 \quad 0.2249 \quad 0.1545 \quad 0.1484 \quad 0.1641 \quad 0.1564]$$

Loglikelihood = -2840.9

Different arbitrary covariance matrix for each component:

Means =

27.0657	27.8469
20.4026	32.2799
26.7345	22.1308
16.5658	34.9462
27.7356	24.6174
23.7732	30.7316

$$\sigma_1^2 = \begin{matrix} 11.4354 & -6.4857 \\ -6.4857 & 15.0396 \end{matrix} \quad \sigma_2^2 = \begin{matrix} 17.2703 & -6.9651 \\ -6.9651 & 5.9445 \end{matrix} \quad \sigma_3^2 = \begin{matrix} 8.7812 & 3.9498 \\ 3.9498 & 7.8339 \end{matrix}$$

$$\sigma_4^2 = \begin{matrix} 2.8967 & -2.4776 \\ -2.4776 & 10.2600 \end{matrix} \quad \sigma_5^2 = \begin{matrix} 6.6901 & 2.0524 \\ 2.0524 & 16.2957 \end{matrix} \quad \sigma_6^2 = \begin{matrix} 21.0891 & -9.4459 \\ -9.4459 & 8.3212 \end{matrix}$$

$$\propto = [\,0.0881 \quad 0.1775 \quad 0.1861 \quad 0.2988 \quad 0.1102 \quad 0.1394]$$

Loglikelihood = -2786.1

Class – 2

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

Same spherical covariance matrices:

Means =

24.1603	24.5614
21.2268	16.9669
20.0160	21.7882
25.5747	15.1597

$$\sigma^2 = 3.3548$$

$$\propto = [\,0.2509 \quad 0.2717 \quad 0.2809 \quad 0.1965]$$

Loglikelihood = -2553.9

Different diagonal covariance matrix for each component:

Means =

20.0066	21.5072
23.8060	24.4600
24.8731	15.3040
20.9843	17.4244

$$\sigma_1^2 = \begin{matrix} 1.7578 & 0 \\ 0 & 1.9549 \end{matrix} \qquad \sigma_2^2 = \begin{matrix} 2.7869 & 0 \\ 0 & 1.7583 \end{matrix}$$

$$\sigma_3^2 = \begin{matrix} 2.5650 & 0 \\ 0 & 1.6579 \end{matrix} \qquad \sigma_4^2 = \begin{matrix} 1.6713 & 0 \\ 0 & 1.7416 \end{matrix}$$

$$\propto = [0.2533 \quad 0.2755 \quad 0.2477 \quad 0.2235]$$

Loglikelihood = -2629.6

Different arbitrary covariance matrix for each component.

Means =

$$\begin{matrix} 22.1904 & 19.3838 \\ 21.9629 & 20.3342 \\ 22.6670 & 21.8739 \\ 23.1961 & 17.7683 \end{matrix}$$

$$\sigma_1^2 = \begin{matrix} 7.0477 & -1.2440 \\ -1.2440 & 14.8529 \end{matrix} \qquad \sigma_2^2 = \begin{matrix} 7.3909 & -0.9325 \\ -0.9325 & 14.2864 \end{matrix}$$

$$\sigma_3^2 = \begin{matrix} 8.1466 & 4.1037 \\ 4.1037 & 14.3743 \end{matrix} \qquad \sigma_4^2 = \begin{matrix} 9.5373 & -6.1425 \\ -6.1425 & 14.0738 \end{matrix}$$

$$\propto = [\,0.2642 \quad 0.2627 \quad 0.2441 \quad 0.2289]$$

Loglikelihood = -2620.0

Class – 3

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

Same spherical covariance matrices:

Means =

$$\begin{matrix} 16.2973 & 28.9652 \\ 21.5333 & 27.7659 \\ 12.9462 & 32.5838 \\ 11.2343 & 37.5773 \end{matrix}$$

$$\sigma^2 = 3.3548$$

$$\propto = [\,0.2509 \quad 0.2717 \quad 0.2809 \quad 0.1965]$$

Loglikelihood = -2553.9

Different diagonal covariance matrix for each component:

Means =

```
16.1877   28.9867
11.2291   37.5679
12.9153   32.5123
21.4852   27.7377
```

$\sigma_1^2 =$
```
1.2913         0
     0    1.5803
```
$\sigma_2^2 =$
```
1.6920         0
     0    1.8887
```

$\sigma_3^2 =$
```
1.3115         0
     0    1.2065
```
$\sigma_4^2 =$
```
2.0300         0
     0    1.3807
```

$$\propto = [0.2442 \quad 0.2612 \quad 0.2574 \quad 0.2372]$$

Loglikelihood = -2755.6

Different arbitrary covariance matrix for each component:

Means =

```
13.2460   32.9243
11.8892   36.1917
16.3875   29.6121
20.0878   28.2813
```

$\sigma_1^2 =$
```
 7.0392   -6.1907
-6.1907   12.5637
```
$\sigma_2^2 =$
```
 3.4422   -1.9475
-1.9475    9.8242
```

$\sigma_3^2 =$
```
12.3789   -5.8668
-5.8668    7.4357
```
$\sigma_4^2 =$
```
10.8802   -1.3662
-1.3662    2.9417
```

$$\propto = [\ 0.2615 \quad 0.2556 \quad 0.2425 \quad 0.2405]$$

Loglikelihood = -2581.4

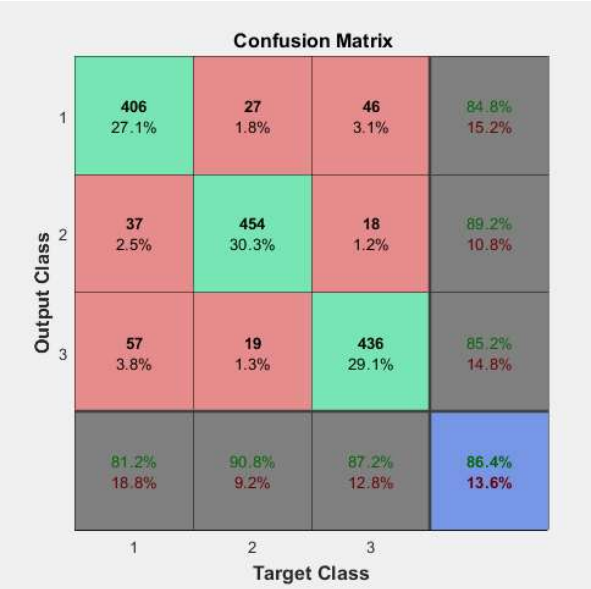
Now that we have the density function parameters for each class, we can test the classifiers. For classifying new coming data, we know that we should classify it as the same class that it is more likely to be a member of, i.e. it should be labelled as the same class with the highest probability of belonging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

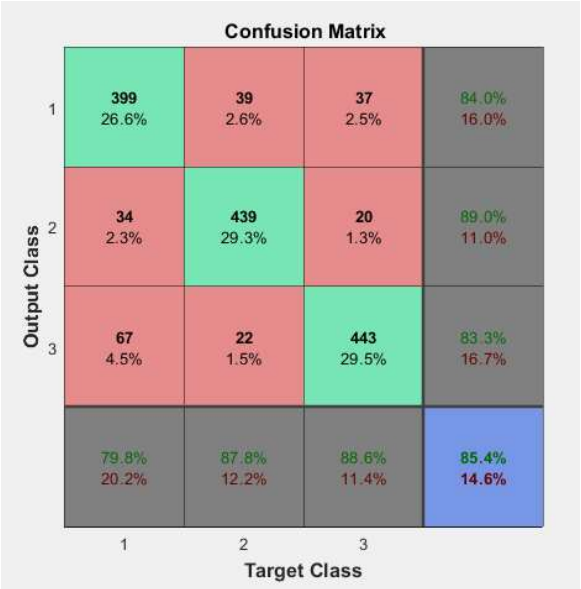
$$\frac{p_i(x|\theta_i)}{\sum_j p_j(x|\theta_j)}$$

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.

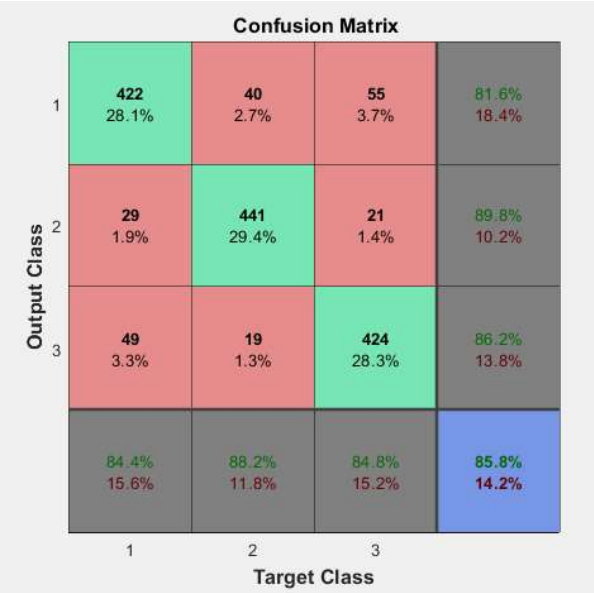
New data classified as the class whose class-conditional density gives the higher result for the new data. Then confusion matrices has been formed by the help of matlab function `plotconfusion`. Resulted confusion matrices are as below.

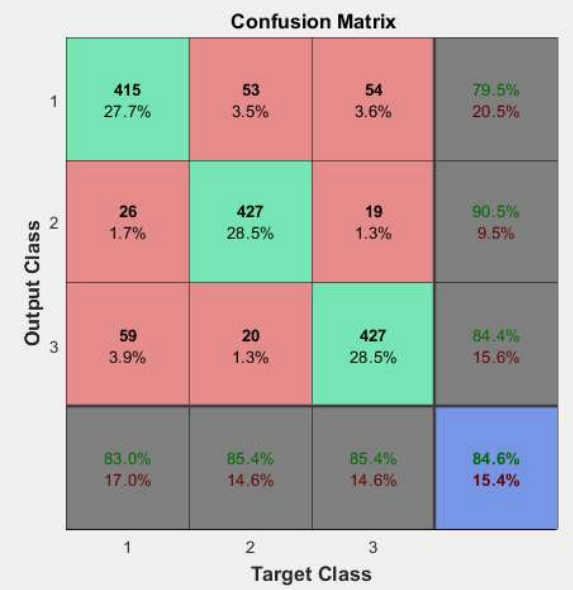
Confusion matrix of train and test sets respectively, for classifier with same spherical covariance matrices:



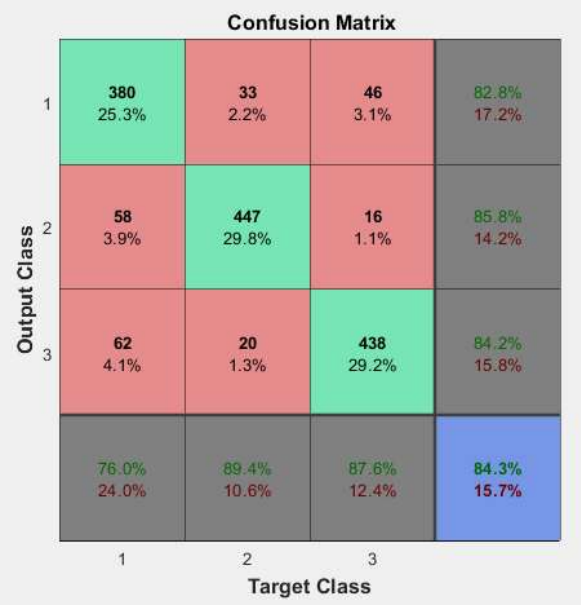


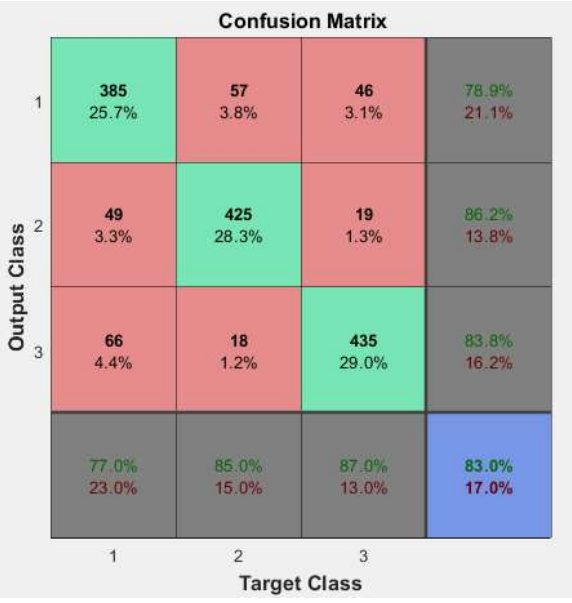
Confusion matrix of train and test sets respectively, for classifier with different diagonal covariance matrices:





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:

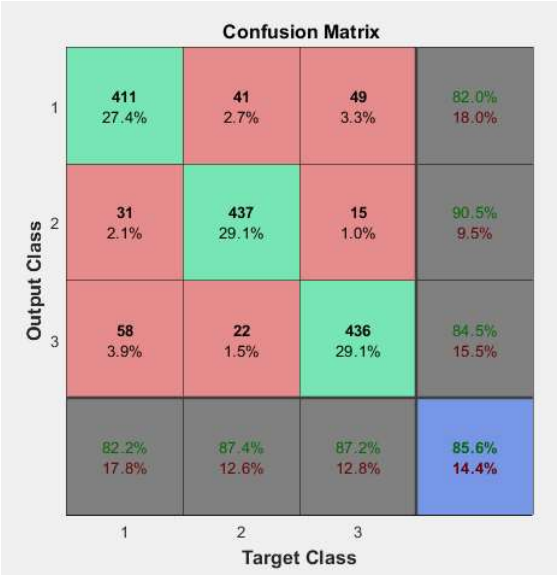
In this part, we will use histograms to estimate the densities of the different class data. Data has two features so bin size will be vectors and the volume will be the area of each bin. Density will be calculated as below:

$$p(x) = \frac{k}{n * V}$$

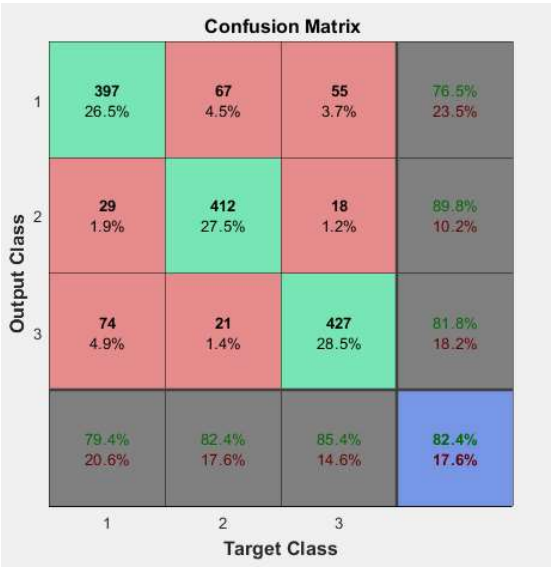
where n is the total number of samples, k is the number of samples in the cell that includes x, and V is the volume of that cell. To separate two dimensional data into bins, I have used matlab function `histcounts2`. Its input are feature vectors and bin numbers vector (one for each feature). Then it 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:

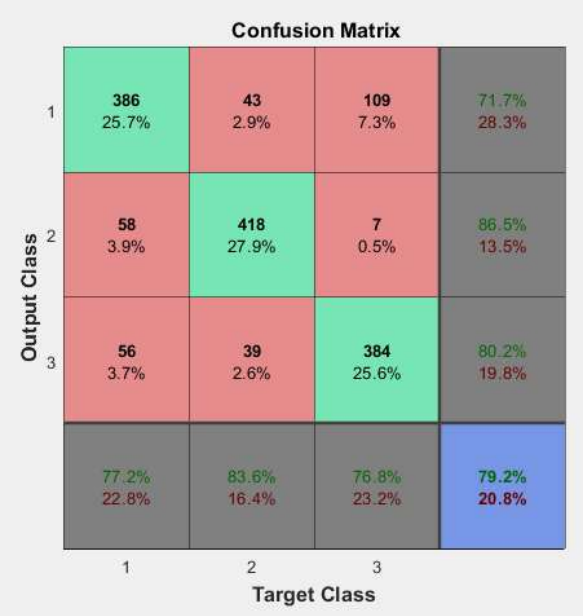


Test:

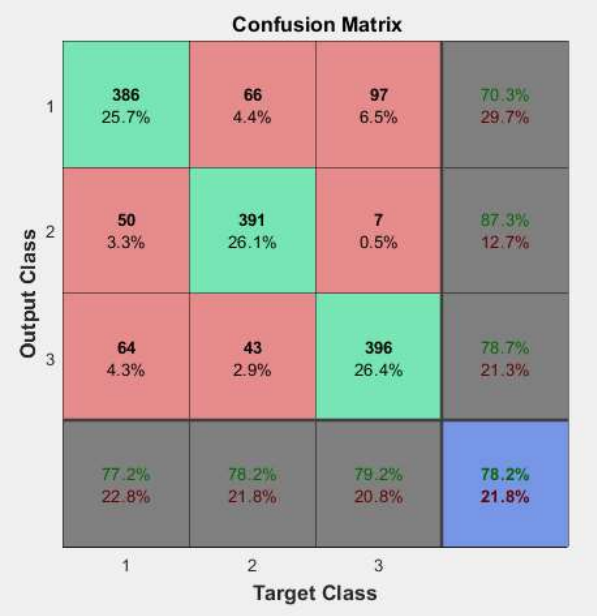


Nbin = [5,5]

Train:

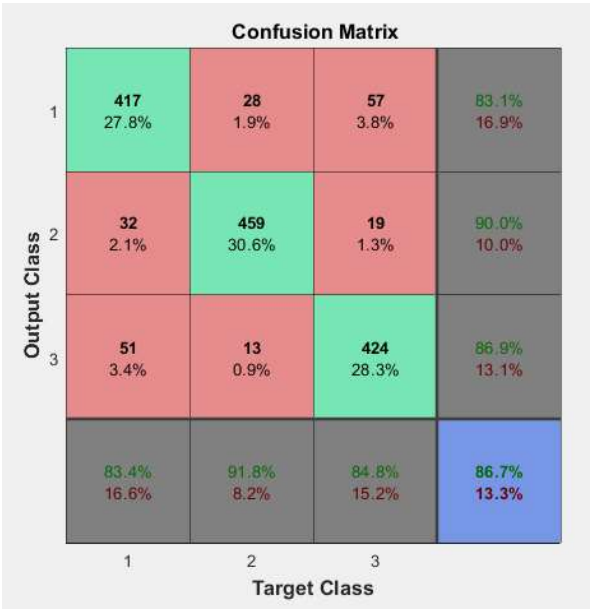


Test:

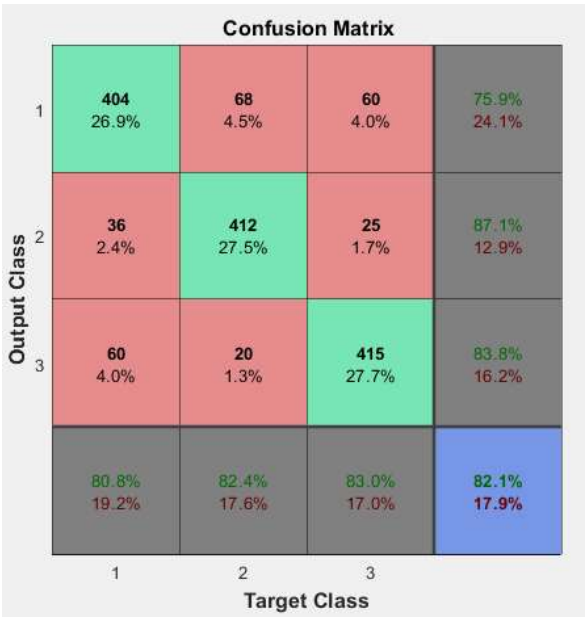


Nbin = [15,15]

Train:

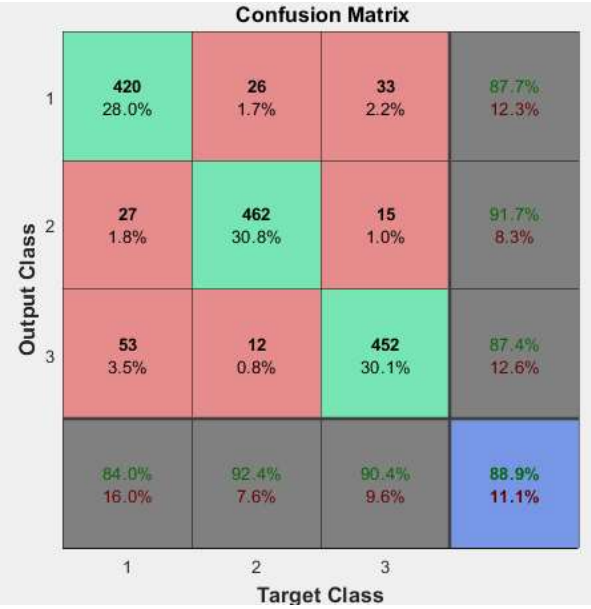


Test:

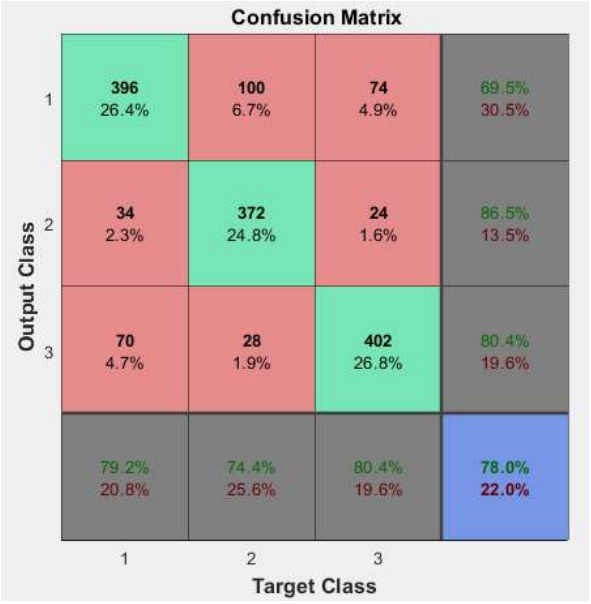


Nbin = [20,20]

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 [alpha,mu,sigma,loglikelihood] = Em(data,c,s)
[n,~]=size(data);
[alpha,mu,sigma]= Starting(data, c);
count = 0;
alp = zeros(1,c);
mean = zeros(c,2);
covarience = zeros(2,2,c);
loglikelihood = -100000;
difference = 100;
while difference>10
    for i=1:c
        [alp(1,i),mean(i,:),covarience(:,:,i)] =
Parametersj(alpha,data,mu,sigma,c,i,s);
    end
    alpha = alp;
    mu = mean;
    sigma = covarience;

    loglikelihoodprev=loglikelihood;
    loglikelihood = 0;
    for k=1:n
        loglik = 0;
        for i=1:c
            loglik = loglik + alpha(1,i)*Gauss(data(k,:),
mu(i,:),sigma(:,:,i));
        end
        loglikelihood = loglikelihood + log(loglik);
    end
    difference = loglikelihood-loglikelihoodprev;
    count = count + 1;
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)
```