MACHINE LEARNING – ASSIGNMENT 2

140361229 - TANMAIYII RAO

MoG Modelling using the EM algorithm

Task 1

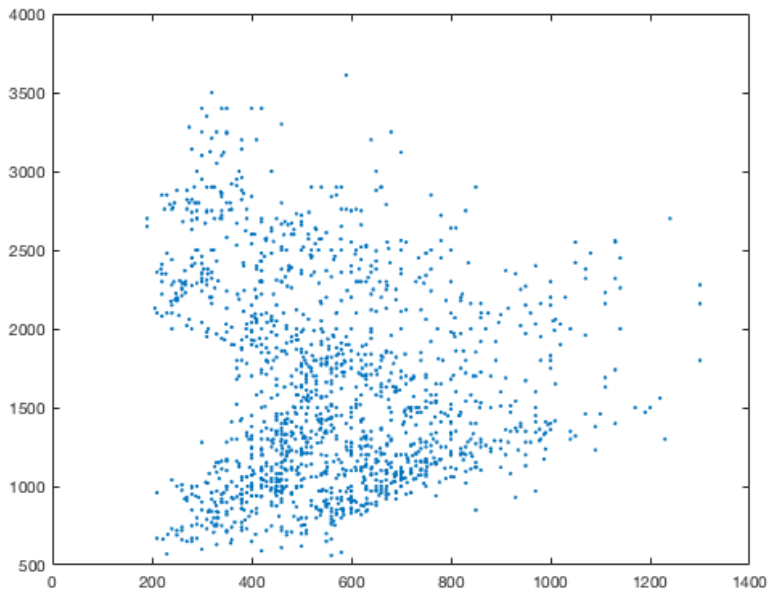
The dataset PB_data.mat was loaded onto Matlab. For which, only the data for F1 and F2 was used. The following lines of code have been given to plot F1 against F2.

First, the data was loaded by typing the following into the command window:

```
>> load('PB_data.mat')
>> J = [f1, f2]
```

Then, the following code was used to produce the scatter plot –

```
>> plot(J(:,1),J(:,2),'.')
```



Task 2

Here, it can be seen that the mog.m code is calculating the mean(mu), covariance(s2) and the prior distribution (p) of the data for phoneme1 and phoneme2 respectively. The dataset has been split into train and test data where the first 121 training examples (79.6% - apprx. 80%) have been put under the training set. Thus, it is a 80:20 split where the remaining is used for testing the models in task3.

A dataset 'x' is generated such that it only contains F1 and F2 for the first phoneme for both K =3 and K = 6 gaussians. The same steps are repeated for second phoneme. For each K value, the code is run several times and the model (comprising mu, s2 and p) are saved.

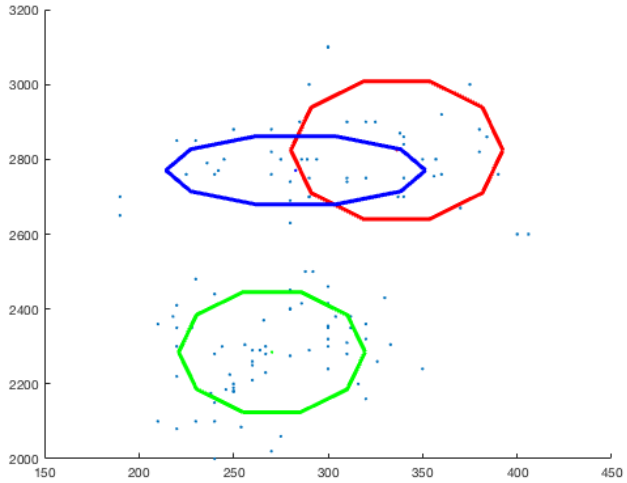
The respective graphs and the models have been illustrated below.

The following lines of code have been added to the mog.m file-
For each phoneme, the code for parameters is updated with phoneme1 and phoneme2. Also, the k value is updated for k=3 and k=6 for phoneme1 and phoneme2 respectively. The respective models for each phoneme and k value have been saved. The lines of code for saving the model is uncommented accordingly as per the k value and phoneme. The models have been trained separately for each phoneme.

```
%Task 2

% Initialise parameters
data = load('PB_data.mat');
J = [f1 f2];
%for phno =1
set1 = find(phno==1);
%for phno =2
set2 = find(phno==2);
%the training set
x = J(set2(1:121),:);

[n D] = size(x);                    % number of observations (n) and dimension (D)
k = 3;                              % number of components
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

%Saving the model for phno==1, k=3
a ={}
a.mu = mu;
a.s2 = s2;
a.p = p;
save('a.mat','a');

% Saving the model for phno==1, k=6
%c ={}
%b ={}
%b.mu = mu;
%b.s2 = s2;
%b.p = p;
%save('b.mat','b');

% Saving the model for phno==2, k=3
%c ={}
%c.mu = mu;
%c.s2 = s2;
%c.p = p;
%save('c.mat','c');

%Saving the model for phno==2, k=6
%d ={}
%d.mu = mu;
%d.s2 = s2;
%d.p = p;
%save('d.mat','d');
```

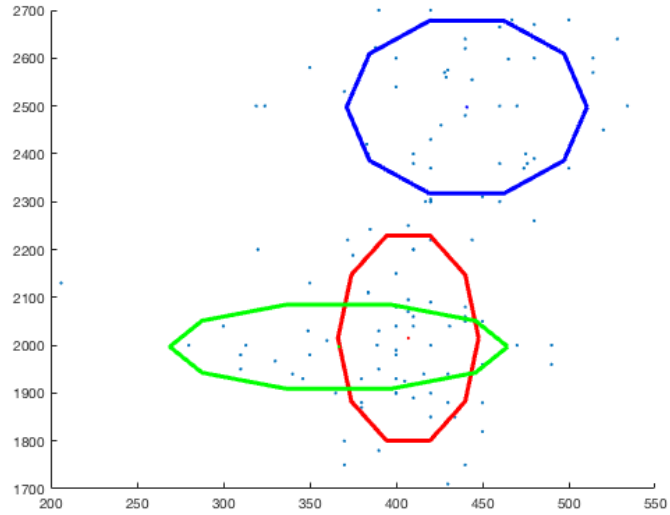
For K=3 phoneme1

Mu	F1	336.374601096605	270.313322571615	282.906908867921
	F2	2824.45622022999	2285.09696439105	2770.57968271356
S2		val(:,:,1) = 1.0e+04 * 0.1570 0 0 1.8796	val(:,:,2) = 1.0e+04 * 0.1214 0 0 1.4243	val(:,:,3) = 1.0e+03 * 2.3574 0 0 4.5693
p		0.204550283419668	0.545399355983121	0.250050360597210



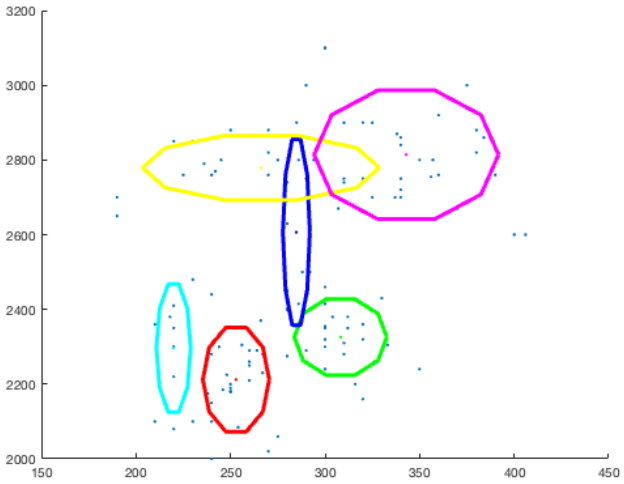
K=3 Phoneme 2

Mu	F1	407.062770787854	366.774484426137	440.920072872354
	F2	2014.93094620342	1996.93763056165	2497.57916439817
S2		val(:,:,1) = 1.0e+04 * 0.0829 0 0 2.5365	val(:,:,2) = 1.0e+03 * 4.7863 0 0 4.2658	val(:,:,3) = 1.0e+04 * 0.2422 0 0 1.8022
p		0.408765911960510	0.173628001460273	0.417606086579217



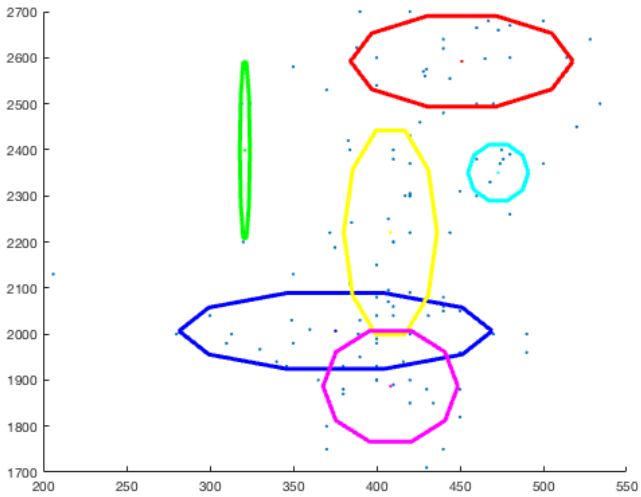
For K=6 Phoneme 1

Mu	F1	252.9245	308.1979	284.8291	266.1293	342.9769	219.8678
	F2	2212.2748	2325.7953	2606.6692	2779.1908	2814.0669	2296.3105
S2		val(:,:,1) = 1.0e+04 * 0.0154 0 0 1.0741	val(:,:,2) = 1.0e+03 * 0.2991 0 0 5.7263	val(:,:,3) = 1.0e+04 * 0.0025 0 0 3.4243	val(:,:,4) = 1.0e+03 * 1.9596 0 0 4.1074	val(:,:,5) = 1.0e+04 * 0.1189 0 0 1.6406	val(:,:,6) = 1.0e+04 * 0.0041 0 0 1.6235
p		0.2228	0.1950	0.0942	0.1774	0.2290	0.0815



For K=6 Phoneme 2

Mu F1	451.0658	320.9782	375.3940	408.2836	408.3195	472.9222
F2	2592.2318	2398.9782	2006.6177	2220.5568	1886.5138	2350.4784
S2	val(:,:,1) = 1.0e+03 * 2.2282 0 0 5.3490	val(:,:,2) = 1.0e+04 * 0.0005 0 0 2.0103	val(:,:,3) = 1.0e+03 * 4.4084 0 0 3.7572	val(:,:,4) = 1.0e+04 * 0.0392 0 0 2.7149	val(:,:,5) = 1.0e+03 * 0.8174 0 0 8.0120	val(:,:,6) = 1.0e+03 * 0.1642 0 0 2.0305
p	0.2349	0.0243	0.2166	0.2571	0.1879	0.0791



Task 3

Now, in task 3, the aim is to classify phoneme1 and phoneme2 for k=3 and k=6 individually. The results from task 2 have been used to build the classifier. They have been classified using the ML (maximum likelihood criterion by implementing the EM algorithm). The trained models for phoneme1 and phoneme2 for k =3 and k=6 respectively have been now used on the test data that takes the last 31 training examples. The required classifiers have been built for k=3 and k=6 and the misclassification rate has been noted.

The classifiers have been saved in task3_k3.m and task3_k6.m respectively. The following lines of code have been written to generate the results. The code aims to give the misclassification rate of examples belonging to phoneme 1 and phoneme2. The test set has been combined with the last 31 examples from each phoneme. So, 62 training examples in total. The code in mog.m has been modified to compute the EM algorithm for Gaussian mixtures (Z1 for model 1 and Z2 for model 2 respectively. Then, the task is to compare both, to decide which phoneme belongs to which model. In order to do this, I have computed the ground truth in the code to detect the true label of the phonemes and then used the max function to decide which classification the value

$$p(\underline{x}) = \sum_{k=1}^{K} \frac{p(c_k)}{(2\pi)^{D/2} \det(\Sigma_k)^{-0.5}} exp(-\frac{1}{2}(\underline{x} - \underline{\mu}_k)^T \Sigma_k^{-1} (\underline{x} - \underline{\mu}_k))$$

This has been given in the assignment although in the denominator, it should be det($\sum_k$)$^{0.5}$ not det($\sum_k$)$^{-0.5}$ since it's already in the denominator. Thus, I have implemented the corrected formula below (as given in the lecture slides and resources for the actual formula for MoG, to calculate the probability p(x).

In task3_k3.m

```
%loading the model for phno =1 for k=3
load('a.mat');
mu_1 = a.mu;
s2_1 = a.s2;
p1 = a.p;

%loading the model for phno =2 for k=3
load('b.mat');
mu_2 = b.mu;
s2_2 = b.s2;
p2 = b.p;


% Initialise parameters
load('PB_data.mat');
J = [f1 f2];

%phoneme sets
set1 = find(phno==1);
set2 = find(phno==2);

% test set
x_t1 = J(set1(122:152),:);
x_t2 = J(set2(122:152),:);
% combining for phno 1 and phno 2 in one test set
x_t = vertcat(x_t1, x_t2);

%ground_truth is used to detect the true labels of the phonemes
ground_truth = vertcat(ones(length(x_t1), 1), ones(length(x_t2), 1) + 1);


 %Classifying for k =3
 [n D] = size(x_t);
 k=3;

 % p1(x) for phno 1
 for i=1:k
      Z1(:,i) = p1(i)*(2*pi)^(-D*0.5)*det(s2_1(:,:,i))^(-0.5)*exp(-0.5*sum((x_t'-
repmat(mu_1(:,i),1,n))'*inv(s2_1(:,:,i)).*(x_t'-repmat(mu_1(:,i),1,n))',2));
 end
    sum_1 = sum(Z1,2);

%p2(x) for phno 2
 for i=1:k
      Z2(:,i) = p2(i)*(2*pi)^(-D*0.5)*det(s2_2(:,:,i))^(-0.5)*exp(-0.5*sum((x_t'-
repmat(mu_2(:,i),1,n))'*inv(s2_2(:,:,i)).*(x_t'-repmat(mu_2(:,i),1,n))',2));
 end
    sum_2 = sum(Z2,2);
```

```
  % phoneme classifier
    compare = [sum_1, sum_2];
    % comparing the values in each phoneme using the max function
    [~, model] = max(compare, [], 2)

   %accuracy and the misclassification rate
   test_acc = sum((model == ground_truth)/length(ground_truth))*100
   test_error = 100 – test_acc
```

test accuracy rate= 91.9355%
test error rate = 8.0645%

Thus, for k=3, the misclassification rate for phoneme1 and phoneme2 is 8.0645% and the accuracy rate is 91.9355%.

In task3_k6.m

```
%loading the model for phno =1 for k=6
load('c.mat');
mu_3 = c.mu;
s2_3 = c.s2;
p3 = c.p;

%loading the model for phno =2 for k=6
load('d.mat');
mu_4 = d.mu;
s2_4 = d.s2;
p4 = d.p;

% Initialise parameters
load('PB_data.mat');
J = [f1 f2];

%phoneme sets
set1 = find(phno==1);
set2 = find(phno==2);

% test set
x_t1 = J(set1(122:152),:);%J(set1(101:152),:);
x_t2 = J(set2(122:152),:);%J(set2(101:152),:);

% combining for phno 1 and phno 2 in one test set
x_t = vertcat(x_t1, x_t2);

ground_truth = vertcat(ones(length(x_t1), 1), ones(length(x_t2), 1) + 1);


%Classifying for model 1 = phoneme 1 , k =6
[n D] = size(x_t);
 k=6;

 % p1(x) for phno 1
 for i=1:k
     Z3(:,i) = p3(i)*(2*pi)^(-D*0.5)*det(s2_3(:,:,i))^(-0.5)*exp(-0.5*sum((x_t'-
repmat(mu_3(:,i),1,n))'*inv(s2_3(:,:,i)).*(x_t'-repmat(mu_3(:,i),1,n))',2));
 end
    sum_3 = sum(Z3,2);

 %p2(x) for phno 2
 for i=1:k
      Z4(:,i) = p4(i)*(2*pi)^(-D*0.5)*det(s2_4(:,:,i))^(-0.5)*exp(-0.5*sum((x_t'-
repmat(mu_4(:,i),1,n))'*inv(s2_4(:,:,i)).*(x_t'-repmat(mu_4(:,i),1,n))',2));
 end
    sum_4 = sum(Z4,2);
```

```
    % phoneme classifier
    compare = [sum_3, sum_4];
    % comparing the values in each phoneme using the max function
    [~, model] = max(compare, [], 2)

   %accuracy and the misclassification rate
   test_acc = sum((model == ground_truth)/length(ground_truth))*100
   test_error = 100 – test_acc
```

Test accuracy rate = 90.3266%
Test error = 9.6774%

Thus, for k=6, the misclassification error for phoneme1 and phoneme2 is 9.68% and the accuracy is 90.32%.

Thus, it can be seen that classifier performs better for k=3 (91.9355%) than k=6 (90.3266%), as the accuracy is higher for k=3 classifier. This is likely to happen to due to over-fitting. This can be justified by testing the train accuracy for k = 3 and k =6 (classifying for the train data). The code has been saved in task3K3train.m and task3K6train.m respectively.

The code is the same as the previous steps, although the data parameters for the train data are altered as given below:

```
% Initialise parameters
J = [f1 f2];
set1 = find(phno==1);
set2 = find(phno==2);
x_t1 = J(set1(1:121),:);
x_t2 = J(set2(1:121),:);
x_t = vertcat(x_t1,x_t2);
```

When we run the code, we get the following results –
For k=3
Train accuracy rate = 96.2810%
Train error rate= 3.7190%

For k=6
Train accuracy rate= 96.6942%
Train error rate= 3.3058%

It can be seen, the train accuracy for k=6(96.6942%) is slightly higher than k=3(96.2810%). This combined with the test accuracy results where k=3 has a higher test accuracy than k=6, it is evident that the for k=6, the model over-fits as the k value increases however the size of the train data remains the same.

Task 4

A meshgrid is created for both phoneme1 and phoneme2 (PB12.mat). A classification matrix is created using classifier for k=3. The following codes of lines have been implemented in task4.m:

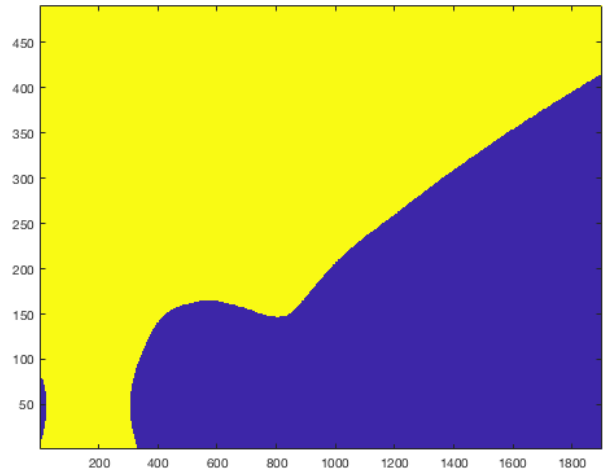
```
load('PB12.mat');
%concatenating the data into one matrix
X = vertcat(X1, X2);
%calculating the min and max for X1(phoneme1)
min_x1 = min(X(:,1));
max_x1 = max(X(:,1));
%calculating the min and max for X2(phoneme2)
min_x2 = min(X(:,2));
max_x2 = max(X(:,2));
```

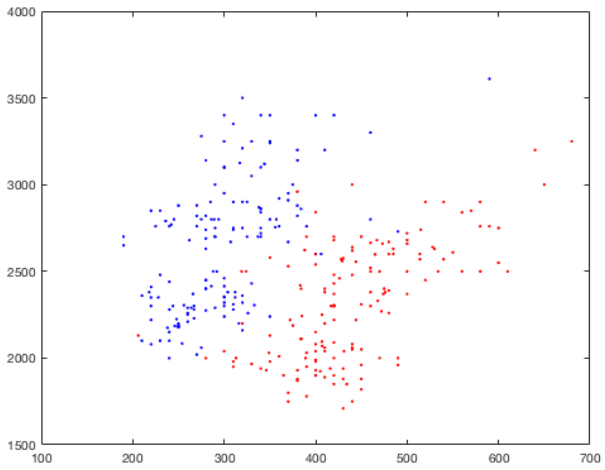
```
%making a grid for all the values between the min and max of X1 and X2
[grid1, grid2] = meshgrid(min_x1:max_x1,min_x2:max_x2) ;


M = [grid1(:), grid2(:)];

%loading the model for phno =2 for k=3
model_1= load('a.mat');
mu_1 = a.mu;
s2_1 = a.s2;
p1 = a.p;

%loading the model for phno =2 for k=3
model_2 = load('b.mat');
mu_2 = b.mu;
s2_2 = b.s2;
p2 = b.p;


% Initialise parameters
x1 = M;



 %Classifying for model 1 = phoneme 1 , k =3
 [n D] = size(x1);
 k=3;

 % p1(x) for phno 1
 for i=1:k
      Z_1(:,i) = p1(i)*(2*pi)^(-D*0.5)*det(s2_1(:,:,i))^(-0.5)*exp(-0.5*sum((x1'-
repmat(mu_1(:,i),1,n))'*inv(s2_1(:,:,i)).*(x1'-repmat(mu_1(:,i),1,n))',2));
 end
    sum_a = sum(Z_1,2);

%p2(x) for phno 2
 for i=1:k
      Z_2(:,i) = p2(i)*(2*pi)^(-D*0.5)*det(s2_2(:,:,i))^(-0.5)*exp(-0.5*sum((x1'-
repmat(mu_2(:,i),1,n))'*inv(s2_2(:,:,i)).*(x1'-repmat(mu_2(:,i),1,n))',2));
 end
    sum_b = sum(Z_2,2);


    % vector of classications
    assign1 = sum_a >= sum_b;

    %checking whether example belongs to phoneme1 or phoneme2

    lh = [sum_a, sum_b];
    [~, M1] = max(lh, [], 2);

    %reshaping the vector into a grid
    M1 = reshape(M1, [1901, 491]);

    %plotting the grid
    imagesc(M1);
```

Once, the classification matrix has been plotted, to check which colour belongs to which phoneme the following code is generated -

```
    %to check which color belongs to which phoneme
    %then compare with the plot of the classification matrix
    plot(X1(:,1),X1(:,2), 'b.'); %phoneme1
    hold on;
    plot(X2(:,1),X2(:,2), 'r.'); %phoneme2
```

The plot for the classification matrix is rotated to start from (0,0 axis). This rotated plot is depicted below and you can check which colour belongs to which class by comparing it with the plot for the phoneme 1 and 2.





On comparing the plots above, it can be noticed that that yellow region belongs to phoneme1 and blue region belongs to phoneme2.

Task 5

The code in mog.m is changed so that a MoG with a full covariance is to fit the data. The modified code is saved in task5.m. Also, the data vector J is set to be a 3-dimensional vector as follows –

```
 % Initialise parameters
data = load('PB_data.mat');
J = [f1 f2, f1+f2];
%for phno =1
set1 = find(phno==1);
%for phno =2
set2 = find(phno==2);
%the training set
x = J(set1(1:121),:);
```

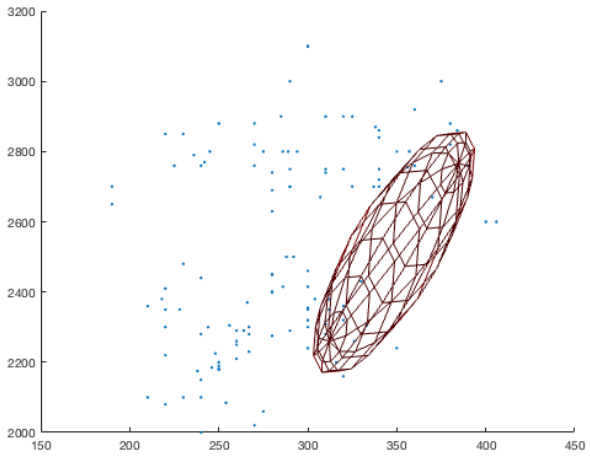
To fit the model with the full covariance, the first line is commented and the second line is uncommented.

```
% We will fit Gaussians with diagonal covariances:

%s2(:,:,i) = diag((x'-repmat(mu(:,i),1,n)).^2*Z(:,i)./sum(Z(:,i)));
% To fit general Gaussians use the line:
s2(:,:,i) = (x'-repmat(mu(:,i),1,n))*(repmat(Z(:,i),1,D).*(x'-
repmat(mu(:,i),1,n))')./sum(Z(:,i));
```

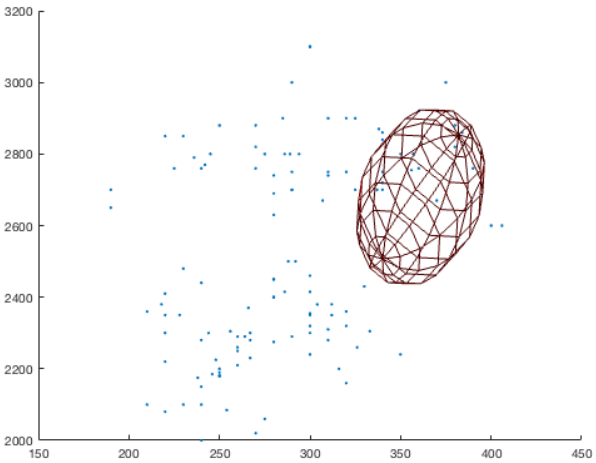
When we run this code, we get an error saying 'Numerical Error in Loop - Possibly Singular Matrix' after just completing the 1st iteration. This problem of singularity occurs because of 'over-fitting and due to the curse of dimensionality'. To overcome this singularity problem, we need to regularise the e-step. This is done by adding a diagonal matrix (identity I) to the covariance.

As can be seen the graphs are not accurate –

For k =3



For k=6



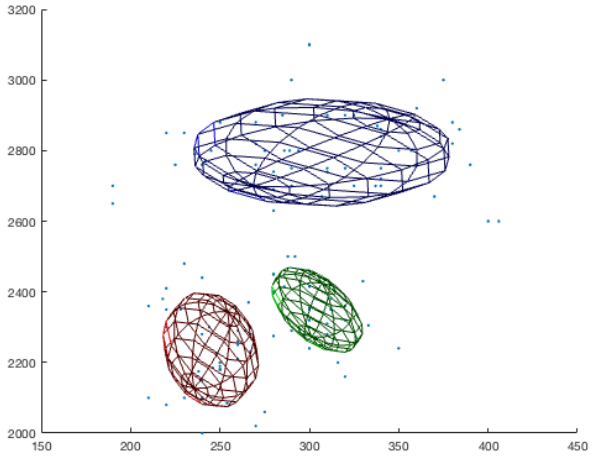
$\Sigma_0 = \sigma^2 I$ (where σ^2 is taken as 0.2 in the code – above some minimum value)

The code has been modified to include the Identity matrix-

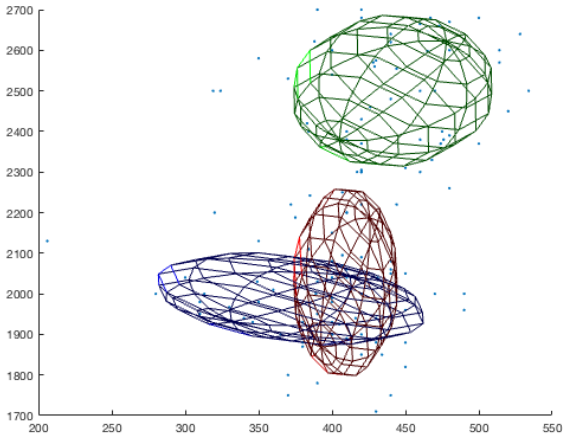
```
 % To fit general Gaussians use the line:
 %to avoid the singularity problem, we regularise the e-step
 %regularise by adding the identity matrix to the covariance
s2(:,:,i) = (x'-repmat(mu(:,i),1,n))*(repmat(Z(:,i),1,D).*(x'-
repmat(mu(:,i),1,n))')./sum(Z(:,i))+ eye(D)*0.2;
```

After the modifications, the code has been run for k =3 and k=6 for phno1 and phno2. The following graphs have been generated-

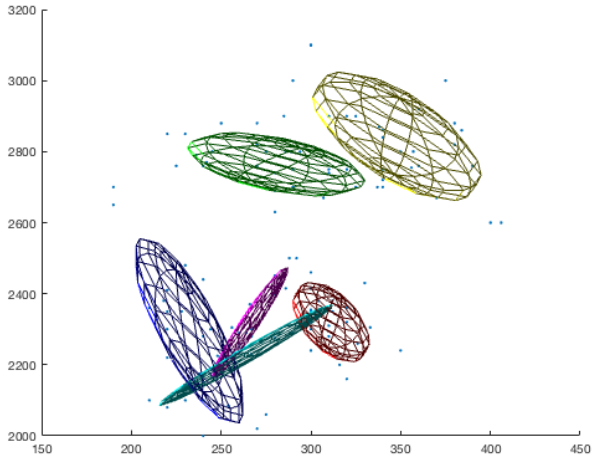
For k =3, phoneme1



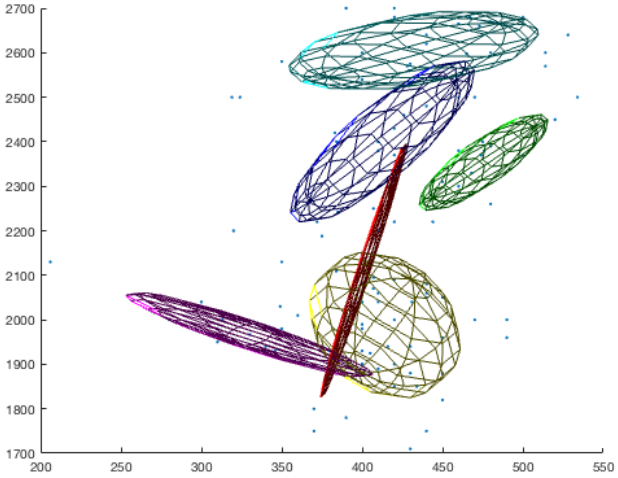
k =3, phoneme2



For k = 6, phoneme1



k=6, phoneme2



From the above graphs, it can be seen that after adding the regularisation matrix, the graphs generated for the full covariance matrix for 3D vector J are accurate.