Project #7

SAMPLES AND STATISTICS

Tulika Asthana(2671577263) | EE 511 - Simulation of Stochastic Processes | 4/20/2017

Tool : MATLAB v2016a|Windows 10(Professional)

Problem #1

Implement a random number generator for a random vector $X = [X_1, X_2, X_3]^T$ having multivariate Gaussian distribution with

$$\mu = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \quad \Sigma = \begin{bmatrix} 3 & -1 & 1 \\ -1 & 5 & 3 \\ 1 & 3 & 4 \end{bmatrix}$$

– **EXPERIMENT:** The purpose of this problem is to implement a random number generator for a random vector 'X' with a multivariate Gaussian Distribution with the above given specifications. For doing so, we generate 'Z', a standard normal random variable row vector i.e.

$$Z \sim N(0,1)$$

Since X is a vector, it can be expressed as:

$$X = [\ X1\ ,\ X2\ ,\ X3\ ]$$

Then, we can write:

$$X^T = A.(Z^T) + \mu$$

$$\text{where, } \Sigma = A.A^T$$

We can calculate the lower triangular matrix A by using the Choleski Decomposition, which says that:

"*For any n × n symmetric and positive definite matrix* ***M****, there is an n × n lower triangular matrix* ***A*** *such that* ***M*** = ***AA_****, where by lower triangular we mean that all elements in the upper triangle of the matrix are equal to 0.* " (M.Ross, 2013)

Hence, I generated a row vector Z of length # and initialize it with standard normal random variables using normrnd.

I generated the lower triangular square matrix 'A' by using the chol() function provided in MATLAB, and hence obtained the column vector 'X' of length 3.

I have generated N samples of 'X' and calculated the sample mean and covariance.

- **CODE:**

```
function randNumGenerate(N)
XArray = zeros(3,N);
for i = 1:N
    mu = [1 2 3];
    sigma = [  3  -1   1;
              -1   5   3;
               1   3   4 ];
    A = chol(sigma,'lower');
    Z = normrnd(0,1,1,3);
    X = A*Z.' + mu.';
    X = transpose(X);
    for k = 1:3
        XArray(k,i) = X(k);
    end
end
X1Array = zeros(1,N);
X2Array = zeros(1,N);
X3Array = zeros(1,N);
for i = 1:N
X1Array(1,i) = mean(XArray(1,i));
X2Array(1,i) = mean(XArray(2,i));
X3Array(1,i) = mean(XArray(3,i));
end
display('The sample mean is:');
formatSpecMean = 'The mean of the samples is %.2f\n';
meanX1 = mean(X1Array);
meanX2 = mean(X2Array);
meanX3 = mean(X3Array);
fprintf(formatSpecMean,meanX1);
fprintf(formatSpecMean,meanX2);
fprintf(formatSpecMean,meanX3);

covariance = cov(XArray.');
display('The covariance of the samples is:');
display(covariance);

end
```

- **OUTPUT**

```
>> randNumGenerate(1000)
The sample mean is:
The mean of the samples is 1.05
The mean of the samples is 1.94
The mean of the samples is 2.99
The covariance of the samples is:

covariance =

    2.9673   -0.9940    1.0418
   -0.9940    4.7800    2.8178
    1.0418    2.8178    3.9098
```

```
>> randNumGenerate(100000)
The sample mean is:
The mean of the samples is 1.00
The mean of the samples is 2.01
The mean of the samples is 3.01
The covariance of the samples is:

covariance =

    2.9843   -0.9837    0.9994
   -0.9837    4.9968    3.0092
    0.9994    3.0092    4.0101
```

ANALYSIS: As can be viewed from the output, the mean and the covariance of the samples generated matches very closely with the specified mean and covariance. The margin of difference decreases when the number of samples is increased.

Problem #2

Implement a random number generator for a random variable with the following mixture distribution: f(x) = 0.4N(-1,1) + 0.6N(1,1). Generate a histogram and overlay the theoretical p.d.f. of the random variable.

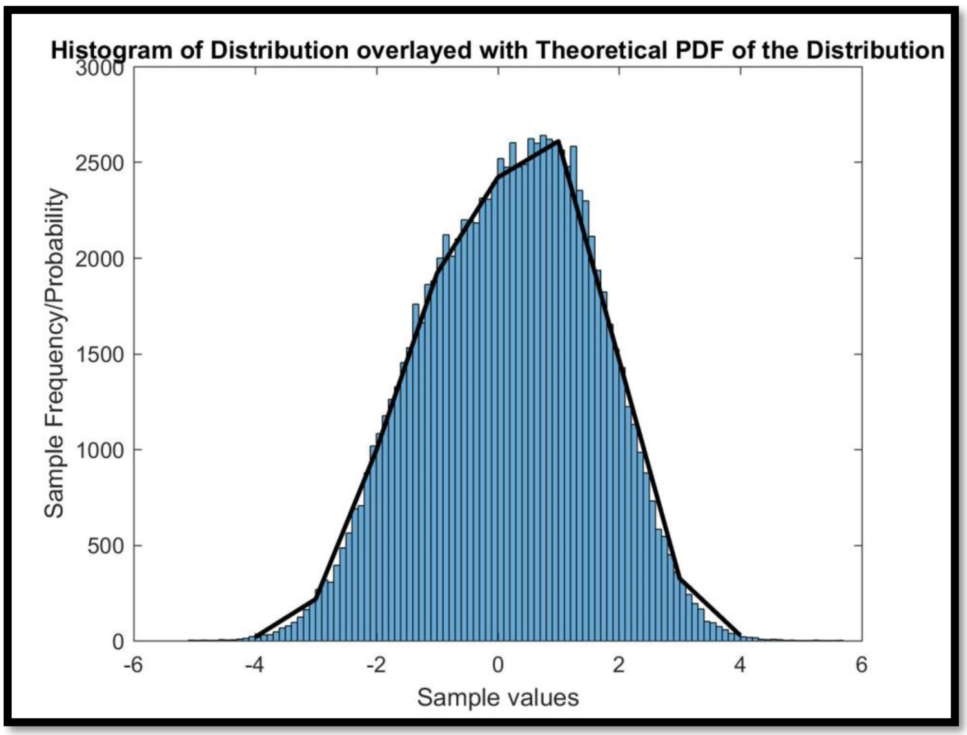
EXPERIMENT: I generated N uniform random variables in X and set a constraint that if :
X(i) <= 0.4
Then the distribution be generated from N(-1,1), otherwise, the distribution be generated from N(1,1). I have then plotted the histogram of X and overlayed it with the theoretical PDF.

- **CODE**

```
function mixtureDist(N)
X = rand(N,1);
Z = zeros(N,1);
%% Distribution Generation
for i = 1:N
    if(X(i) <= 0.4)
        Z(i) = normrnd(-1,1);
    else
        Z(i) = normrnd(1,1);
    end
end
```

```
grid on;
histogram(Z);
x = -4:4;
hold on
%% Histogram and Theoretical PDF
pdf = 10000*(0.4*normpdf(x,-1,1)+ 0.6*normpdf(x,1,1));
plot(x,pdf,'linewidth',2,'color','black');
title('Histogram of Distribution overlayed with Theoretical PDF of the
Distribution');
xlabel('Sample values');
ylabel('Sample Frequency/Probability');
end
```

- **OUTPUT**



- **ANALYSIS:** As can be seen, the theoretical PDF and the histogram of X are overlayed very well together. The quality of fit increases with increae in number of samples of X.

Problem #3

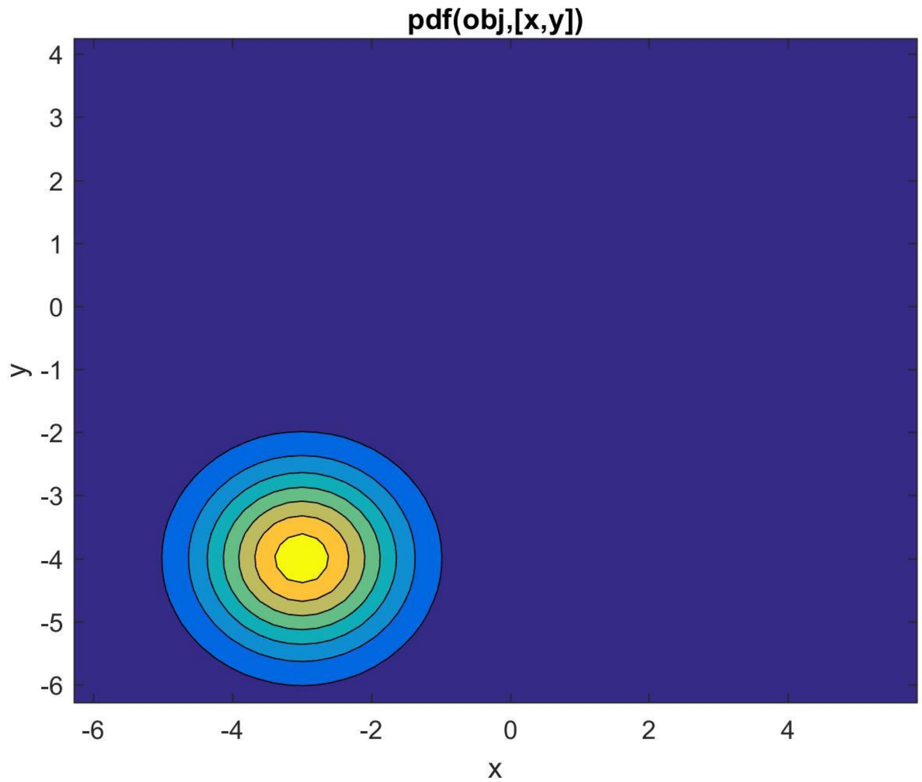
Implement a 2-dimensional random number generator for a Gaussian mixture model (GMM) pdf with 2 subpopulations. Use the expectation maximization (EM) algorithm to estimate the pdf parameters of the 2-D GMM from samples. Compare the quality and speed of your GMM-EM estimates using 300 samples from different GMM distributions (e.g. spherical vs ellipsoidal covariance, close vs well-separated subpopulations, etc.).

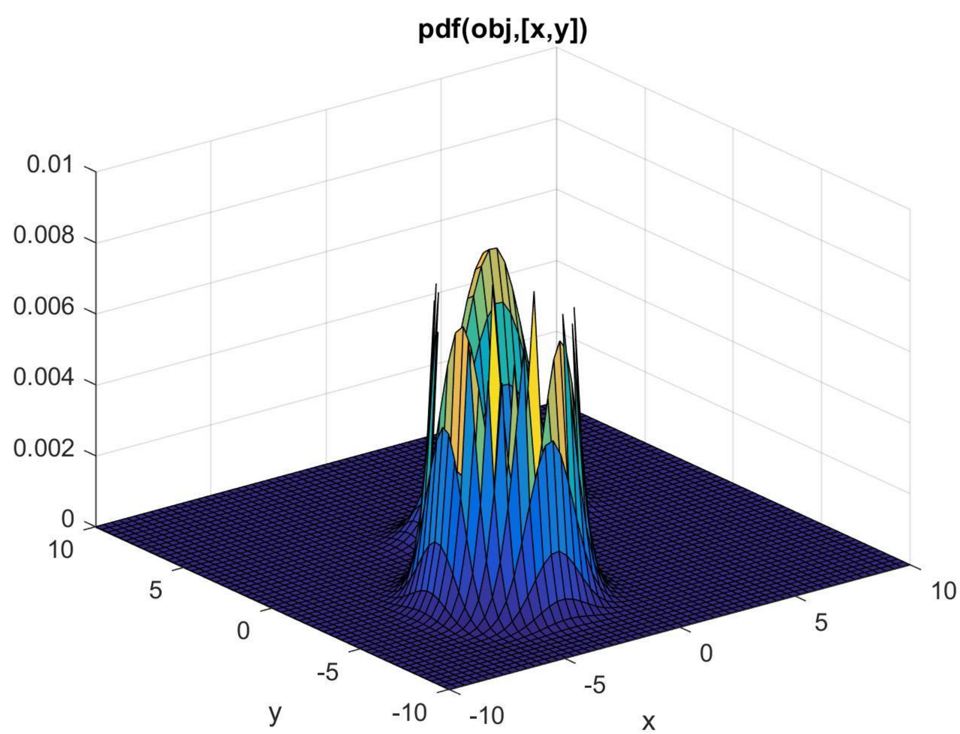
- **EXPERIMENT:** I created a Gaussian Mixture Distribution object using the gmdistribution() function which creates a distribution with the given specifications. I, then, used the random() to randomly choose 300 samples from the distribution 'obj'. The, I used the gmdistribution.fit() function which applies the Expectation Maximization algorithm on the Gaussian Mixture Model.
  I then used the ezcontour() and the ezsurf() to draw the contour and scatter plots.

- **CODE:**

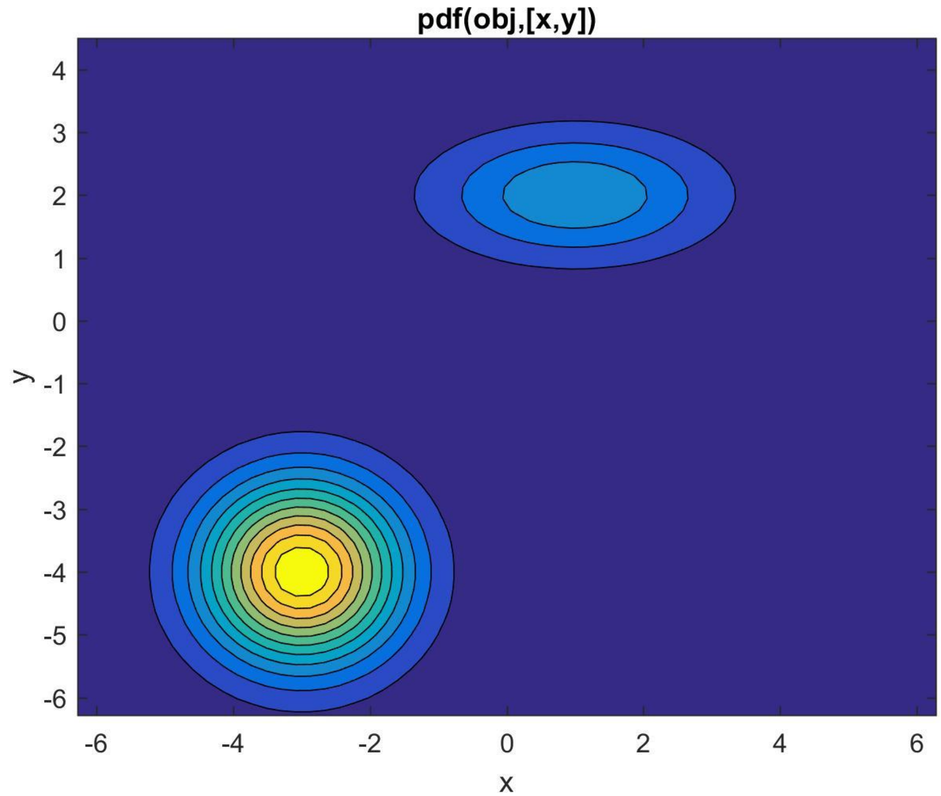
```
function expectationMaximization()

%%  first distribution is centered at (0,0), second at (-1,3)
mu = [1 -2;3 4];

%%  covariance matrix
sigma = cat(3,[2 0;0 .5],[1 0;0 1]);

%%  weight
p = [0.05,0.95];

%%  build GMM
obj = gmdistribution(mu,sigma,p);
samples = random(obj,300);
options = statset('Display','final');
EM = gmdistribution.fit(samples, 2, 'options',options);

%%  2D projection
figure;
ezcontourf(@(x,y) pdf(obj,[x y]));

%%  view PDF surface
figure;
ezsurf(@(x,y)pdf(obj,[x y]),[-10 10],[-10 10])
end
```

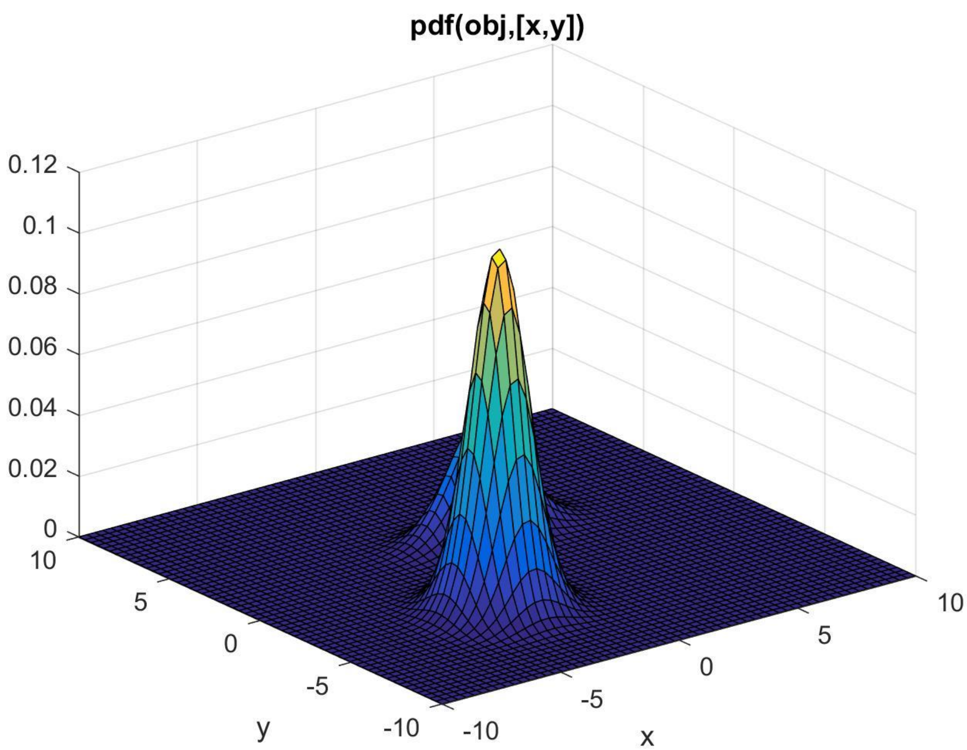
- **OUTPUT:**
  μ = [1 2;-3 -4], p = [0.05 0.95];



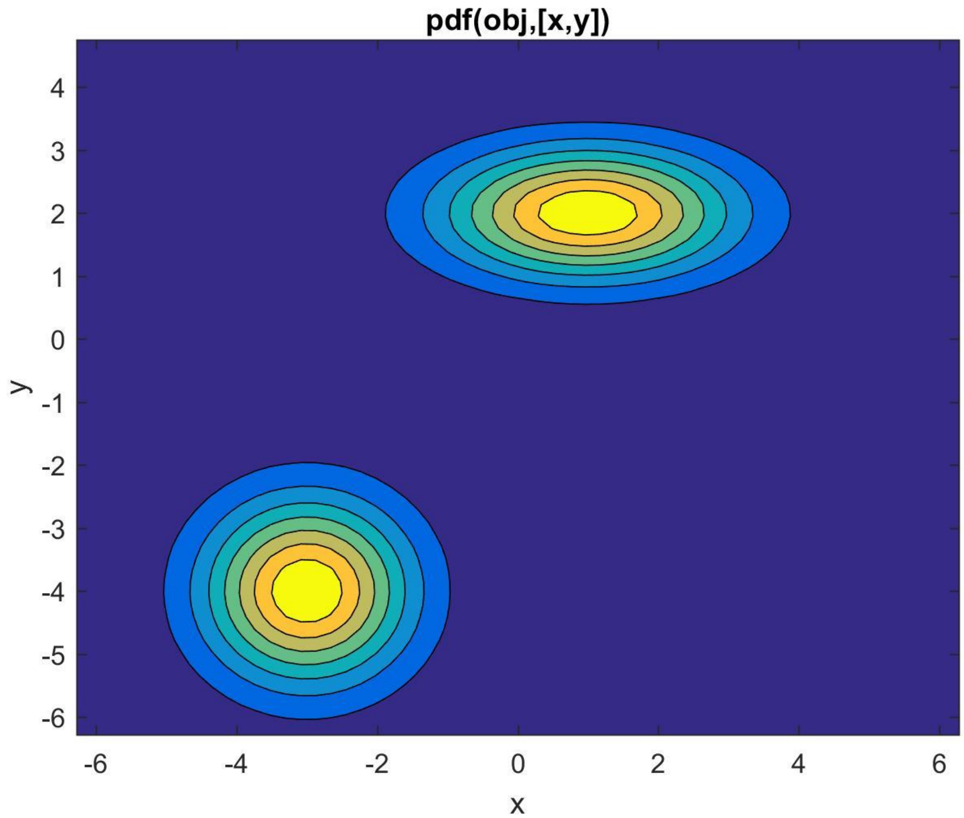


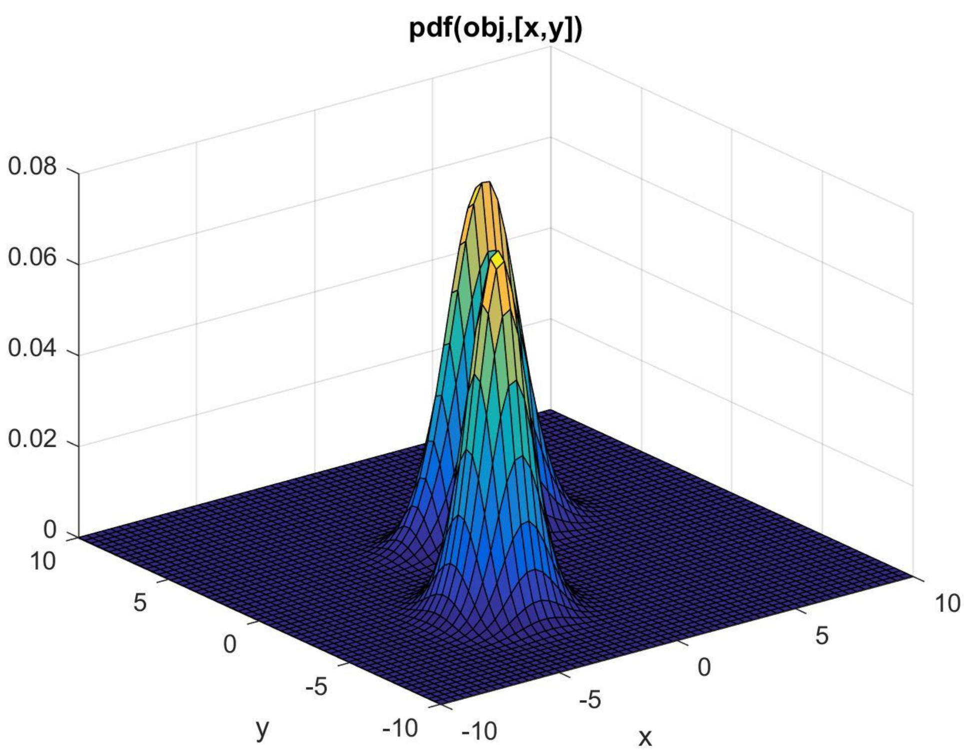
μ = [1 2;-3 -4], p = [0.25 0.75];



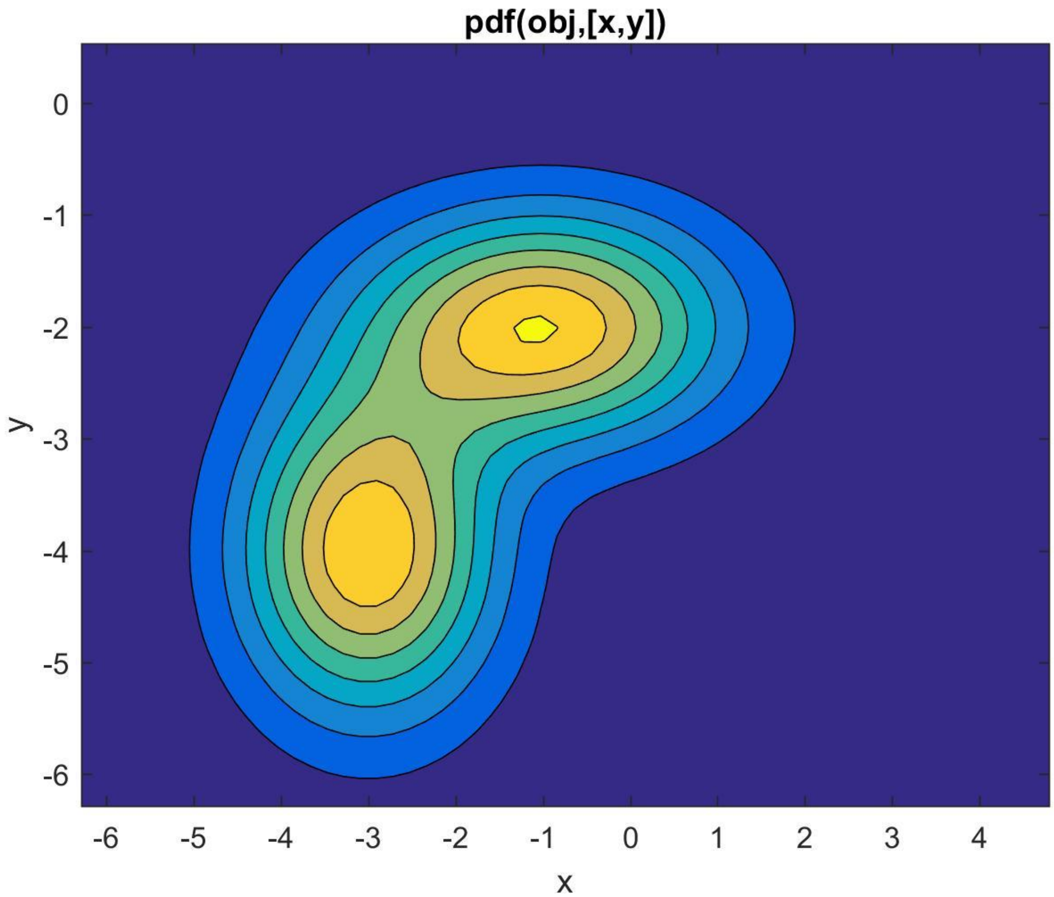


μ = [1 2;-3 -4], p = [0. 5 0. 5]


pdf(obj,[x,y])
4
3
2
1
0
-1
-2
-3
-4
-5
-6
y
-6
-4
-2
0
2
4
6
x


pdf(obj,[x,y])
0.08
0.06
0.04
0.02
0
10
5
0
-5
-10
y
-10
-5
0
5
10
x

μ = [-1 -2;-3 -4]



- **ANALYSIS:** As we can observe, changing the value of μ changes the proximity f the contour plots of the clusters to each other. Changing the values of the weights changes how heavily the distribution is effected by each weight. Changing the values of the ∑ changes the shapes of the contours.

Problem #4

A geyser is a hot spring characterized by an intermittent discharge of water and steam. Old Faithful is a famous cone geyser in Yellowstone National Park, Wyoming. It has a predictable geothermal discharge and since 2000 it has erupted every 44 to 125 minutes. Refer to the addendum data file that contains waiting times and the durations for 272 eruptions.

a. Generate a 2-D scatter plot of the data. Run a ⍰-means clustering routine on the data for k = 2. Show the two clusters on a scatterplot.

b. Use a GMM-EM algorithm to fit the dataset to a GMM pdf. Draw a contour plot of your final GMM pdf. Overlay the contour plot with a scatterplot of the data set. How can you use the GMM pdf estimates to cluster the data?

- **EXPERIMENT:** After gathering the data from the data file, I used the kmeans function to generate the clusters and then plot them together by using the hold function. For plotting the contour, I have again used the ezcontour() function and passed the distribution returned by the gmdistribution.fit() function.

- **CODE**

```
X = [            3.600      79;
                 1.800      54;
                 3.333      74;
                 2.283      62;
                 4.533      85;
                 2.883      55;
                 4.700      88;
                 3.600      85;
                 1.950      51;
                 4.350      85;
                 1.833      54;
                 3.917      84;
                 4.200      78;
                 1.750      47;
                 4.700      83;
                 2.167      52;
                 1.750      62;
                 4.800      84;
                 1.600      52;
                 4.250      79;
                 1.800      51;
                 1.750      47;
                 3.450      78;
                 3.067      69;
                 4.533      74;
                 3.600      83;
                 1.967      55;
                 4.083      76;
                 3.850      78;
                 4.433      79;
                 4.300      73;
                 4.467      77;
                 3.367      66;
                 4.033      80
                 3.833      74;
                 2.017      52;
                 1.867      48;
                 4.833      80;
                 1.833      59;
                 4.783      90;
                 4.350      80;
                 1.883      58;
                 4.567      84;
                 1.750      58;
                 4.533      73;
                 3.317      83;
                 3.833      64;
                 2.100      53;
                 4.633      82;
                 2.000      59;
                 4.800      75;
                 4.716      90;
                 1.833      54;
                 4.833      80;
                 1.733      54;
                 4.883      83;
                 3.717      71;
                 1.667      64;
                 4.567      77;
```

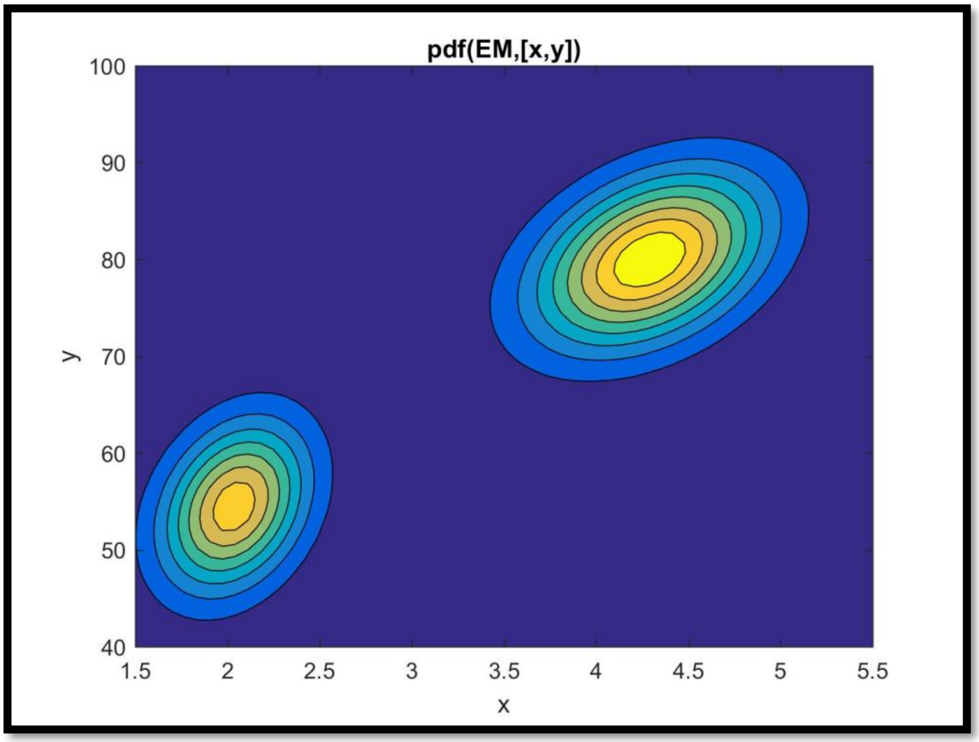
```
4.317      81;
2.233      59;
4.500      84;
1.750      48;
4.800      82;
1.817      60;
4.400      92;
4.167      78;
4.700      78;
2.067      65;
4.700      73;
4.033      82;
1.967      56;
4.500      79;
4.000      71;
1.983      62;
5.067      76;
2.017      60;
4.567      78;
3.883      76;
3.600      83;
4.133      75;
4.333      82;
4.100      70;
2.633      65;
4.067      73;
4.933      88;
3.950      76;
4.517      80;
2.167      48;
4.000      86;
2.200      60;
4.333      90;
1.867      50;
4.817      78;
1.833      63;
4.300      72;
4.667      84;
3.750      75;
1.867      51;
4.900      82;
2.483      62;
4.367      88;
2.100      49;
4.500      83;
4.050      81;
1.867      47;
4.700      84;
1.783      52;
4.850      86;
3.683      81;
4.733      75;
2.300      59;
4.900      89;
4.417      79;
1.700      59;
4.633      81;
2.317      50;
4.600      85;
1.817      59;
4.417      87;
2.617      53;
4.067      69;
```

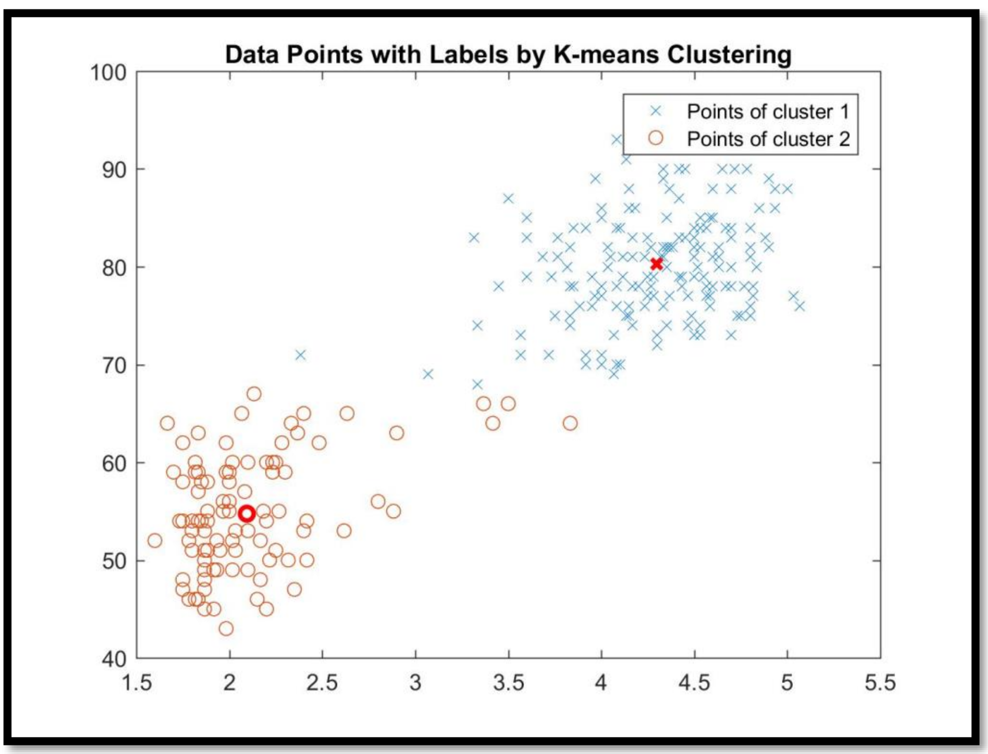
```
4.250       77;
1.967       56;
4.600       88;
3.767       81;
1.917       45;
4.500       82;
2.267       55;
4.650       90;
1.867       45;
4.167       83;
2.800       56;
4.333       89;
1.833       46;
4.383       82;
1.883       51;
4.933       86;
2.033       53;
3.733       79;
4.233       81;
2.233       60;
4.533       82;
4.817       77;
4.333       76;
1.983       59;
4.633       80;
2.017       49;
5.100       96;
1.800       53;
5.033       77;
4.000       77;
2.400       65;
4.600       81;
3.567       71;
4.000       70;
4.500       81;
4.083       93;
1.800       53;
3.967       89;
2.200       45;
4.150       86;
2.000       58;
3.833       78;
3.500       66;
4.583       76;
2.367       63;
5.000       88;
1.933       52;
4.617       93;
1.917       49;
2.083       57;
4.583       77;
3.333       68;
4.167       81;
4.333       81;
4.500       73;
2.417       50;
4.000       85;
4.167       74;
1.883       55;
4.583       77;
4.250       83;
3.767       83;
2.033       51;
```

4.433	78;
4.083	84;
1.833	46;
4.417	83;
2.183	55;
4.800	81;
1.833	57;
4.800	76;
4.100	84;
3.966	77;
4.233	81;
3.500	87;
4.366	77;
2.250	51;
4.667	78;
2.100	60;
4.350	82;
4.133	91;
1.867	53;
4.600	78;
1.783	46;
4.367	77;
3.850	84;
1.933	49;
4.500	83;
2.383	71;
4.700	80;
1.867	49;
3.833	75;
3.417	64;
4.233	76;
2.400	53;
4.800	94;
2.000	55;
4.150	76;
1.867	50;
4.267	82;
1.750	54;
4.483	75;
4.000	78;
4.117	79;
4.083	78;
4.267	78;
3.917	70;
4.550	79;
4.083	70;
2.417	54;
4.183	86;
2.217	50;
4.450	90;
1.883	54;
1.850	54;
4.283	77;
3.950	79;
2.333	64;
4.150	75;
2.350	47;
4.933	86;
2.900	63;
4.583	85;
3.833	82;
2.083	57;
4.367	82;

```
                    2.133       67;
                    4.350       74;
                    2.200       54;
                    4.450       83;
                    3.567       73;
                    4.500       73;
                    4.150       88;
                    3.817       80;
                    3.917       71;
                    4.450       83;
                    2.000       56;
                    4.283       79;
                    4.767       78;
                    4.533       84;
                    1.850       58;
                    4.250       83;
                    1.983       43;
                    2.250       60;
                    4.750       75;
                    4.117       81;
                    2.150       46;
                    4.417       90;
                    1.817       46;
                    4.467       74;];

%% kmeans and scatter plot
[y C] = kmeans(X,2); % Find the assignment y and the means C of
each cluster

figure(2)
plot(X(y==1,1),X(y==1,2), 'x');
hold on
plot(X(y==2,1),X(y==2,2), 'o');
% plot(X(y==3,1),X(y==3,2), '^');
% plot(X(y==4,1),X(y==4,2), '+');
plot(C(1,1),C(1,2), 'rx','LineWidth',2);
plot(C(2,1),C(2,2), 'ro','LineWidth',2);
% plot(C(3,1),C(3,2), 'r^','LineWidth',2);
% plot(C(4,1),C(4,2), 'r+','LineWidth',2);
legend('Points of cluster 1','Points of cluster 2')
title('Data Points with Labels by K-means Clustering')
hold off

%% GMM Distribution
EM = gmdistribution.fit(X,2);

%%  2D projection
figure;
% x = 1.5:5.5;
% y = 40:100;
ezcontourf(@(x,y) pdf(EM,[x y]),[1.5 5.5, 40 100]);
```

OUTPUT





- **ANALYSIS :** As can be seen in the plots, the clusters are well apart and the scatter plot is spherical.