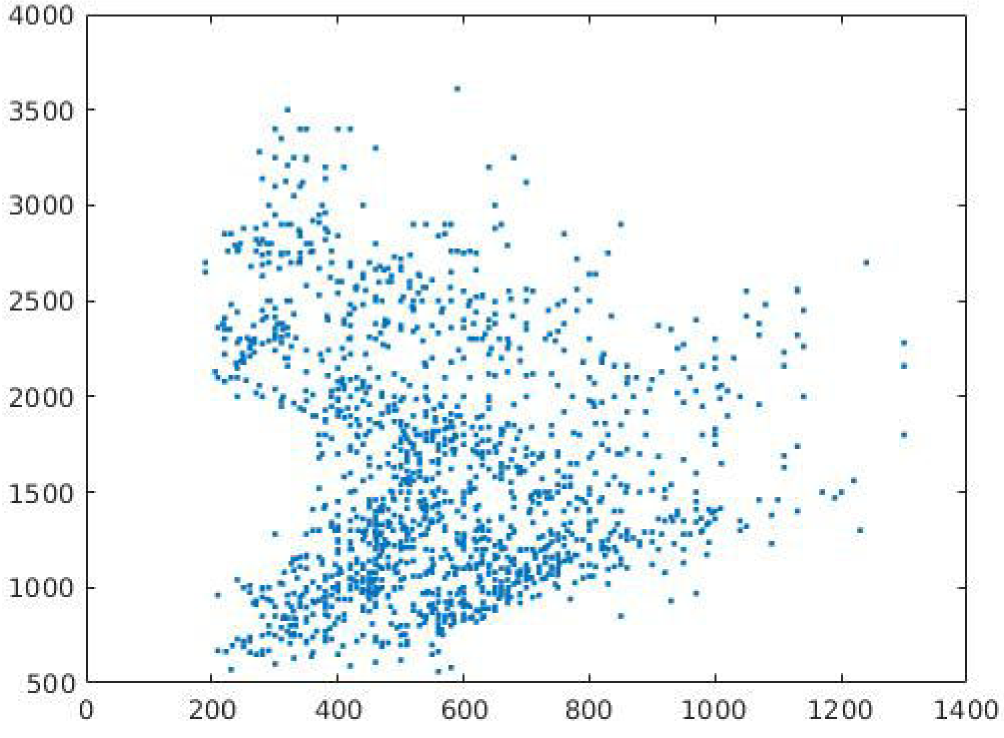
Aayush Saxena
170751708

ASSIGNMENT 2

TASK 1 :

Include in your report the corresponding lines of your code and the plot.

```
J = [f1,f2]
plot(J(:,1),J(:,2),'.')
```



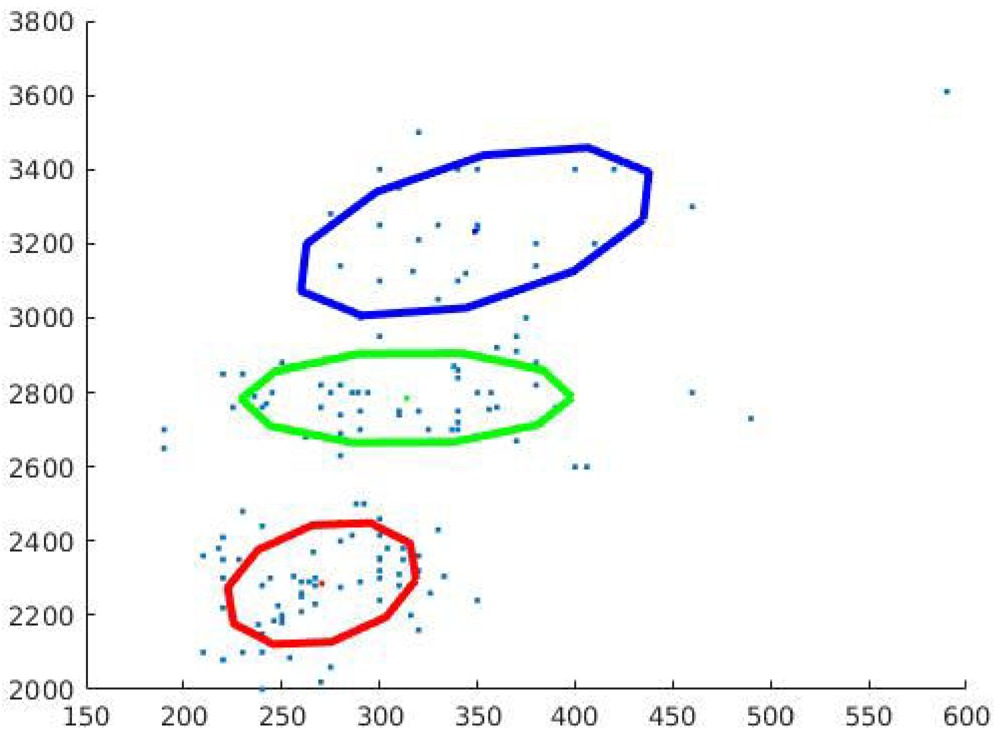
TASK 2:

Include in your report the lines of code you wrote, and results that illustrate the learnt models

For Phoneme :1

For K =3

```
load('PB_data.mat')
x=X1;
```



p (Prob)=

0.3810 0.4351 0.1839

mu (Mean)=

1.0e+03 *

0.3126 0.2704 0.3508

2.7839   2.2855   3.2263

>> s2 (CoVariance Matrix)

s2(:,:,1) =

1.0e+03 *

3.5626	**0**
0	**7.6578**

s2(:,:,2) =

1.0e+04 *

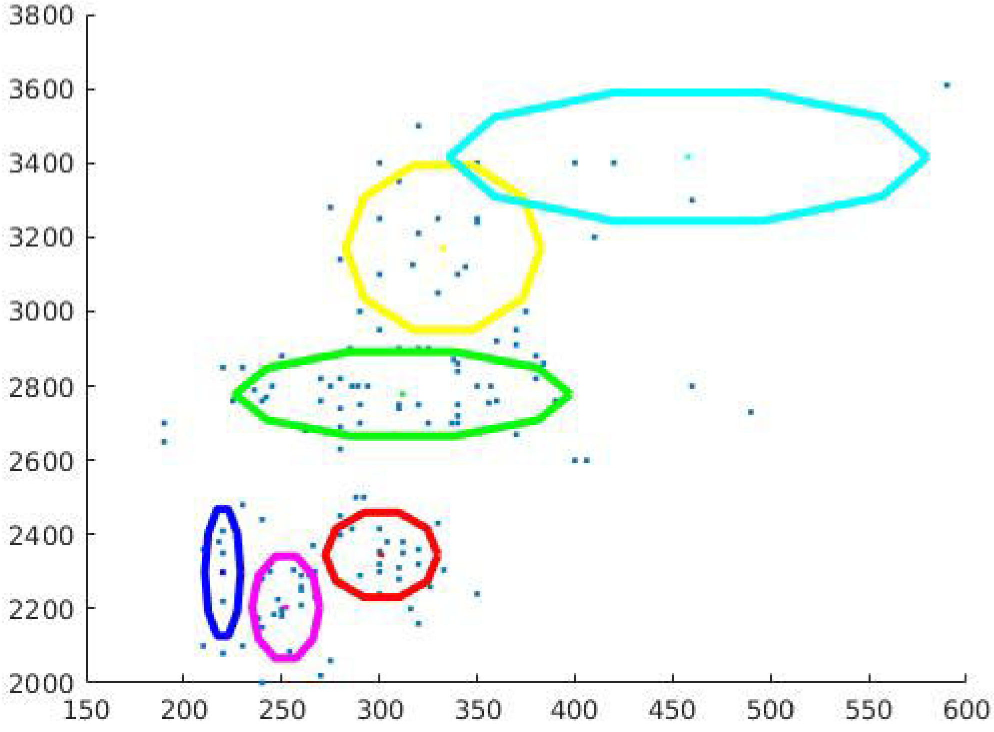
0.1214	**0**
0	**1.4278**

s2(:,:,3) =

1.0e+04 *

0.4103	**0**
0	**2.7830**

For K =6



```
p =

    0.2050    0.3676    0.0649    0.1722    0.1642    0.0261

mu =

   1.0e+03 *

    0.3011    0.3119    0.2199    0.3327    0.2521    0.4578
    2.3450    2.7784    2.2976    3.1717    2.2040    3.4164


>> s2

s2(:,:,1) =

   1.0e+03 *

    0.4176         0
         0    7.2095


s2(:,:,2) =

   1.0e+03 *

    3.6839         0
         0    7.0567


s2(:,:,3) =

   1.0e+04 *

    0.0041         0
         0    1.6137


s2(:,:,4) =

   1.0e+04 *

    0.1252         0
         0    2.7156


s2(:,:,5) =
```

1.0e+04 *

0.0150 0
0 1.0444

s2(:,:,6) =

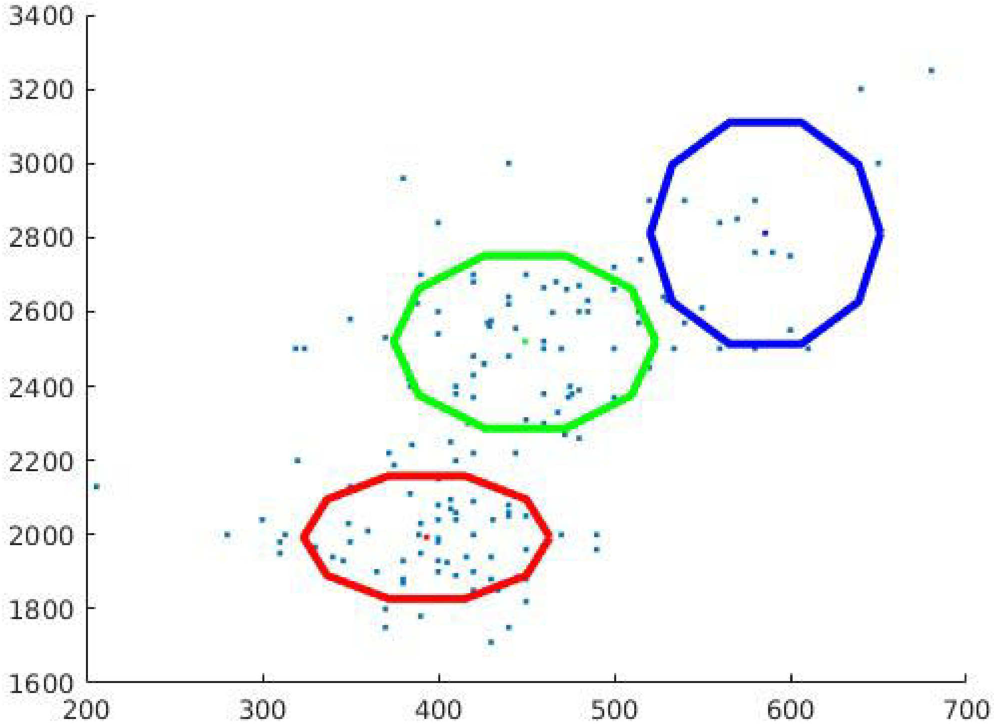
1.0e+04 *

0.7475 0
0 1.6557

For Phoneme :2

For K =3

```
load('PB12.mat')
x=X2;
```



p =

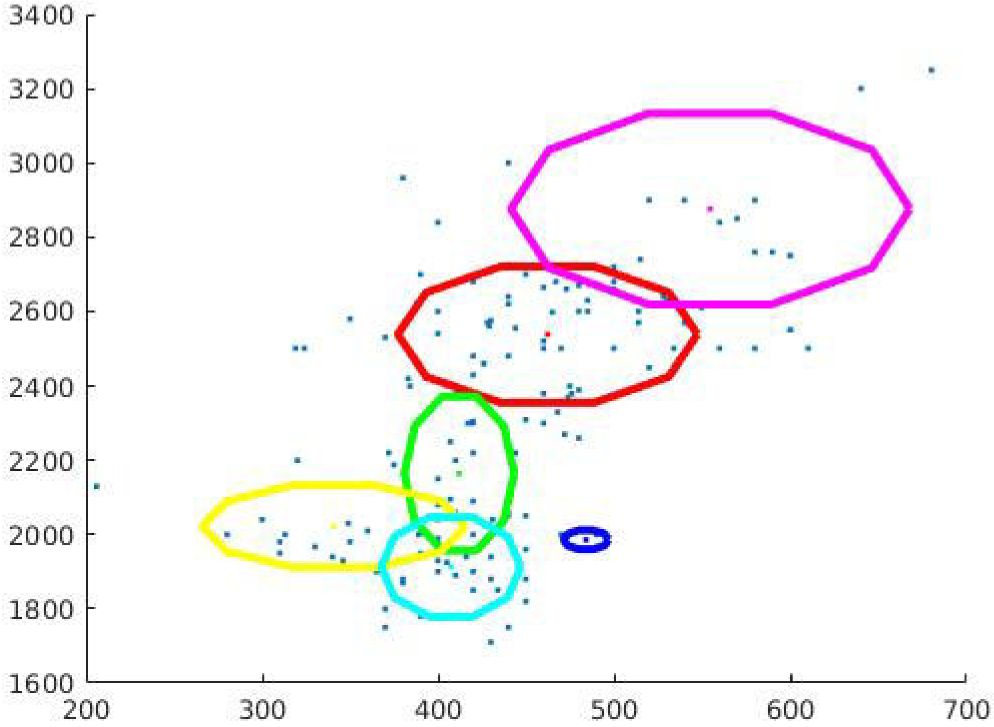
0.4345 0.4691 0.0964

mu =

1.0e+03 *

```
    0.3934    0.4493    0.5858
    1.9928    2.5189    2.8115

>> s2

s2(:,:,1) =

   1.0e+04 *

    0.2457         0
         0    1.5200


s2(:,:,2) =

   1.0e+04 *

    0.2785         0
         0    2.9797


s2(:,:,3) =

   1.0e+04 *

    0.2133         0
         0    4.9419
```

For K =6



p =

0.4064 0.1817 0.0181 0.1164 0.0953 0.1822

mu =

1.0e+03 *

0.4623	0.4121	0.4842	0.3408	0.5546	0.4074
2.5382	2.1645	1.9859	2.0223	2.8762	1.9127

>> **s2**

s2(:,:,1) =

1.0e+04 *

```
    0.3630         0
         0    1.8521


s2(:,:,2) =

   1.0e+04 *

    0.0489         0
         0    2.3847


s2(:,:,3) =

   83.0090         0
         0  364.9986


s2(:,:,4) =

   1.0e+03 *

    2.8080         0
         0    6.7625


s2(:,:,5) =

   1.0e+04 *

    0.6410         0
         0    3.6594


s2(:,:,6) =

   1.0e+03 *

    0.7865         0
         0    9.9407
```

TASK 3 :

Include in your report the lines of the code that your wrote, explanations of what the code does and comment on the differences on the classification performance.

For Task 3 , separate main file is created **mog(task3).m**

mog(task3).m

```
%Loading data (X1,X2)

load('PB12.mat')
[n D] = size(X1);

%Spliting Data into Training & Test

tr_index = floor(0.8*n);
xtr1=X1(1:tr_index,:);
xtr2=X2(1:tr_index,:);

X1_test = X1(tr_index+1:end,:);
X2_test = X2(tr_index+1:end,:);
X_test =[X1_test;X2_test];
Y1_test = ones(size(X1_test,1),1);
Y2_test = ones(size(X2_test,1),1)*2;
Y_test = [Y1_test;Y2_test];

% number of Guassians
k = 6;

% A new function Exp_Max is called and given arguments Xtr,k. The function will
return learnt parameters - mean, covariance matrix and probability of k
Gaussians.

[mu_1,s2_1,p_1] = Exp_Max(xtr1,k);
[mu_2,s2_2,p_2] = Exp_Max(xtr2,k);

% A new function predict is called and given arguments Xtest,k,mean,covariance
matrix and probability of k Gaussian. The function will return probability
vector for respective phonemes

[p_test_1] = predict(X_test,k,p_1,mu_1,s2_1);
[p_test_2] = predict(X_test,k,p_2,mu_2,s2_2);

% Classification: If the probability of Phoneme1 is greater than Phoneme2, it is
given Class 1 otherwise Class2

class =zeros(size(X_test,1),1);
for i = 1:size(X_test,1)
    if p_test_1(i) > p_test_2(i)
        class(i) =1;
    else
        class(i) =2;
    end
end
```

```
% Determining Accuracy:
counter =0.0;
for i =1:size((X_test),1)
    if class(i) == Y_test(i)
        counter = counter+1;
    end
end
Accuracy = counter/size(X_test,1)
```

Exp_Max.m

```
function [mu,s2,p]= Exp_Max(x,k)

[n D] = size(x);                    % number of observations (n) and dimension
(D)
p = ones(1,k)/k;                    % mixing proportions
mu = x(ceil(n.*rand(1,k)),:)';      % means picked rqandomly from data
s2 = zeros(D,D,k);                  % covariance matrices
niter=100;                          % number of iterations

% initialize covariances

for i=1:k
  s2(:,:,i) = cov(x)./k;      % initially set to fraction of data covariance
end

set(gcf,'Renderer','zbuffer');

clear Z;
try

% run EM for niter iterations

  for t=1:niter,
    fprintf('t=%d\r',t);

% Do the E-step:

    for i=1:k
      Z(:,i) = p(i)*det(s2(:,:,i))^(-0.5)*exp(-0.5*sum((x'-
repmat(mu(:,i),1,n))'*inv(s2(:,:,i)).*(x'-repmat(mu(:,i),1,n))',2));
    end
    Z = Z./repmat(sum(Z,2),1,k);

% Do the M-step:

    for i=1:k
      mu(:,i) = (x'*Z(:,i))./sum(Z(:,i));

 % We will fit Gaussians with diagonal covariances:

      s2(:,:,i) = diag((x'-repmat(mu(:,i),1,n)).^2*Z(:,i)./sum(Z(:,i)));
      p(i) = mean(Z(:,i));
    end

    clf
    hold on
```

```
    plot(x(:,1),x(:,2),'.');
    for i=1:k
      plot_gaussian(2*s2(:,:,i),mu(:,i),i,11);
    end
    drawnow;
  end

catch
  disp('Numerical Error in Loop - Possibly Singular Matrix');
end;
end
```

predict.m

```
% Here, the learnt parameters from Training dataset are used to predict
probability of the test data belonging to each cluster of the given class

function [p_test]= predict(x,k,p,mu,s2)

[n D] = size(x);
set(gcf,'Renderer','zbuffer');
clear Z;

%Calculate the sum of probability of the data point belonging to each cluster of
a given class

    for i=1:k
      Z(:,i) = p(i)*det(s2(:,:,i))^(-0.5)*exp(-0.5*sum((x'-
repmat(mu(:,i),1,n))'*inv(s2(:,:,i)).*(x'-repmat(mu(:,i),1,n))',2));
    end
  Z=sum(Z,2);
  p_test =Z;
end
```

Results:

For k=6, Accuracy = 0.9194

For K=3, Accuracy = 0.8871

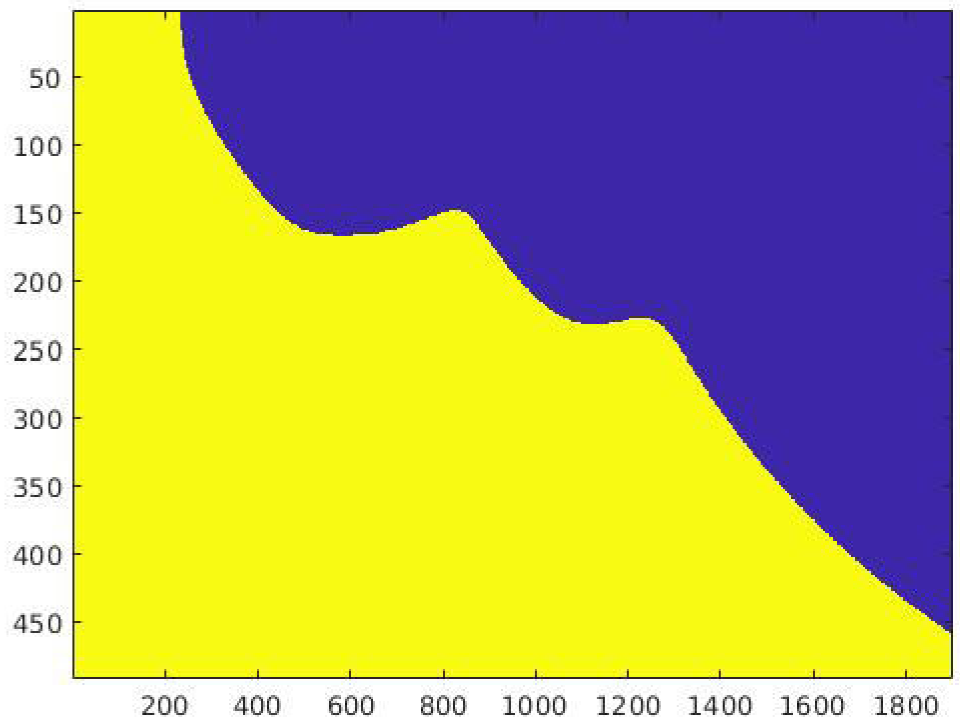
Clearly, K =6 gives higher accuracy than k=3 as it can generalize the data well.

TASK 4

Include the lines of code in your report, comment them, and the display the classification matrix.

task_4.m

```
%Load data
load('PB_data.mat')
load('PB12.mat')
k = 3;

%function Exp_Max is called and given arguments X,k. The function will return
learnt parameters - mean, covariance matrix and probability of k Gaussian.

[mu_1,s2_1,p_1] = Exp_Max(X1,k);
[mu_2,s2_2,p_2] = Exp_Max(X2,k);

%Creating the grid of points that spans the two datasets

X_min1 = min([X1(:,1);X2(:,1)]);
X_min2 = min([X1(:,2);X2(:,2)]);
X_max1 = max([X1(:,1);X2(:,1)]);
X_max2 = max([X1(:,2);X2(:,2)]);

%Initializing M
M = zeros(X_max1 -X_min1,X_max2-X_min2);

for i = X_min1:X_max1
   for j =X_min2:X_max2

 %function predict is called to compute the probability of the data point
belonging to each cluster of the given class

      [p_test_1] = predict([i,j],k,p_1,mu_1,s2_1);
      [p_test_2] = predict([i,j],k,p_2,mu_2,s2_2);

% Classification: If the probability of Phoneme1 is greater than Phoneme2, it is
given Class 1 otherwise Class2

        if p_test_1>p_test_2
            M(i-X_min1+1,j-X_min2+1) =1;
        else
            M(i-X_min1+1,j-X_min2+1) =2;
        end

   end
end

% Displaying Classification Matrix M
imagesc(M);
```



50
100
150
200
250
300
350
400
450
200
400
600
800
1000
1200
1400
1600
1800

TASK 5

Suggest ways of overcoming the singularity problem and implement them.
Include the lines of code in your report, and graphs/plots so as to support your observations.

```
% New Data:
load('PB_data.mat')
J = [f1,f2,f1+f2];
phoneme_number = 1;
x = J(phno==phoneme_number,:)
```

```
 % To fit general Gaussians:
      s2(:,:,i) = (x'-repmat(mu(:,i),1,n))*(repmat(Z(:,i),1,D).*(x'-
repmat(mu(:,i),1,n))')./sum(Z(:,i));
```

Observations:

Here, we are using full Covariance matrix.
After fitting the Mog model into new data, it gives following error:

Numerical Error in Loop - Possibly Singular Matrix
This is due to the addition of third redundant dimension which is linearly dependent on first two dimensions resulting into non singular covariance matrix. MoGs model overfits in this particular case.

Solution:

We will regularize the covariance matrix by adding a diagonal matrix which constrains the covariance always above some minimum value.

```
% New Data with 'eps':
load('PB_data.mat')
J = [f1,f2,f1+f2];
phoneme_number = 1;
eps = 0.001;
x = J(phno==phoneme_number,:)
```

```
 % To fit general Gaussians after adding the diagonal matrix:
s2(:,:,i) = (x'-repmat(mu(:,i),1,n))*(repmat(Z(:,i),1,D).*(x'-
repmat(mu(:,i),1,n))')./sum(Z(:,i)) + eps*eye(D);
```

For k=6,