Machine Learning and Pattern Recognition

Homework-4

From: Tejas Krishna Reddy

NUID: 001423166

Index:

Problem 1:

MATLAB Code:

```
clc
clear all
close all

%% Generating the samples
beta_true1 = [0.75,0.20,0.05];
beta_true2 = [0.80, 0.15, 0.05];
beta_true3 = [0.90, 0.05, 0.05];
mu_true(1,:) = [0, 0];
mu_true(2,:) = [3, 0];
mu_true(3,:) = [0, 2];
Sigma_true(:,:,1) = [1 0;0 1];
Sigma_true(:,:,2) = [1 0;0 1];
Sigma_true(:,:,3) = [1 0;0 1];


%% For 1st Beta component value.
BIC = zeros(1,5);     % M ranges from 1 to 5.
BIC_min_index = zeros(1,1000); % the number of monte_carlo_runs
BIC_MxR = zeros(1,5); % to start vertical concatenation
for monte_carlo = 1:1000
    % To avoid ill conditioned error
    try
X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true1(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true1(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true1(3))];

% Change the component of data generation for beta_2 and beta_3
combinations.

%X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true2(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true2(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true3(3))];
%X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true3(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true2(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true3(3))];

gm1 =
fitgmdist(X1,3,'MaxIter',500,'covariancetype','diagonal','Regulariza
tionValue',10^-15);


% Now calculate the BIC{m} where M ranges from 1 to 5
gm = cell(1,5);
for M = 1:5
    gm{M} = fitgmdist(X1,M);
    BIC(M)= gm{M}.BIC;
end
%Visualise the components of BIC
```

```
BIC;

% Displaying the number of components for which AIC value is the
least.
[minBIC,numComponents] = min(BIC);
BIC_min_index(monte_carlo) = numComponents;

%For each value of M for R=1000 monte carlo runs storing the BIC in
% a single matrix

BIC_MxR = vertcat(BIC_MxR, BIC); % each column represents M, and row
represents # of montecarlo runs

% the best model is
% gm2 = gm{numComponents};
    catch
        continue;
    end
end
figure(2), histogram(BIC_min_index); title('Histogram of BIC
minimization');

% Printing the theta(m^) values. That is Mu and Sigma values at
which EM
% algorithm operates.
EM_theta_cap_mu = gm1.mu;
EM_theta_cap_sigma = gm1.Sigma;

% To get the values of 5% 50% and 90% BIC values
BIC_MxR(1,:) = []; % Removing the zero padding
sorted_BIC = sort(BIC_MxR);
BIC_5percent = sorted_BIC(0.05*monte_carlo,:);
BIC_50percent = sorted_BIC(0.5*monte_carlo, :);
BIC_90percent = sorted_BIC(0.9*monte_carlo, :);

% potting BIC**percent values Vs M=[1, 2, 3, 4, 5];
% Also, plotting corresponding Medians
figure(3), plot([1 2 3 4 5], BIC_5percent,'r'); title('BIC 5% Vs
M'); xlabel('M'); ylabel('BIC values');
hold on
plot([1 2 3 4 5], BIC_50percent, 'g'); title('BIC 50% Vs M');
xlabel('M'); ylabel('BIC values');
hold on
plot([1 2 3 4 5], BIC_90percent, 'b'); title('BIC **% Vs M');
xlabel('M'); ylabel('BIC values');
hold on
plot(3, BIC_5percent(3),'kx','MarkerSize',12);
hold on
plot(3, BIC_50percent(3),'kx','MarkerSize',12);
hold on
plot(3, BIC_90percent(3),'kx','MarkerSize',12)
legend('BIC_5percent','BIC_50percent', 'BIC_90percent','Medians',
'Location','Best');
hold off
```

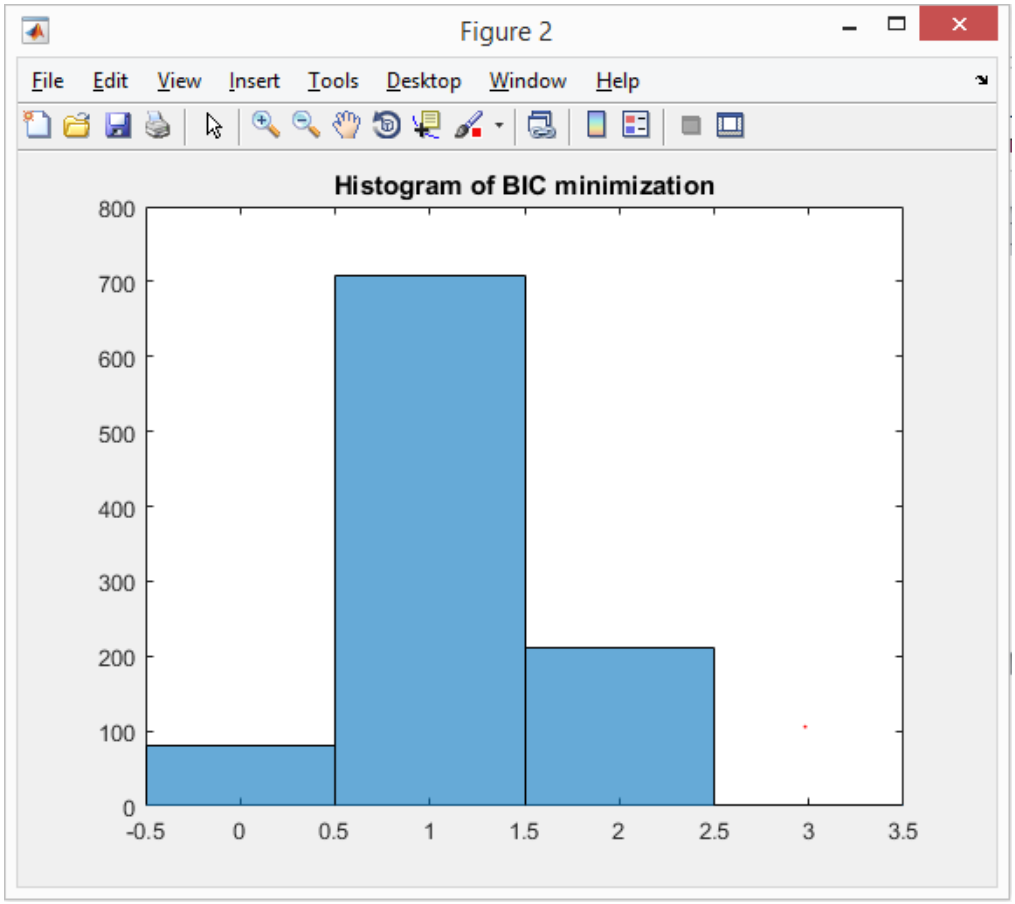
```
%% For Beta_Combination 2

%% For 2nd Beta component value.
BIC = zeros(1,5);     % M ranges from 1 to 5.
BIC_min_index = zeros(1,1000); % the number of monte_carlo_runs
BIC_MxR = zeros(1,5); % to start vertical concatenation
for monte_carlo = 1:1000
    % To avoid ill conditioned error
    try
%X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true1(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true1(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true1(3))];
% Change the component of data generation for beta_2 and beta_3
components..
X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true2(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true2(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true3(3))];
%X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true3(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true2(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true3(3))];
gm1 =
fitgmdist(X1,3,'MaxIter',500,'covariancetype','diagonal','RegularizationVal
ue',10^-15);

% Now calculate the BIC{m} where M ranges from 1 to 5
gm = cell(1,5);
for M = 1:5
    gm{M} = fitgmdist(X1,M);
    BIC(M)= gm{M}.BIC;
end
%Visualise the components of BIC
BIC;

% Displaying the number of components for which AIC value is the least.
[minBIC,numComponents] = min(BIC);
BIC_min_index(monte_carlo) = numComponents;

% For each value of M for R=1000 monte carlo runs storing the BIC in a
% single matrix

BIC_MxR = vertcat(BIC_MxR, BIC); % each column represents M, and row
represents # of montecarlo runs
    catch
        continue;
    end
end
figure(2), histogram(BIC_min_index); title('Histogram of BIC
minimization');

% Printing the theta(m^) values. That is Mu and Sigma values at which EM
% algorithm operates.
EM_theta_cap_mu = gm1.mu
EM_theta_cap_sigma = gm1.Sigma

% To get the values of 5% 50% and 90% BIC values
BIC_MxR(1,:) = []; % Removing the zero padding
sorted_BIC = sort(BIC_MxR);
BIC_5percent = sorted_BIC(0.05*monte_carlo,:);
BIC_50percent = sorted_BIC(0.5*monte_carlo, :);
BIC_90percent = sorted_BIC(0.9*monte_carlo, :);
```

```
% potting BIC**percent values Vs M=[1, 2, 3, 4, 5];
% Also, plotting corresponding Medians
figure(3), plot([1 2 3 4 5], BIC_5percent,'r'); title('BIC 5% Vs M');
xlabel('M'); ylabel('BIC values');
hold on
plot([1 2 3 4 5], BIC_50percent, 'g'); title('BIC 50% Vs M'); xlabel('M');
ylabel('BIC values');
hold on
plot([1 2 3 4 5], BIC_90percent, 'b'); title('BIC **% Vs M'); xlabel('M');
ylabel('BIC values');
hold on
plot(3, BIC_5percent(3),'kx','MarkerSize',12);
hold on
plot(3, BIC_50percent(3),'kx','MarkerSize',12);
hold on
plot(3, BIC_90percent(3),'kx','MarkerSize',12)
legend('BIC_5percent','BIC_50percent', 'BIC_90percent','Medians',
'Location','Best');
hold off


%% For 3rd Combination of Beta given.

%% For 2nd Beta component value.
BIC = zeros(1,5);     % M ranges from 1 to 5.
BIC_min_index = zeros(1,1000); % the number of monte_carlo_runs
BIC_MxR = zeros(1,5); % to start vertical concatenation
for monte_carlo = 1:1000
    % To avoid ill conditioned error
    try

%X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true1(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true1(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true1(3))];
% Change the component of data generation for beta_2 and beta_3
components..
%X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true2(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true2(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true3(3))];
X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true3(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true2(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true3(3))];

gm1 =
fitgmdist(X1,3,'MaxIter',500,'covariancetype','diagonal','RegularizationVal
ue',10^-15);

% Now calculate the BIC{m} where M ranges from 1 to 5
gm = cell(1,5);
for M = 1:5
    gm{M} = fitgmdist(X1,M);
    BIC(M)= gm{M}.BIC;
end
%Visualise the components of BIC
BIC;

% Displaying the number of components for which AIC value is the least.
[minBIC,numComponents] = min(BIC);
BIC_min_index(monte_carlo) = numComponents;
```

```
% For each value of M for R=1000 monte carlo runs storing the BIC in a
% single matrix

BIC_MxR = vertcat(BIC_MxR, BIC); % each column represents M, and row
represents # of montecarlo runs
    catch
        continue;
    end
end
figure(2), histogram(BIC_min_index); title('Histogram of BIC
minimization');

% Printing the theta(m^) values. That is Mu and Sigma values at which EM
% algorithm operates.
EM_theta_cap_mu = gm1.mu
EM_theta_cap_sigma = gm1.Sigma

% To get the values of 5% 50% and 90% BIC values
BIC_MxR(1,:) = []; % Removing the zero padding
sorted_BIC = sort(BIC_MxR);
BIC_5percent = sorted_BIC(0.05*monte_carlo,:);
BIC_50percent = sorted_BIC(0.5*monte_carlo, :);
BIC_90percent = sorted_BIC(0.9*monte_carlo, :);

% potting BIC**percent values Vs M=[1, 2, 3, 4, 5];
% Also, plotting corresponding Medians
figure(3), plot([1 2 3 4 5], BIC_5percent,'r'); title('BIC 5% Vs M');
xlabel('M'); ylabel('BIC values');
hold on
plot([1 2 3 4 5], BIC_50percent, 'g'); title('BIC 50% Vs M'); xlabel('M');
ylabel('BIC values');
hold on
plot([1 2 3 4 5], BIC_90percent, 'b'); title('BIC **% Vs M'); xlabel('M');
ylabel('BIC values');
hold on
plot(3, BIC_5percent(3),'kx','MarkerSize',12);
hold on
plot(3, BIC_50percent(3),'kx','MarkerSize',12);
hold on
plot(3, BIC_90percent(3),'kx','MarkerSize',12)
legend('BIC_5percent','BIC_50percent', 'BIC_90percent','Medians',
'Location','Best');
hold off
```

RESULTS:

The histogram of BIC values for Beta_1 combination of values.

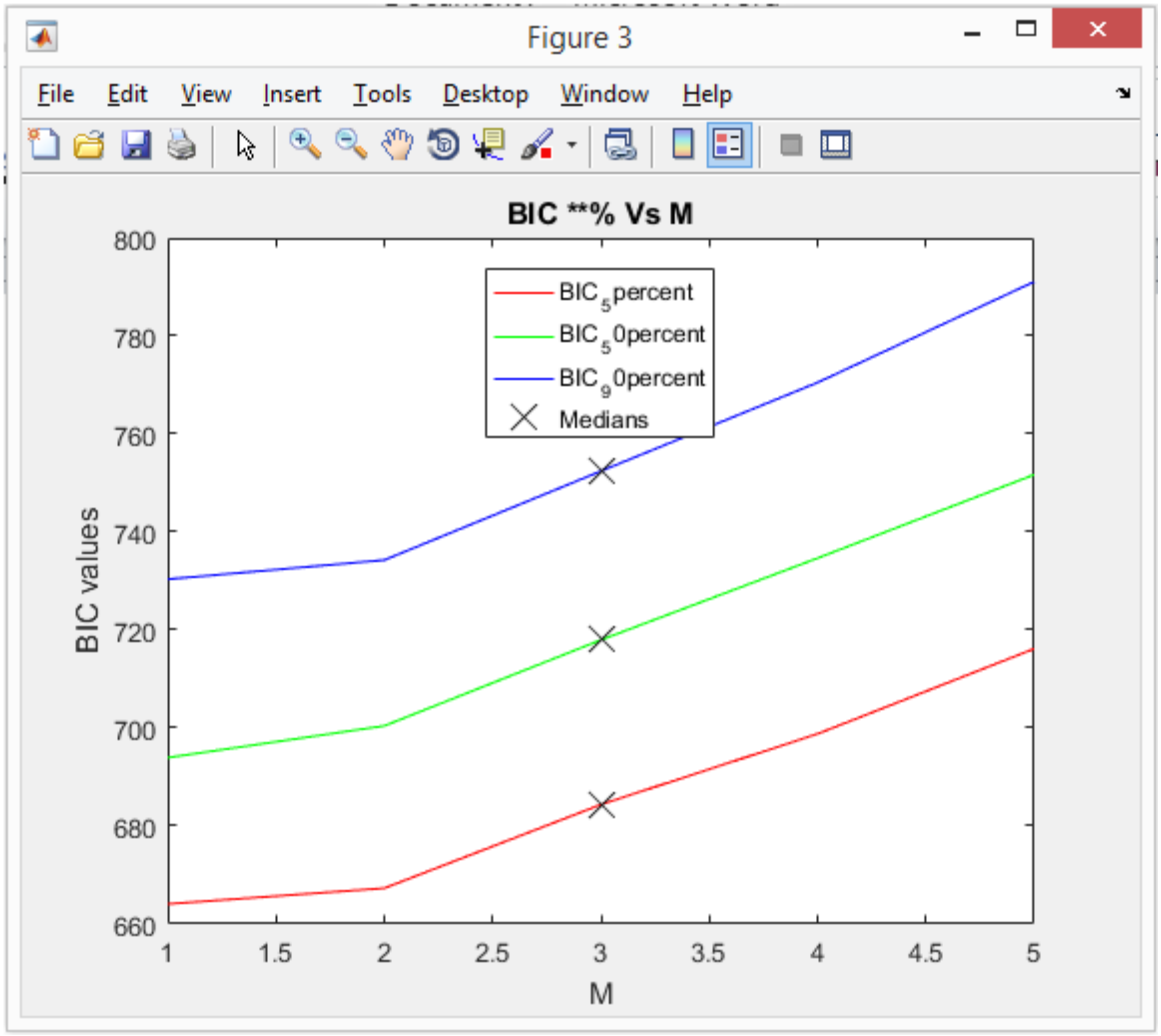


In most cases, BIC value at M=1 is the least. From the above bar graph we can see, that among 1000 monte carlo iterations 700+ iterations had their least BIC value calculated at M=2. The least BIC tends to be towards the max component in the given Beta components.

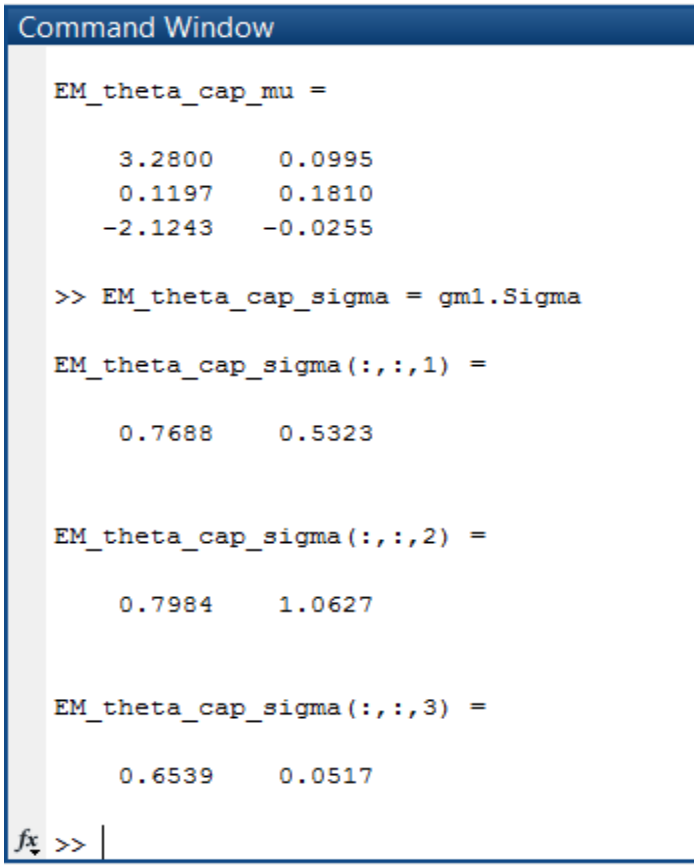
In the figure below, we can see that the BIC at 5%, 50% and 90% results. We can see that, at M=2, there is a gradual rise in slope. The median values over each BIC are marked with a 'cross' in the plotted figures.

The Gaussian distribution fitting was done by the inbuilt function `fitgmdist`. The algorithm was initialised with maximum iterations of 500 to converge and with a very low regularization value.

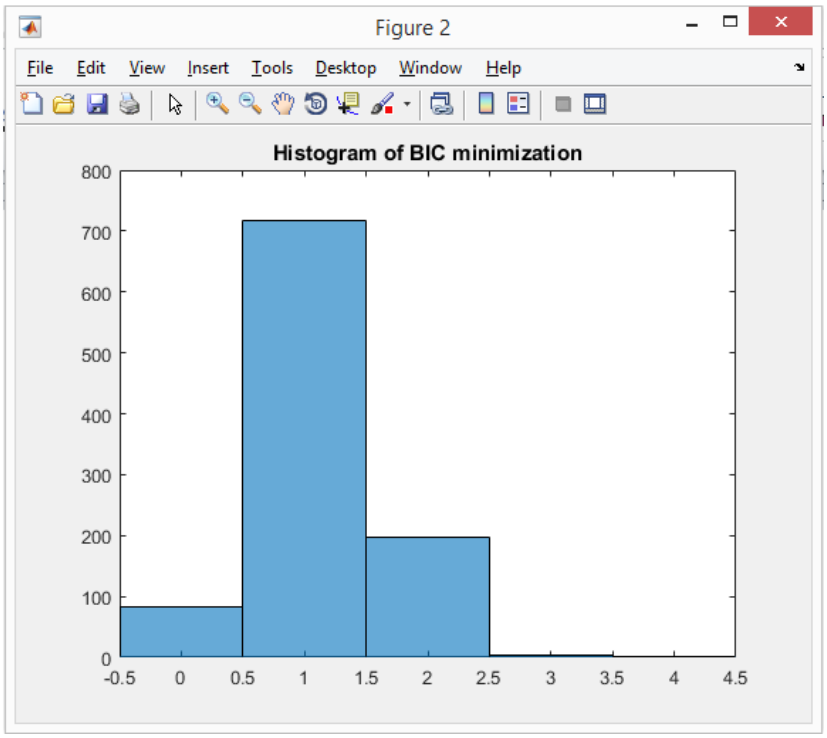
The 5%, 50% and 95% BIC values for Gaussian samples generated with Beta 1 component is as follows –



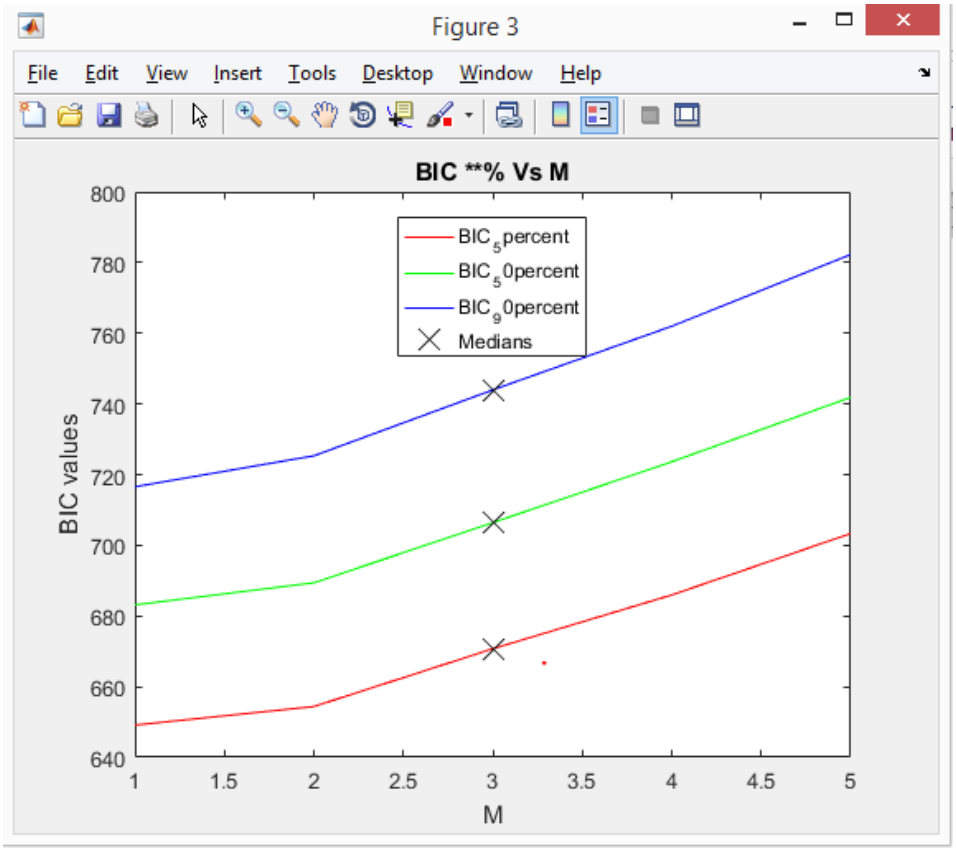
The (Theta^) values that are estimated through EM algorithm is as follows –



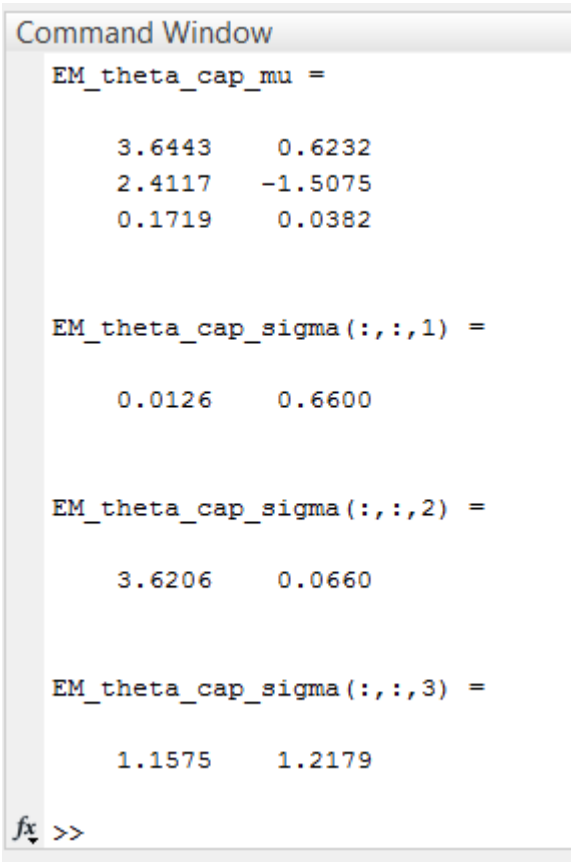
For 2nd Beta Combination, The histogram can be visualised as:



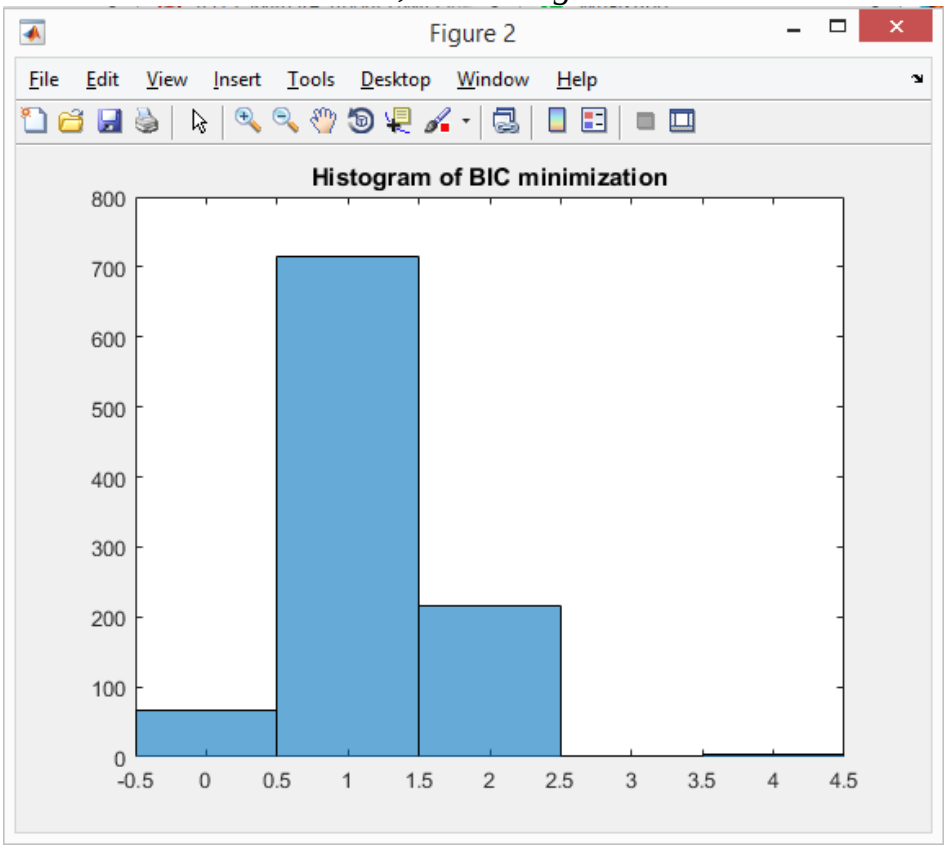
The 5%, 50% and 95% BIC values for Gaussian samples generated with Beta 1 component is as follows –



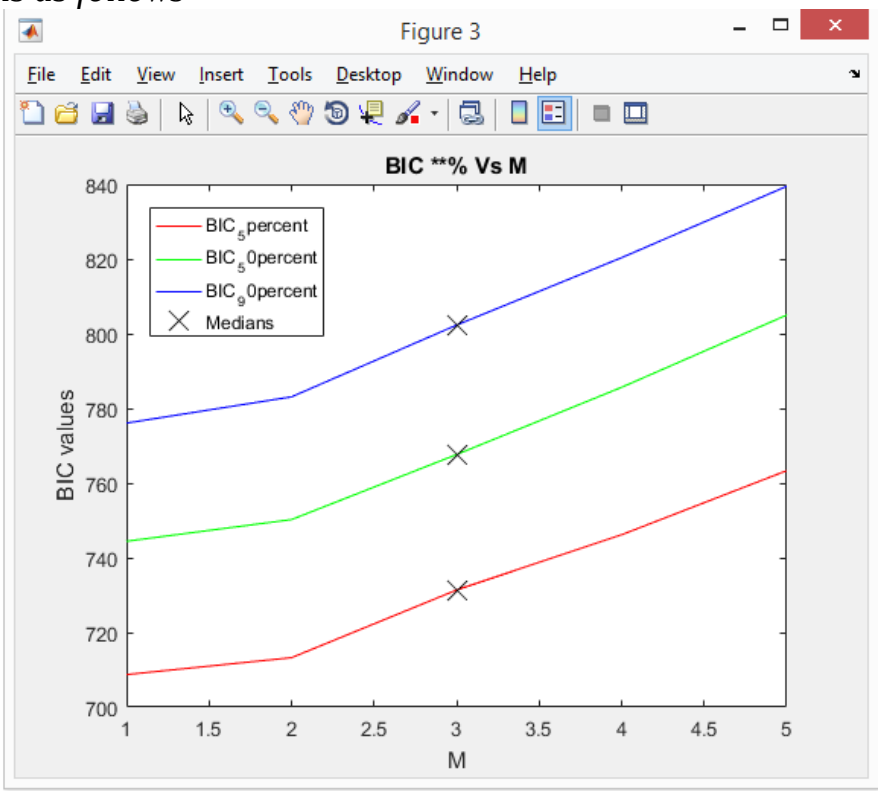
The (Theta^) values that are estimated through EM algorithm is as follows –



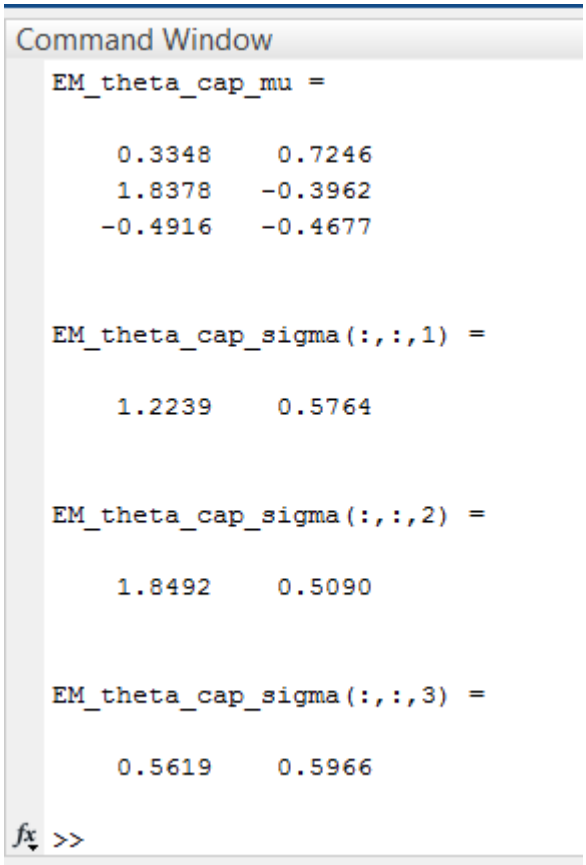
For 3rd Beta Combination, The histogram can be visualised as:



The 5%, 50% and 95% BIC values for Gaussian samples generated with Beta 1 component is as follows –



The (Theta^) values that are estimated through EM algorithm is as follows –



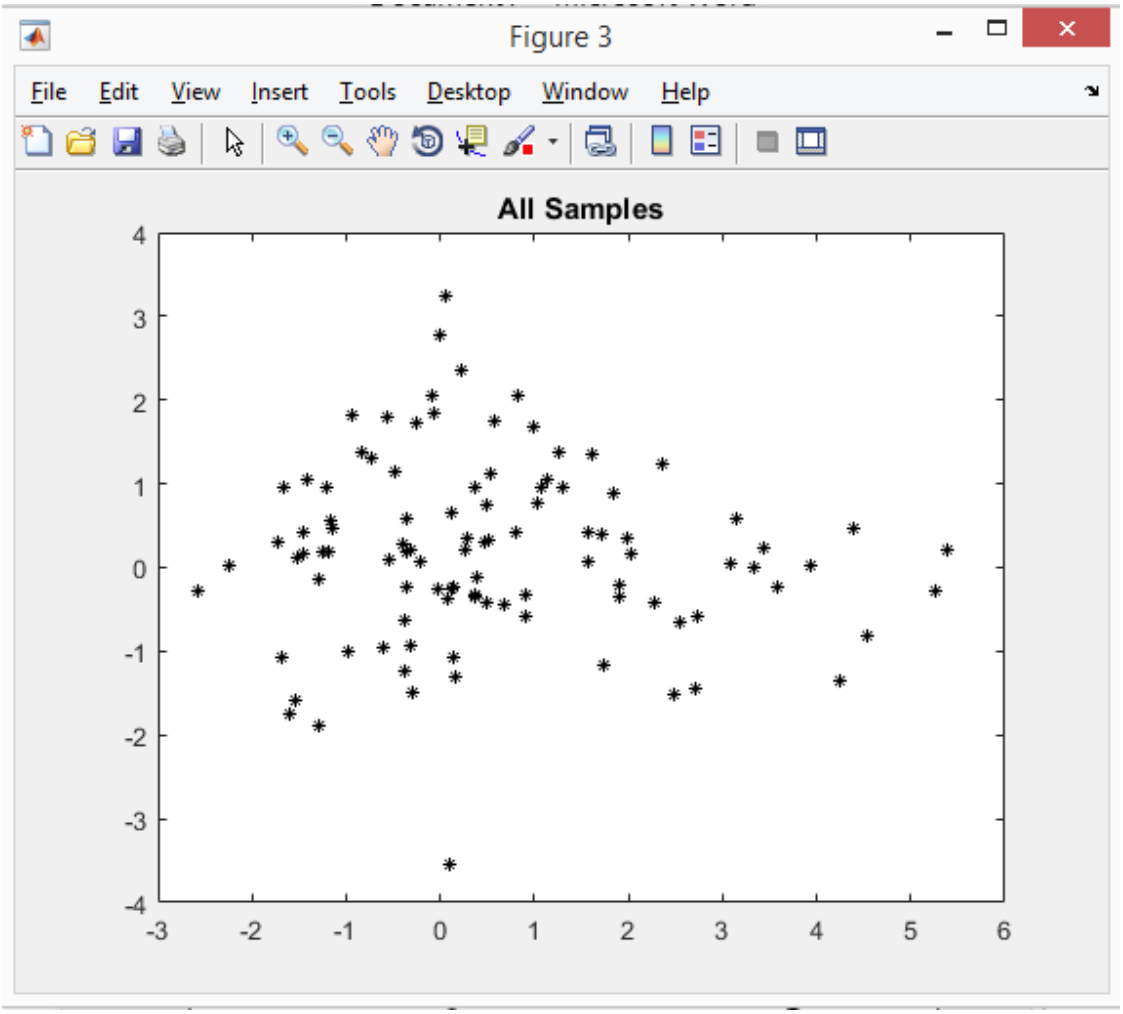
Problem 2:

MATLAB CODE:

```
clc
clear all
close all

beta_true1 = [0.75,0.20,0.05];
beta_true2 = [0.80, 0.15, 0.05];
beta_true3 = [0.90, 0.05, 0.05];
mu_true(1,:) = [0, 0];
mu_true(2,:) = [3, 0];
mu_true(3,:) = [0, 2];
Sigma_true(:,:,1) = [1 0;0 1];
Sigma_true(:,:,2) = [1 0;0 0.5];
Sigma_true(:,:,3) = [0.5 0;0 1];

X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),100*beta_true1(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),100*beta_true1(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),100*beta_true1(3))];
gm1 = fitgmdist(X1,3);

clusterX1 = cluster(gm1, X1);     % Gaussian clustering
figure(1),
h1= scatter(X1(:,1), X1(:,2), clusterX1);   % plain data
figure(2),
h2 = gscatter(X1(:,1), X1(:,2), clusterX1); % clustered GMM datax

% Kmeans clustering
[grp,c] = kmeans(X1,3,'Distance','sqeuclidean');
figure(3);
plot(X1(:,1),X1(:,2),'k*','MarkerSize',5);
figure(4);gscatter(X1(:,1),X1(:,2),grp,'brg','+++');

% Plotting with centroids for final Kmeans clusters

opts = statset('Display','final');
[idx,C] =
kmeans(X1,3,'Distance','sqeuclidean','Replicates',5,'Options',opts);

figure(5);
plot(X1(idx==1,1),X1(idx==1,2),'r.','MarkerSize',12) % class 1
hold on
plot(X1(idx==2,1),X1(idx==2,2),'b.','MarkerSize',12) % class 2
plot(C(:,1),C(:,2),'kx','MarkerSize',15,'LineWidth',3)  % Plotting
centroids
hold on
plot(X1(idx==3,1),X1(idx==3,2),'g.','MarkerSize',12) % class 3
legend('Cluster 1','Cluster 2', 'Centroids','Cluster 3','Location','NW')
title ('Cluster Assignments and Centroids');
hold off
```

Results:

The figure below shows all the Gaussian samples produced.



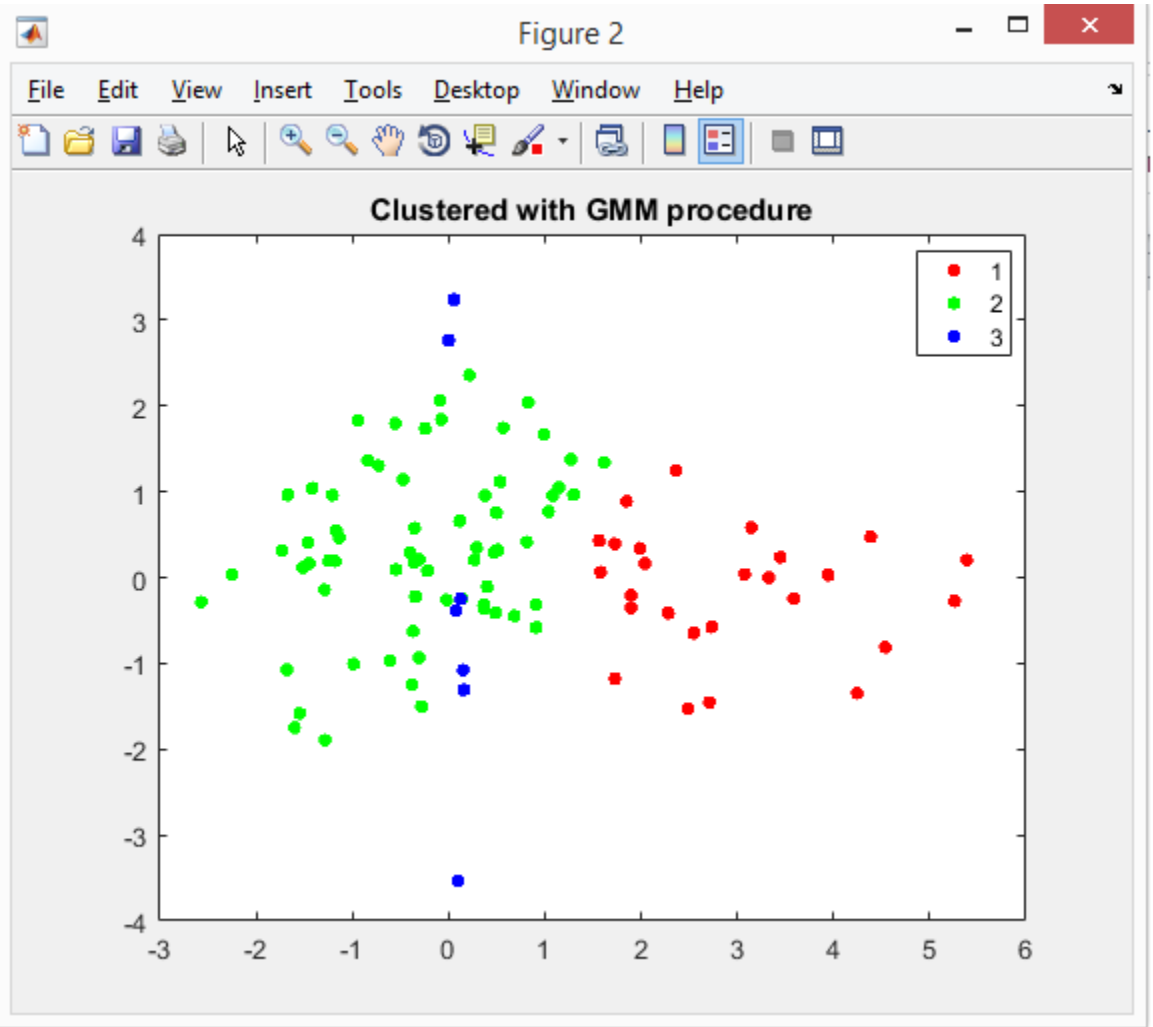
Analysis:

The gaussian samples were generated based on the given conditions. The samples were subjected to clustering using 2 different methods.

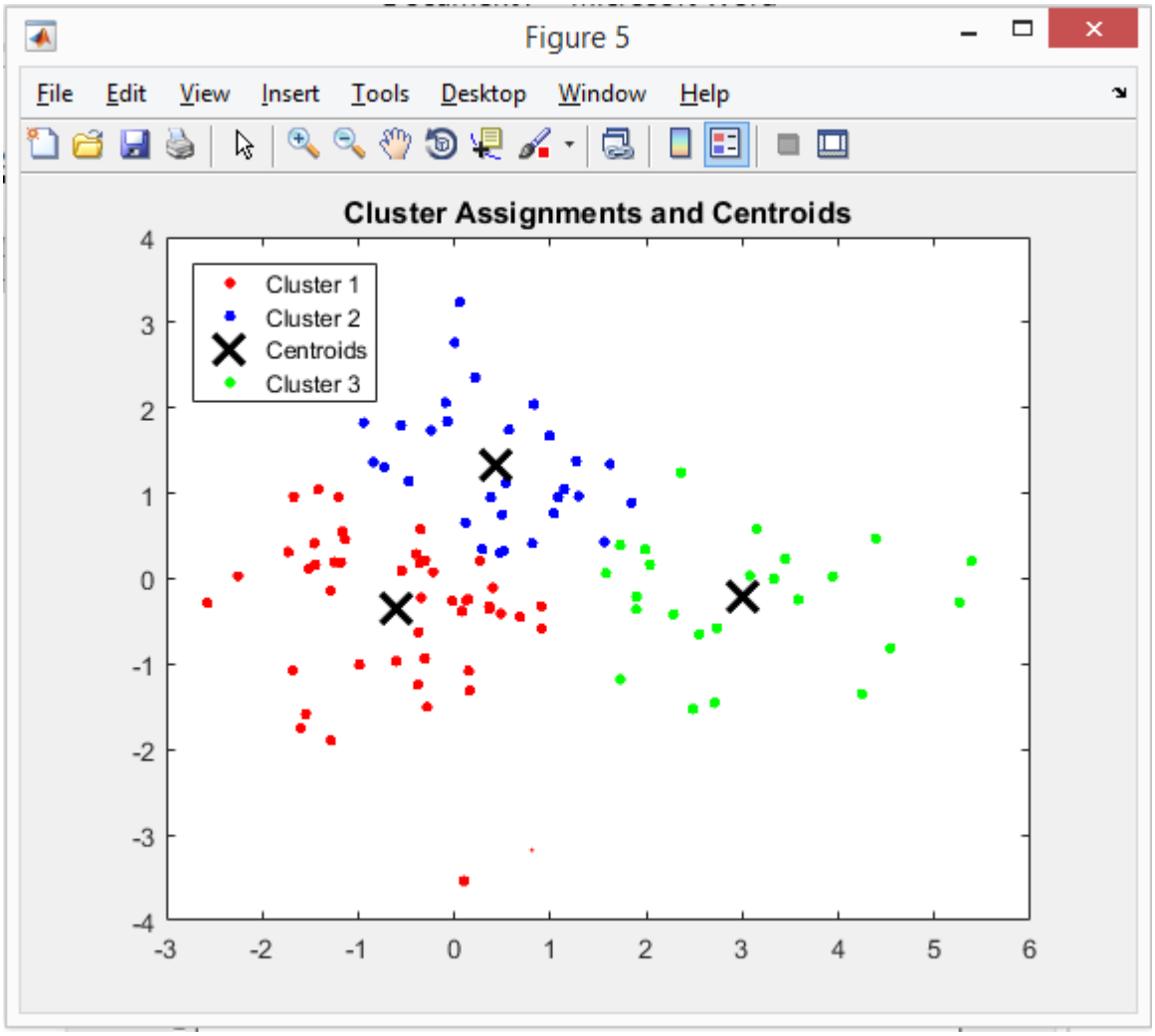
1. The GMM fitting was done using the MATLAB inbuilt function fitgmdist(). Cluster() inbuilt funciton was used over the fit model to cluster the samples such that the index of the Gaussian component with the largest posterior is assigned.
2. Kmeans clustering was performed with 3 components and using Squared Euclidean distance where each centroid is the mean of the points in that cluster.

The results obtained through 1, clusters the samples based on the guassian distribution fitting, the source of the samples/ their Gaussian model. After clustering the number of samples can be visualised to be number of samples generated with the corresponding Beta_component. The results from method 2, are obtained by repeated trials where the 3 clusters are formed with means as their centroids.

The Figure below shows the results after clustering the samples with GMM procedure.



The Figure below shows the results after clustering using Kmeans clustering algorithm.



Problem 3

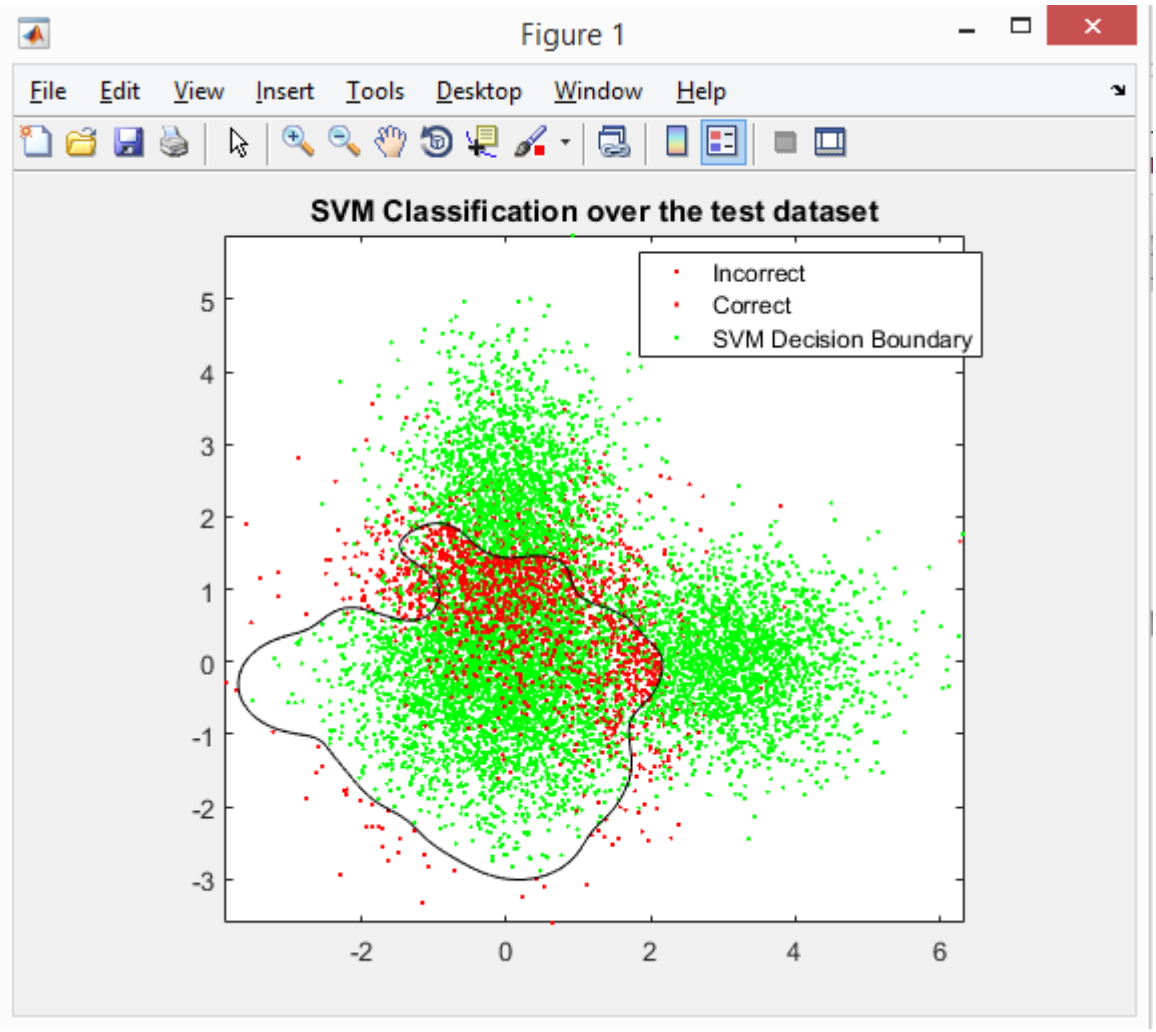
MATLAB Code

```
clc
clear all
close all

%% Prepare training dataset.
N=100;
beta_true1 = [0.5,0.25,0.25];
mu_true(1,:) = [0, 0];
mu_true(2,:) = [3, 0];
mu_true(3,:) = [0, 2];
Sigma_true(:,:,1) = [1 0;0 1];
Sigma_true(:,:,2) = [1 0;0 0.5];
Sigma_true(:,:,3) = [0.5 0;0 1];

X1 = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),N*beta_true1(1));
mvnrnd(mu_true(2,:),Sigma_true(:,:,2),N*beta_true1(2));
mvnrnd(mu_true(3,:),Sigma_true(:,:,3),N*beta_true1(3))];
labels = ones(1,N);              % The first component class labels = +1
Class1_start_index = N*beta_true1(1); % Number of class +1 samples

for i= Class1_start_index: N
    labels(i) = -1;          % Second and Third component guassian samples
are regarded as class -1.
end

%% Now create an SVM model to train the 100 samples generated.
% Guassian Kernel with less box constraint value gives considerably good
% results over the trained model.
% Guassian Kernel - since it is a TALL data than wide data.

SVM_Model =
fitcsvm(X1,labels,'Standardize',true,'KernelFunction','Gaussian',...
    'KernelScale','auto','BoxConstraint', 1,'OutlierFraction', 0.05
);

% 10 fold cross validation of the trained SVM model.
Cross_Validation = crossval(SVM_Model, 'KFold', 10);

% Calculating the K-fold loss of the trained model.
% Emperical probability of error
Loss_train = kfoldLoss(Cross_Validation, 'LossFun', 'ClassifError')

% Now generating test set
N_test = 10000;

X1_test = [mvnrnd(mu_true(1,:),Sigma_true(:,:,1),N_test*beta_true1(1));...
    mvnrnd(mu_true(2,:),Sigma_true(:,:,2),N_test*beta_true1(2));...
    mvnrnd(mu_true(3,:),Sigma_true(:,:,3),N_test*beta_true1(3))];

%% Class labels generation. Componenet 1 = +1, component2 and component3
```

```
% generated values = -1.
labels_test = ones(1,N_test);
Class1_start_index_test = N_test*beta_true1(1); % Number of class +1
samples in test set

for i= Class1_start_index_test: N_test
    labels_test(i) = -1;          % Second and Third component guassian
samples are regarded as class -1.
End

labels_test = transpose(labels_test);
predict_labels = predict(SVM_Model, X1_test);
% Loss_test = loss(SVM_Model,predict_labels, labels_test);

%% Now to calculate the emperical probability of error with the classifier
% Manually check to find the number of misclassified elements.
count =0;
for i=1:N_test
    if(labels_test(i) ~= predict_labels(i)) % checking if misclassified.
        count = count + 1;                  % count the number of
misclassifications
    end
end

Loss_calculated = count/ N_test       %misclassified/ total num of test
data points.

%% Visualise the results obtained.
figure(1),
for i=1:N_test
    if(labels_test(i) == predict_labels(i))
        plot(X1_test(i,1), X1_test(i,2),'g.','MarkerSize',5);
        hold on
    else
         plot(X1_test(i,1), X1_test(i,2),'r.','MarkerSize',5);
         hold on
    end
end

d = 0.02;
[x1Grid,x2Grid] =
meshgrid(min(X1_test(:,1)):d:max(X1_test(:,1)),min(X1_test(:,2)):d:max(X1_t
est(:,2)));
xGrid = [x1Grid(:),x2Grid(:)];
[~,scores] = predict(SVM_Model,xGrid);
%Plotting the Decision Boundry
contour(x1Grid,x2Grid,reshape(scores(:,2),size(x1Grid)),[0 0],'k');
legend({'Incorrect','Correct','SVM Decision Boundary'},'Location','Best');
axis equal
title 'SVM Classification over the test dataset'

hold off
```

Results:

The following figure shows 10000 samples classified by a trained SVM model over 100 samples. The red shows the incorrectly classified data points and green shows the correctly classified data points. The boundary line is clearly delineated.

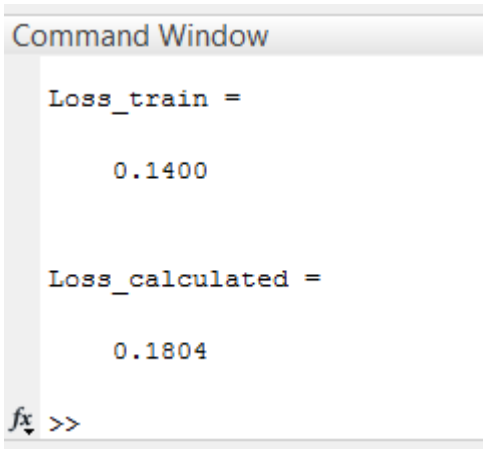


The SVM model was developed by the inbuilt function in MATLAB 'fitcsvm'. The parameters were to Standardize the predictors. Assumed 5% of the samples were outliers, Kernel scaling was chosen to be automatic and Gaussian SVM kernel was chosen over Linear SVM since the data was of the size 10000x2, which is a considerably tall data that is best suited for Gaussian SVM models. A small regularization term was included (BoxConstraint).

From the true class labels and the predicted labels, the number of misclassified data points was calculated.

Empirical probability of error = (num of misclassified datapoints / num of total datapoints) * 100;

Empirical probability calculated is shown below as Loss_calculated



Empirical probability of error = Loss_calculated * 100;

18.04%