EE511 Project 7

Zhang Fan

USC ID: 1417-68-5115

Contents

Question 1

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

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

Analysis:

Basic Theory:

Generate a multivariate normal random variable:

$$X_1 = a_{11}Z_1 + a_{12}Z_2 + \cdots + a_{1m}Z_{1m} + u_1$$
$$X_2 = a_{21}Z_1 + a_{22}Z_2 + \cdots + a_{2m}Z_{2m} + u_2$$
$$\cdots$$
$$X_n = a_{n1}Z_1 + a_{n2}Z_2 + \cdots + a_{nm}Z_{nm} + u_n$$

Therefore, $\mathrm{X} = \mathrm{AZ}' + \mathrm{u}'$, where $\mathrm{X} = \begin{bmatrix} X_1 \\ X_2 \\ \cdots \\ X_n \end{bmatrix}$; $\mathrm{A} = [\mathrm{a_{ik}}]$, where $\mathrm{i} = 1,2 \ldots \mathrm{n}$; $\mathrm{k} = 1,2 \ldots \mathrm{m}$; $\mathrm{Z} = [\mathrm{Z_1}, \mathrm{Z_2} .. \mathrm{Z_m}]$, $\mathrm{u} = [\mathrm{u_1} + \mathrm{u_2} \ldots \mathrm{u_n}]$. $\mathrm{Cov}(\mathrm{X_i}, \mathrm{X_j}) = \sum_{\mathrm{k}}^{\mathrm{m}} a_{ik} a_{jk} = [AA']$, where i,j=1,2,3…n

Our goal

Generate $\mathrm{X} = (\mathrm{X_1}, \mathrm{X_2}, \mathrm{X_3})$ and we have known $\sum = [AA']$ and u.

Algorithm

Firstly, generate three N~(0,1).

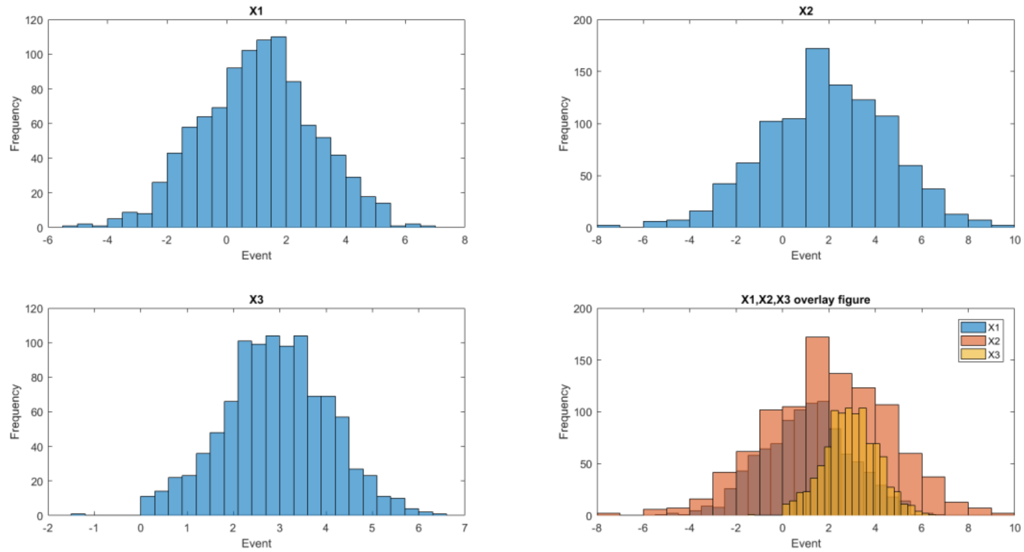
Secondly, apply Cholesky decomposition directly for $\sum$ $\quad and\ obtain\ A$.

Thirdly, According $\mathrm{X} = \mathrm{AZ}' + \mathrm{u}'$, we can easily generate X.

Code

```
%----------------------
clear
Z=0;
Xsum=0
for i=1:1000
z1=normrnd(0,1);
z2=normrnd(0,1);
z3=normrnd(0,1);
mu = [1,2,3];
sigma = [3,-1,1;-1,5,3;1,3,4];
A=chol(sigma);
X(1:3,i)=A*[z1,z2,z3]'+mu'
end
```

Result



Question 2

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.

Algorithm

Step1: generate N1~(1,1) and N2~(-1,1) and U~(0,1)

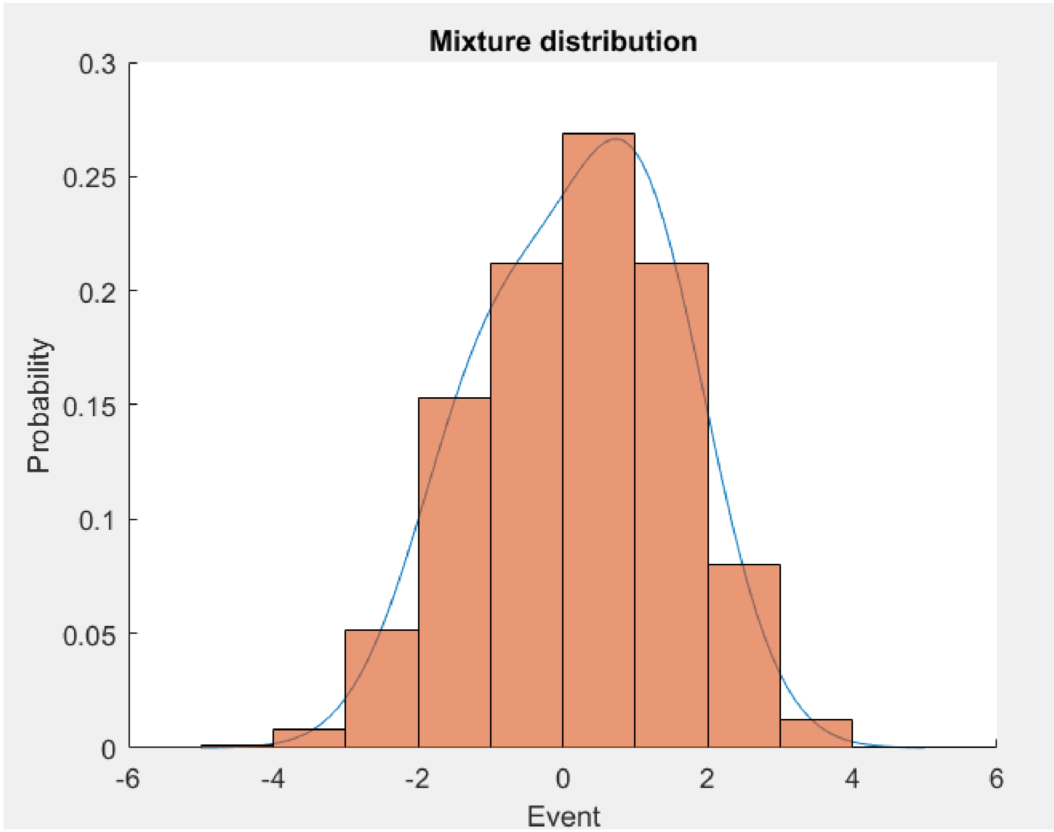
Step2: if U>0.4, then X=N2, else X=N1

Step3: repeat and generate histogram for X.

Code

```
clear
for i=1:10000
    x=rand;
    n1=normrnd(-1,1);
    n2=normrnd(1,1);
    if x>0.4    X(i)=n2;
    else X(i)=n1;
    end
end

hold on
norm_x=-5:0.1:5;
y=0.4*normpdf(norm_x,-1,1)+0.6*normpdf(norm_x,1,1);
plot(norm_x,y);
histogram(X,'Normalization','probability','Binwidth',1)
title('Mixture distribution')
xlabel('Event');
ylabel('Probability')
```

Result



Question 3

3. Implement a 2-dimensional random number generator for a Gaussian mixture model (GMM) pdf with 2 sub-populations. 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.).

Analysis:

In this question, we can apply the method used in question 1 to generate multivariate Gaussian distribution and the method in question 2 to generate mixture distribution. And then apply EM algorithm to estimate the pdf parameters.

To compare the spherical vs ellipsoidal covariance, the difference of the sigma at one of the component should be relatively large to make the distribution look like a spherical. To compare the close vs well-separated subpopulations, the expectation of two component should be tuned.

Main code

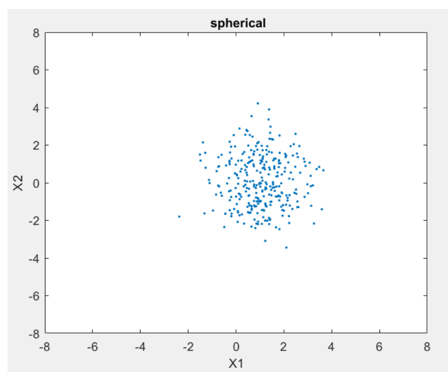
```
clear;
%Implement a random number generator for a random(the method used in Q1)
```

```
mu1 = [-3,3];
sigma1 = [1 0;0 1];
mu2 = [3,-3];
sigma2 = [1 0;0 1];
%mixture distribution, method in Q2
p=0.5;
for i=1:30000
x=rand;
if x>p
    r(i,:) = mvnrnd(mu1,sigma1,1);
else
    r(i,:) = mvnrnd(mu2,sigma2,1);
end
end
%EM to estimate the paremeter
tic;
GMModel = fitgmdist(r,2)
toc;
plot(r(:,1),r(:,2),'.');
title(' sphere')
xlabel('X1');
ylabel('X2')
axis([-8 8 -8 8])
```

Results

Close:

```
mu1 = [1,1];
sigma1 = [1 0;0 1];
mu2 = [1,-1];
sigma2 = [1 0;0 1];
```




```
Component 1:
Mixing proportion: 0.476988
Mean:    1.0281   -1.1723

Component 2:
Mixing proportion: 0.523012
Mean:    0.8343    1.0120


Elapsed time is 0.006257 seconds.
```

```
Component 1:
Mixing proportion: 0.364019
Mean:    1.0003   -1.1886

Component 2:
Mixing proportion: 0.635981
Mean:    1.0493    0.7982


Elapsed time is 0.009086 seconds.
```

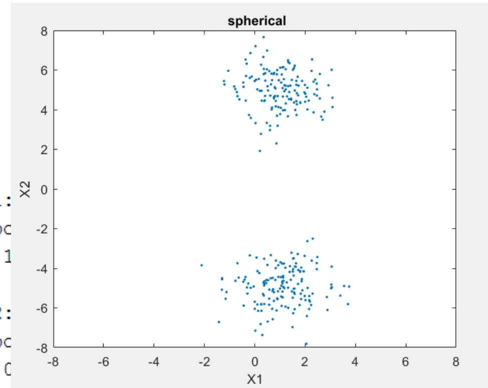
```
Component 1:
Mixing proportion: 0.633142
Mean:    0.9342    0.7436

Component 2:
Mixing proportion: 0.366858
Mean:    1.2528   -1.2811


Elapsed time is 0.012255 seconds.
```

well-separated:

```
mu1 = [1,5];
sigma1 = [1 0;0 1];
mu2 = [1,-5];
sigma2 = [1 0;0 1];
```

```
Component 1:
Mixing proportion: 0.432642
Mean:     0.6748     0.8879

Component 2:
Mixing proportion: 0.567358
Mean:     1.1933    -0.6715


Elapsed time is 0.027114 seconds.
```

```
Component 1:
Mixing propo
Mean:     1.1

Component 2:
Mixing propo
Mean:     1.0


Elapsed time is 0.025618 seconds.
```




```
on: 0.185671
   -1.0189

on: 0.814329
    0.1958


apsed time is 0.018996 seconds.
```

```
Gaussian mixture distribution with 2 componer
Component 1:
Mixing proportion: 0.486667
Mean:    -5.0037     5.0696

Component 2:
Mixing proportion: 0.513333
Mean:     4.8816    -5.0583


Elapsed time is 0.005713 seconds.
```

```
Gaussian mixture distribution with 2 comp
Component 1:
Mixing proportion: 0.483333
Mean:     5.1503    -5.0902

Component 2:
Mixing proportion: 0.516667
Mean:    -4.9977     5.0301


Elapsed time is 0.008544 seconds.
```

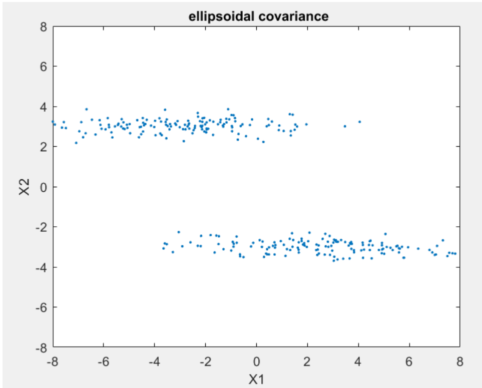
```
GMModel =

Gaussian mixture distribution with 2 components in 2 dimensions
Component 1:
Mixing proportion: 0.513333
Mean:    -5.0276     5.1650

Component 2:
Mixing proportion: 0.486667
Mean:     4.8270    -5.0118


Elapsed time is 0.011700 seconds.
```

```
Gaussian mixture distribution with 2 componer
Component 1:
Mixing proportion: 0.506667
Mean:     5.0279    -5.0423

Component 2:
Mixing proportion: 0.493333
Mean:    -5.0000     4.9041


Elapsed time is 0.003480 seconds.
```

```
Gaussian mixture distribution with
Component 1:
Mixing proportion: 0.450000
Mean:     5.0130    -4.8972

Component 2:
Mixing proportion: 0.550000
Mean:    -5.0603     5.0123


Elapsed time is 0.005697 seconds.
```

```
Gaussian mixture distribution with 2 components in 2
Component 1:
Mixing proportion: 0.500000
Mean:     4.9987    -5.0085

Component 2:
Mixing proportion: 0.500000
Mean:    -4.9711     5.0297


Elapsed time is 0.003828 seconds.
```

ellipsoidal covariance




```
%Implement a random number generator for a ranc
mu1 = [-3,3];
sigma1 = [10 0;0 0.1];
mu2 = [3,-3];
sigma2 = [10 0;0 0.1];
```

```
Gaussian mixture distribution with 2
Component 1:
Mixing proportion: 0.513333
Mean:    -3.3629    3.0218

Component 2:
Mixing proportion: 0.486667
Mean:     2.6443   -3.0070


Elapsed time is 0.015338 seconds.
```

```
Gaussian mixture distribution with 2
Component 1:
Mixing proportion: 0.523333
Mean:     3.1744   -2.9740

Component 2:
Mixing proportion: 0.476667
Mean:    -3.0295    3.0019


Elapsed time is 0.003712 seconds.
```

```
Gaussian mixture distribution with 2 components in 2
Component 1:
Mixing proportion: 0.470000
Mean:     2.8085   -2.9884

Component 2:
Mixing proportion: 0.530000
Mean:    -3.3184    3.0745


Elapsed time is 0.003495 seconds.
```

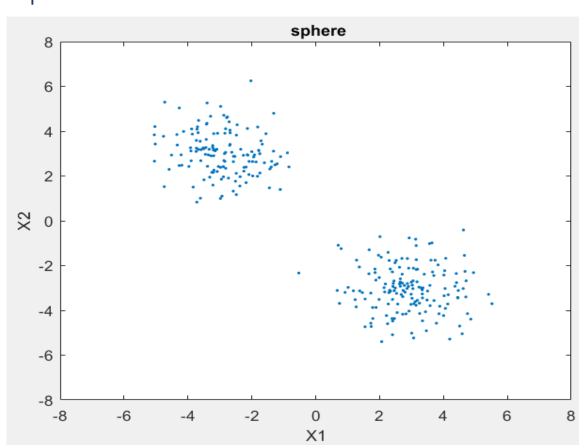
```
Gaussian mixture distribution with 2 c
Component 1:
Mixing proportion: 0.516667
Mean:     2.6706   -3.0030

Component 2:
Mixing proportion: 0.483333
Mean:    -3.2327    2.9900


Elapsed time is 0.004153 seconds.
``
```

```
Gaussian mixture distribution w
Component 1:
Mixing proportion: 0.523333
Mean:    -2.6908    2.9680

Component 2:
Mixing proportion: 0.476667
Mean:     2.9547   -3.0116


Elapsed time is 0.005421 seconc
```

```
Gaussian mixture distribution with 2 com
Component 1:
Mixing proportion: 0.510000
Mean:    -3.1924    3.0073

Component 2:
Mixing proportion: 0.490000
Mean:     3.6770   -2.9654


Elapsed time is 0.004374 seconds.
```

sphere

```
mu1 = [-3,3];
sigma1 = [1 0;0 1];
mu2 = [3,-3];
sigma2 = [1 0;0 1];
```



```
Gaussian mixture distributic
Component 1:
Mixing proportion: 0.470000
Mean:    -2.9194    3.0190

Component 2:
Mixing proportion: 0.530000
Mean:     2.9072   -3.0446
```

```
Gaussian mixture distributio
Component 1:
Mixing proportion: 0.523334
Mean:    -2.8992    2.8746

Component 2:
Mixing proportion: 0.476666
Mean:     2.9779   -2.9971


Elapsed time is 0.008261 sec
```

```
Gaussian mixture distribution with 2 components
Component 1:
Mixing proportion: 0.460010
Mean:    -2.9789    2.8616

Component 2:
Mixing proportion: 0.539990
Mean:     3.0875   -2.9746


Elapsed time is 0.004610 seconds
```

```
Gaussian mixture distribution
Component 1:
Mixing proportion: 0.546667
Mean:    -2.9055    2.9921

Component 2:
Mixing proportion: 0.453333
Mean:     2.9437   -2.9979


Elapsed time is 0.004808 seco
```

```
Gaussian mixture distributio
Component 1:
Mixing proportion: 0.524804
Mean:    -0.3115    0.1949

Component 2:
Mixing proportion: 0.475196
Mean:     0.0645   -0.1864


Elapsed time is 0.003428 sec
```

```
Gaussian mixture distribution with 2 compon
Component 1:
Mixing proportion: 0.513333
Mean:     2.9859   -2.9676

Component 2:
Mixing proportion: 0.486667
Mean:    -3.0091    2.8976


Elapsed time is 0.013306 seconds.
```

Comment

From above results(Actually lots of trials have been implemented) we can see that the closer the two components are, the higher quality and faster the algorithm is.

And the closer that the distribution to sphere, the higher quality and faster of the GMM algorithm for estimating the parameters of pdf. However, it seems that the speed of the GMM algorithm is not highly influenced by the shape of distribution.

Question 4

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 k-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?

Analysis:

In this question, we mainly apply K-means cluster method and GMM-EM algorithm to fit the dataset. For K-means algorithm, Clustering is a famous unsupervised learning problem with the following setup:

Given the n-dimensional data points $\{x_i\}_{n i=1}$ and predetermined number of clusters k, we want to find an assignment for every data points, i.e.

$$y_i=A(x_i)\in\{1,2,\cdots,k\},\forall i$$

k-means algorithm aims to partition the data points $\{x_i\}_{n i=1}$ with centers (means) of each cluster $c_j,j\in\{1,2,\cdots,k\}$. The main idea is to run the 2-step procedure iteratively: First, assign a point x_i to the cluster with the closest center c_j via $y_i=\mathrm{argmin}_j\|x_i-c_j\|2$. Then, after assigning all the points with a cluster index $y_i,\forall i$, the algorithm updates the center of each cluster by calculating the mean i.e.

$$c_j=1 n_j\sum i\in\{i|y_i=j\}x_i,$$

where n_j denotes the number of data points in cluster j. The algorithm runs the assign-and-recalculate-mean steps repeatly until the the assignment remains unchanged.

k-means algorithm is a heuristic way to minimize the squared error objective

$$J=\sum k j=1\sum i\in\{i|y_i=j\}\|x_i-c_j\|2,$$

which can be also viewed as the sum of squared distances from data point x_i to the mean of its cluster c_j.

(Reference: https://piazza.com/class/ixp26i0vk6s5dg?cid=64).

For GMM model, the algorithm is as the following shows:

Step 1: find the complete-data likelihood function for multivariate normal distribution.

Step 2: do E step.

Step 3: do M step.

Step 4: convergence check and terminate.

Code for a:

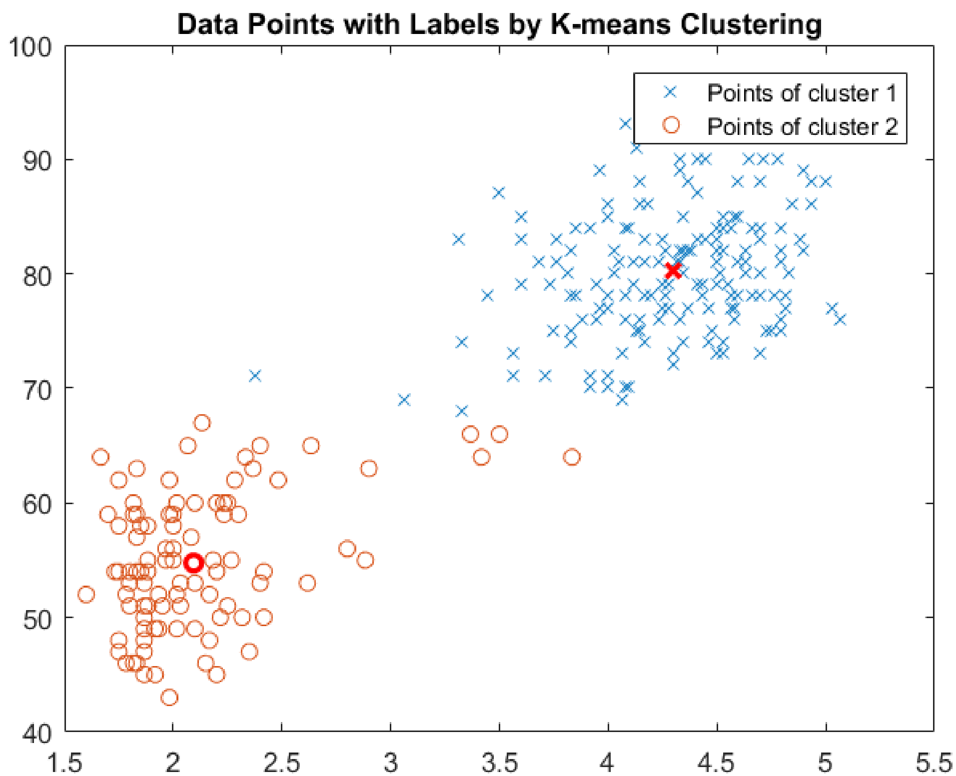
```
%% Generate data for kmeans
clear
X= load('data.txt');
figure(1)
X=X(:,2:3);
plot(X(:,1), X(:,2), 'o');
title('Data Points without Labels')

%% 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(C(1,1),C(1,2), 'rx','LineWidth',2);
plot(C(2,1),C(2,2), 'ro','LineWidth',2);

legend('Points of cluster 1','Points of cluster 2')
title('Data Points with Labels by K-means Clustering')
hold off
```

Code for b

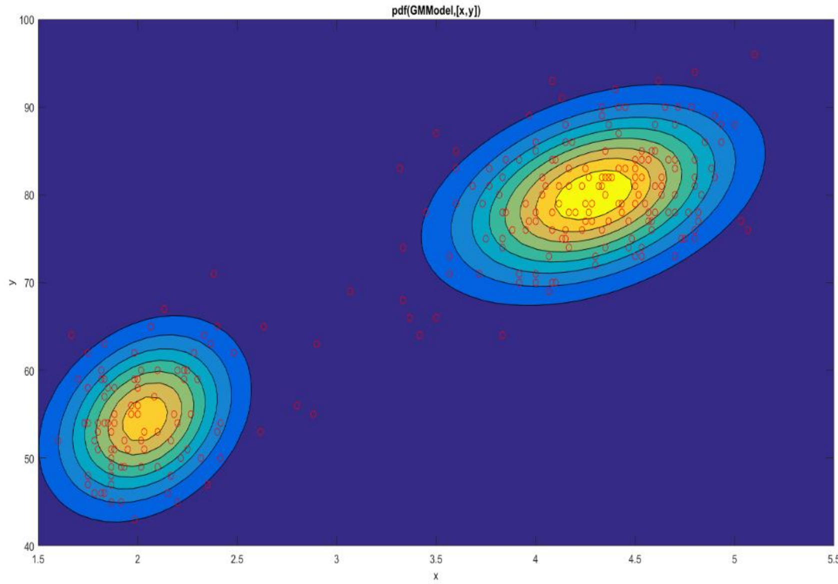
```
%% Generate data for kmeans
clear
X= load('data.txt');
X=X(:,2:3);
title('Data Points without Labels')
GMModel = fitgmdist(X,2);
```

```
% 2D projection
ezcontourf(@(x,y) pdf(GMModel,[x y]),[1.5,5.5],[40,100]);
hold on
plot(X(:,1), X(:,2), 'or');
```

Result for a:



Result for b:



The yellow zone from the above figure means it has higher probability that the data belongs to correspond component. Therefore, we can just set a threshold of probability for the data: if the probability of the data that belongs to one component is higher than this threshold, then accept it. Or reject it.

Comment

K-means method and GMM method both can be used for clustering. But there are still a little difference between them. For example, GMM method is according to the probability while K-means is according to distance between.