Project #7 - Expectation Maximization

Li Yicheng*

USCID:7827077047

email: l.y.c.liyicheng@gmail.com

USC Viterbi of Engineering

I. Problem description

This project details the use of EM for one popular problems of learning a mixture. We consider the problem of learning a GMM (also called EM clustering), where both the mixture weights of the Gaussians and the parameters for each component Gaussian must be learned. We illustrate using EM to learn a GMM when there are additional constraints on the GMM parameters. In section II, we talk about generating samples from a multivariate Gaussian. In section III, we discuss generate a mixture model of multivariate Gaussian. In section III, we will get into how to perform the GMM-EM estimation to learn a GMM model with 2 dimensions and 2 subpopulations. In section IV, we use the practical data from reality and perform the EM estimation to get insight of the feature of the data.

II. Multivariate Gaussian

I. Definition

I.1 Bivariate Case

Suppose we have two random variable, N_1, N_2, $N_1 \sim (\mu_1, \sigma_1^2)$ and $N_2 \sim (\mu_2, \sigma_2^2)$. Consider a plane with $point(x, y)$ on it. If we take one sample from N_1 and take the value as the value of x and take one sample from N_2 and take the value as y. After many sampling action, for one small area on the plane, mark as (x, y) we count the number of samples, and we may get the joint probability of $f_{N_1,N_2}(N_1 = x, N_2 = y)$.
In this case

$$f(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} \exp\left(-\frac{1}{2(1-\rho^2)}\left[\frac{(x-\mu_X)^2}{\sigma_X^2} + \frac{(y-\mu_Y)^2}{\sigma_Y^2} - \frac{2\rho(x-\mu_X)(y-\mu_Y)}{\sigma_X\sigma_Y}\right]\right)$$

where ρ is the correlation between X and Y and where $\sigma_X > 0$ and $\sigma_Y > 0$. In this case, $\boldsymbol{\mu} = \begin{pmatrix}\mu_X \\ \mu_Y\end{pmatrix}$, $\boldsymbol{\Sigma} = \begin{pmatrix}\sigma_X^2 & \rho\sigma_X\sigma_Y \\ \rho\sigma_X\sigma_Y & \sigma_Y^2\end{pmatrix}$. And we can expand this model to higher dimention.

*github link: https://github.com/IAMLYCHEE/EE501-PROJ7

I.2 High Dimensional Case

In this case the distribution has density

$$f_{\mathbf{x}}(x_1,\ldots,x_k) = \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{(2\pi)^k|\boldsymbol{\Sigma}|}}$$
$$= \frac{\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)}{\sqrt{|2\pi\boldsymbol{\Sigma}|}},$$

where $\mathbf{x}$ is a real k-dimensional column vector and $|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$.

II. Construction and Sampling

let $Z_1, Z_2, \ldots, Z_n$ be i.i.d standard normal distribution, that is $Z_i \sim N(0,1)$. For constant $a_{i,j}$ and μ_i, define:

$X_1 = a_{11}Z_1 + a_{12}Z_2 + \cdots a_{1m}Z_m + \mu_1$

$X_2 = a_{21}Z_1 + a_{22}Z_2 + \cdots a_{2m}Z_m + \mu_2$

$\vdots$

$X_n = a_{n1}Z_1 + a_{n2}Z_2 + \cdots a_{nm}Z_m + \mu_n$

The vector $X = [X_1, X_2, \cdots, X_n] is called multivariate normal$

In matrix mode, this can be wirtten as:

$$\begin{pmatrix} X_1 \\ X_2 \\ \vdots \\ X_n \end{pmatrix} = \begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,m} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n,1} & a_{n,2} & \cdots & a_{n,m} \end{pmatrix} * \begin{pmatrix} Z_1 \\ Z_2 \\ \vdots \\ Z_m \end{pmatrix} + \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix}$$

And it is easy to get that the covariance $\Sigma = A * A^T$ where A is the matrix formed by $a_{i,j}$. Therefore, if we were given μ which provides the information about the mean value of each Gaussian components and Σ the covariance between each two of the Gaussian components. We can decompose the Σ into matrix A and generate samples from standard Gaussian random variables, then we can get several samples from Gaussian multivariate.

II.1 Implement

According to the last discussion the code is straightforward:

filename: multiGauGenerator

```
function sample = multiGauGenerator(cov,mu,sampleAmount)
%use:  sample = multiGauGenerator(cov,mu,sampleAmount)
%generate multivariate gaussian random samples with given mean and
%covariance.
%Li Yicheng

%reshape the input
```

```
mu = reshape(mu,[length(mu),1]);
a = chol(cov,'lower');
% a * a';
aSize = size(cov);
sample = zeros(sampleAmount,aSize(1));
for i = 1 : sampleAmount
    x = a * randn(aSize(2),1) + mu;
    sample(i,:) = x';
end
```

II.2 Experiment

give : $\mu = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, \Sigma = \begin{pmatrix} 3 & -1 & 1 \\ -1 & 5 & 3 \\ 1 & 3 & 4 \end{pmatrix}$

we generate samples from this gaussian multivariate:

filename: solution1.m

```
clear
sampleAmount = 300;
cov = [3,-1,1;-1,5,3;1,3,4];
mu = [1;2;3];
sample = multiGauGenerator(cov,mu,sampleAmount);
scatter3(sample(:,1),sample(:,2),sample(:,3))
title('sample distribution')
```

Result:

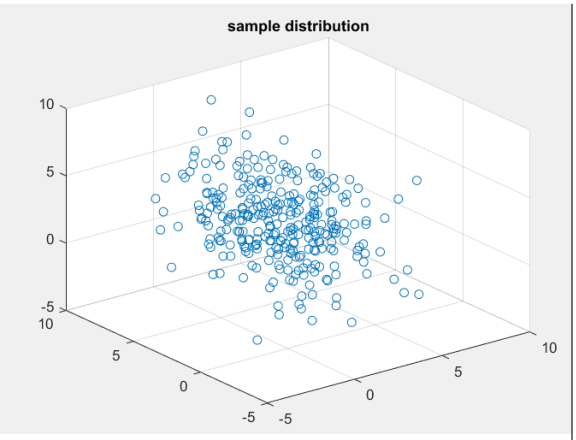

Figure 1: *samples in a 3D view*

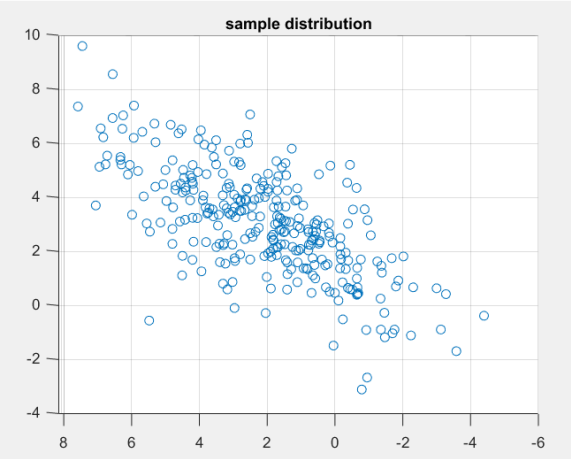

Figure 2: *samples view from* x_1

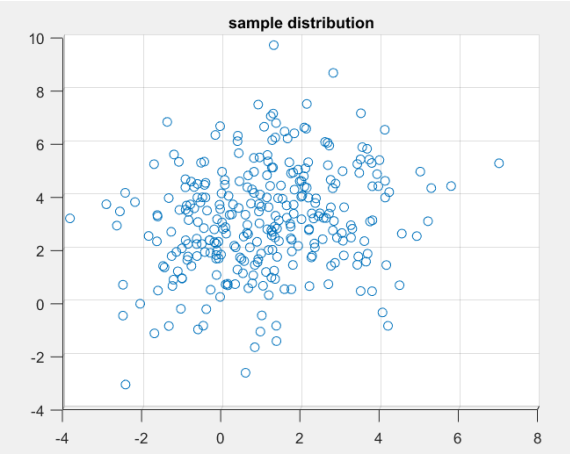

Figure 3: *samples view from* x_2

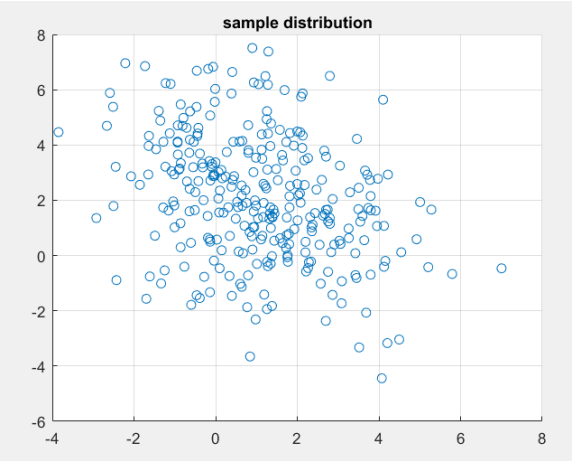

Figure 4: *samples view from* x_3

III. Mixture Model

I. Operation

If we already have several multivariate Gaussian , we mixture them by given each of the multivariate a weight.
When sampling we can first generate a sample from standard uniform distribution, and decide which Gaussian multivariate generator should we choose and then generate samples from that generator.

II. Experiment

In this experiment, we implement the following mixture distribution: $f(x) = 0.4N(-1,1) + 0.6N(1,1)$, according to the above discussion. the code is straightforward:
filename: solution2.m

```
sampleAmount = 100000;
sample = zeros(sampleAmount,1);
for i = 1 : sampleAmount
    if rand(1) < 0.4
        sample(i) = randn(1) - 1;
    else
        sample(i) = randn(1) + 1;
    end
end
histogram(sample,'BinLimits',[-4,4],'Normalization','probability')
hold on
```

```
%theoretical
i = 1;
for t = -4: 0.05 : 4
    c = sqrt(2 * pi);
    p1 = exp(-((t+1)^2)/2);
    p2 = exp(-((t-1)^2)/2);
    ft(i) = 0.4 / c * p1 + 0.6 / c * p2;
    i = i+1;
end

plot(-4 : 0.05 : 4, ft/9);
legend('histogram','theoretical')
title('mixture gaussian(100000 sample)')
```

result:

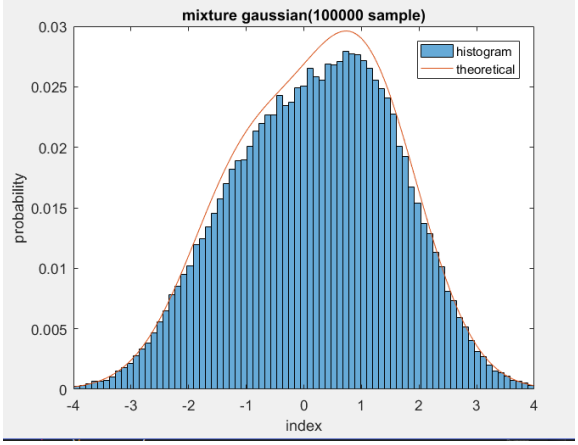

Figure 5: *mixture of 2 1d GM*

IV. GMM-EM estimation

I. Algorithm Overview

1.Initialization

Choose initial estimates $\omega_j^{(0)}, \mu_j^{(0)}, \Sigma_j^{(0)}, j = 1 \cdots k$, where j is the index of the cluster. k is the number of subpopulation.

Compute the initial log-likelihood:

$l^{(0)} = \frac{1}{n} \sum_{i=1}^{n} log(\sum_{j=1}^{k} \omega_j^{(0)} \phi(y_i|\mu_j^{(0)}, \Sigma_j^{(0)}))$

2.E-step

For $j = 1, \cdots, k$ compute

$$\gamma_{ij}^{(m)} = \frac{\omega_j^{(m)} \phi(y_i|\mu_j^{(m)}, \Sigma_j^{(m)})}{\sum_{l=1}^{k} \omega_l^{(m)} \phi(y_i|\mu_l^{(m)}, \Sigma_l^{(m)})}$$

and

$$n_j^{(m)} = \sum_{i=1}^{n} \gamma_{ij}^{(m)}$$

3.M-step

For $j = 1, \cdots, k$ compute the new estimates

$\omega_j^{(m+1)} = \frac{n_j^{(m)}}{n}$

$\mu_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} y_i$

$\Sigma_j^{(m+1)} = \frac{1}{n_j^{(m)}} \sum_{i=1}^{n} \gamma_{ij}^{(m)} (y_i - \mu_j^{(m+1)})(y_i - \mu_j^{(m+1)})^T$

4.Convergence Check: Compute the new likelihood:

$l^{(m+1)} = \frac{1}{n} \sum_{i=1}^{n} log(\sum_{j=1}^{k} \omega_j^{(m+1)} \phi(y_i|\mu_j^{(m+1)}, \Sigma_j^{(m+1)}))$

Return to 2 step if $|l^{(m+1)} - l^{(m)}| > \delta$ for some threshold δ.

II. Analysis

II.1 Initialization

First we initialize by first doing a cheaper clustering with k-means algorithm and seperate the samples into two subpopulation. Then we can derive the center of the two subpopution and take them as first estimation of μ,set the estimation of the covariance to be identity matrices, and the first guess at the weights $\omega_1 = \omega_2 = \cdots = \omega_k$. After we have these parameters, we can calculate the log-likelihood. Using the density function of the multivariate

Initialize (part of GMM2d2popuEMcore.m)

```
k = 2;
w = [1/k,1/k];
[y,C] = kmeans(sample,k);
mu1 = C(1,:);
mu2 = C(2,:);
mu = zeros(2,1,k);
mu(:,:,1) = mu1';
mu(:,:,2) = mu2';
covES = zeros(2,2,k);
covES(:,:,1) = eye(2);
covES(:,:,2) = eye(2);
sumLH = 0;
for i = 1 : sampleAmount
    sumLH = sumLH + log(w(1) * mvnpdf(sample(i,:),mu(:,:,1)',covES(:,:,1)) + ...
```

```
15                  w(2) * mvnpdf(sample(i,:),mu(:,:,2)',covES(:,:,2)));
16 end
17 L = sumLH/sampleAmount;
```

II.2 E-step

Actually the fraction of $\gamma_{i,j}$ tells us what probability $sample[i]$ would appear when its a sample from $GaussianMultivariate[j]$ and the nominator gives the total probabilty.
For example, if $sample(i)$ have biggist probability being from subpopulation 1, then $\gamma_{i,1} > \gamma_{i,2}$(if the number of subpopulation is 2).
After get the $\gamma_{i,j}$, $i = 1 \cdots sampleAmount, j = 1 \cdots k$, we simply add the γ matrix by column and we can get the weight in each subpopulation.

E-step (part of GMM2d2popuEMcore.m)

```
1  gamma = zeros(sampleAmount,k);
2  n = zeros(1,2);
3  while abs(L_new-L) >  ebsilon
4  %E-step
5      L = L_new;
6      for j = 1 : k
7          for i = 1 : sampleAmount
8              sumLHPoint = 0;
9              for l = 1 : k
10                 sumLHPoint = sumLHPoint + w(l) * mvnpdf(sample(i,:),mu(:,:,l)',covES(:,:,
                       l));
11             end
12             gamma(i,j)= w(j) * mvnpdf(sample(i,:),mu(:,:,j)',covES(:,:,j))/sumLHPoint;
13         end
14     end
15     n(1) = sum(gamma(:,1));
16     n(2) = sum(gamma(:,2));
```

II.3 M-step

Using the γ, we can get the new mean and covariance, because γ provides information about each samples's probability of belonging to some subpopulation.
Just use that information to update mean and covariance.

M-step (part of GMM2d2popuEMcore.m)

```
1      for j = 1 : k
2          mu(:,:,j) = (gamma(:,j)' * sample /n(j) )';
3          sumCov = zeros(2,2);
4          for i = 1 : sampleAmount
5              sumCov  = sumCov + gamma(i,j) * (sample(i,:)' - mu(:,:,j))*...
6                          (sample(i,:)' - mu(:,:,j))';
7          end
```

```
        covES(:,:,j)=sumCov / n(j);
    end
```

II.4 Convergence check (part of GMM2d2popuEMcore.m)

```
    sumLH = 0;
    for i = 1 : sampleAmount
        sumLH = sumLH + log(w(1) * mvnpdf(sample(i,:),mu(:,:,1)',covES(:,:,1)) + ...
                w(2) * mvnpdf(sample(i,:),mu(:,:,2)',covES(:,:,2)));
    end
    L_new = sumLH/sampleAmount;
```

The above implements the core algorithm of GMM-EM.

III. Experiment

Run the following script to implement the experiment:

Experiment 1: no covariance

$$\omega = \begin{pmatrix}0.5\\0.5\end{pmatrix} \mu_1 = \begin{pmatrix}2\\5\end{pmatrix} \mu_2 = \begin{pmatrix}0\\1\end{pmatrix} \Sigma_1 = \begin{pmatrix}3 & 0\\0 & 0.5\end{pmatrix} \Sigma_2 = \begin{pmatrix}1 & 0\\0 & 2\end{pmatrix}$$

Experiment 2: with covariance

$$\omega = \begin{pmatrix}0.5\\0.5\end{pmatrix} \mu_1 = \begin{pmatrix}2\\5\end{pmatrix} \mu_2 = \begin{pmatrix}0\\1\end{pmatrix} \Sigma_1 = \begin{pmatrix}3 & 1\\1 & 0.5\end{pmatrix} \Sigma_2 = \begin{pmatrix}1 & 0.5\\0.5 & 2\end{pmatrix}$$

Experiment 3: different weight

$$\omega = \begin{pmatrix}0.2\\0.8\end{pmatrix} \mu_1 = \begin{pmatrix}2\\5\end{pmatrix} \mu_2 = \begin{pmatrix}0\\1\end{pmatrix} \Sigma_1 = \begin{pmatrix}3 & 1\\1 & 0.5\end{pmatrix} \Sigma_2 = \begin{pmatrix}1 & 0.5\\0.5 & 2\end{pmatrix}$$

Experiment 4: close

$$\omega = \begin{pmatrix}0.5\\0.5\end{pmatrix} \mu_1 = \begin{pmatrix}2\\3\end{pmatrix} \mu_2 = \begin{pmatrix}3\\2\end{pmatrix} \Sigma_1 = \begin{pmatrix}3 & 1\\1 & 0.5\end{pmatrix} \Sigma_2 = \begin{pmatrix}1 & 0.5\\0.5 & 2\end{pmatrix}$$

```
clear
close all

clear
%generate samples
%test1 : same weight well seperated spherical
%generate the first population
w = [0.5,0.5];
mu1 = [2;5];
mu2 = [0;1];
cov1 = [3,0;0,1/2];
cov2 = [1,0;0,2];
GMM2d2popuEM(mu1,mu2,cov1,cov2,w,300)
```

```
14
15 clear
16 %test2 : ellipsoidal covariance
17 w = [0.5,0.5];
18 mu1 = [2;5];
19 mu2 = [0;1];
20 cov1 = [3,1;1,1/2];
21 cov2 = [1,1/2;1/2,2];
22 GMM2d2popuEM(mu1,mu2,cov1,cov2,w,300)
23
24 clear
25 %test3 : different weight
26 w = [0.2,0.8];
27 mu1 = [2;5];
28 mu2 = [0;1];
29 cov1 = [3,1;1,1/2];
30 cov2 = [1,1/2;1/2,2];
31 GMM2d2popuEM(mu1,mu2,cov1,cov2,w,300)
32
33 clear
34 %test4 : close
35 w = [0.5,0.5];
36 mu1 = [2;3];
37 mu2 = [3;2];
38 cov1 = [3,1;1,1/2];
39 cov2 = [1,1/2;1/2,2];
40 GMM2d2popuEM(mu1,mu2,cov1,cov2,w,300)
```

the next page Use the second experiment to show the EM process , for the rest of the experiments,data would be provided to show the speed and equality of the algorithm

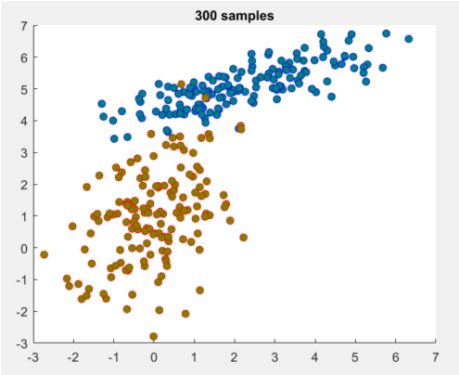

Figure 6: *Samples*

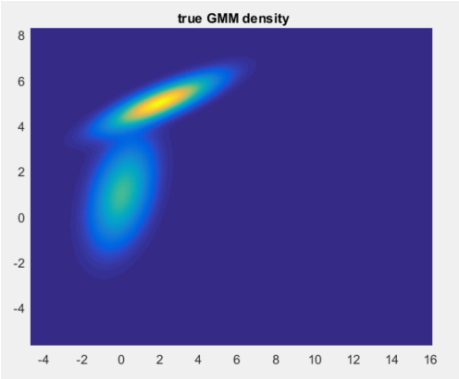

Figure 7: *The true expected GMM*

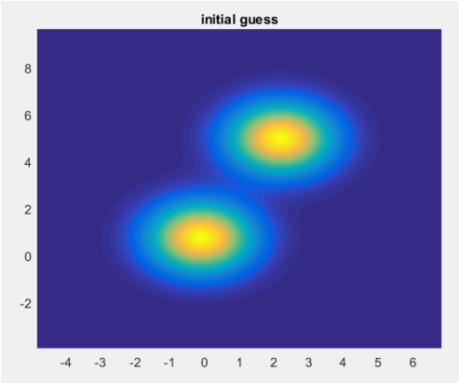

Figure 8: *Initial guess*

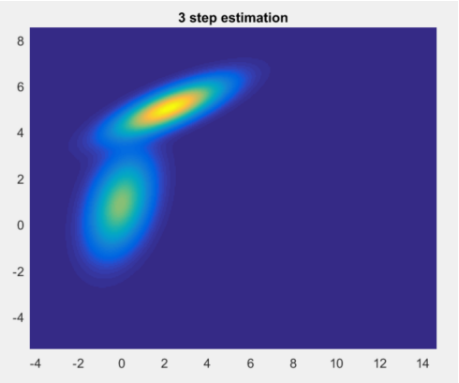

Figure 9: *the 3rd step EM*

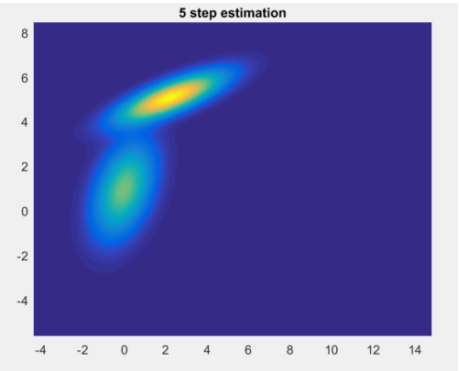

Figure 10: *the 5th step EM*

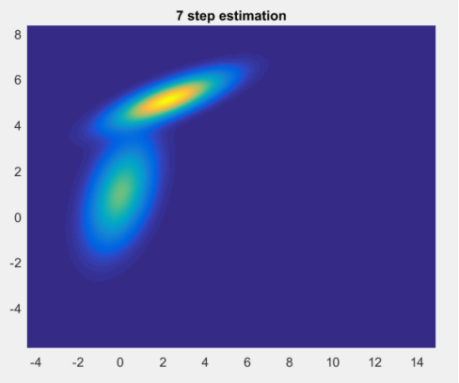

Figure 11: *the 7th step EM*

result:

Experiment 1: true parameter

$\omega = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$ $\mu_1 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$ $\mu_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ $\Sigma_1 = \begin{pmatrix} 3 & 0 \\ 0 & 0.5 \end{pmatrix}$ $\Sigma_2 = \begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$

EM estimation

step:5

mu(:,:,1) =

0.0599 1.0909

mu(:,:,2) =

2.2225 5.0133

covES(:,:,1) =

0.9141 -0.0927 -0.0927 1.9269

covES(:,:,2) =

2.9422 -0.0392 -0.0392 0.4740

w =

0.4995 0.5005

Experiment 2: true parameter

$\omega = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$ $\mu_1 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$ $\mu_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ $\Sigma_1 = \begin{pmatrix} 3 & 1 \\ 1 & 0.5 \end{pmatrix}$ $\Sigma_2 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}$

EM estimation

step:5

mu(:,:,1) =

2.0323 4.9962

mu(:,:,2) =

0.0189 1.1414

covES(:,:,1) =

3.0709 1.0645 1.0645 0.5731

covES(:,:,2) =

1.0111 0.5107 0.5107 1.9100

w =

0.4999 0.5001

Experiment 3: true parameter

$\omega = \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix}$ $\mu_1 = \begin{pmatrix} 2 \\ 5 \end{pmatrix}$ $\mu_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$ $\Sigma_1 = \begin{pmatrix} 3 & 1 \\ 1 & 0.5 \end{pmatrix}$ $\Sigma_2 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}$

result:

step:12

mu(:,:,1) =

2.0810 5.0054

mu(:,:,2) =

0.1008 1.0737

covES(:,:,1) =

1.6086 0.4771 0.4771 0.3798

covES(:,:,2) =

0.9722 0.4338 0.4338 1.8664

w =

0.2133 0.7867

Experiment 4: true parameter

$\omega = \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$ $\mu_1 = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ $\mu_2 = \begin{pmatrix} 3 \\ 2 \end{pmatrix}$ $\Sigma_1 = \begin{pmatrix} 3 & 1 \\ 1 & 0.5 \end{pmatrix}$ $\Sigma_2 = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 2 \end{pmatrix}$

result: step:25

mu(:,:,1) =

3.0958 2.0454

mu(:,:,2) =

1.9381 2.9674

covES(:,:,1) =

0.9031 0.6542 0.6542 2.1917

covES(:,:,2) =

2.5967 0.9282 0.9282 0.5100

w =

0.4805 0.5195

Performance Analysis: The weight actually has not very much influence on the speed although if the weight vary a lot from our initial guess, it may take more iterations. However, if the two Gaussian Multivariate are close, it really took a lot more iterations. And it can be seem from the data. If the threshold is set to 0.0001, the variance between the true mean and the estimation is actually lower than 5% . The covariance may vary a bit between the true parameter and the estimation, that basicly is due to the amount of samples are really small and if large amount of samples are given we may get better estimation.

V. Practical Data EM

I. Problem

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 predicable geothermal discharge and since 2000, it has errupted every 44 to 125 minutes. Refer to the addendum data file that contains waiting times and the durations for 272 erruptions.

II. Estimation using EM

simply run the following script

```
clear
clc
close all
duration = load('duration.mat');
duration = duration.duration;
waiting = load('waiting.mat');
waiting = waiting.waiting;
scatter(duration,waiting);
title('without kmeans clustering')
xlabel('duration')
ylabel('waiting')


%cascade data points
samples = [duration,waiting];
%shuffle
samples = samples(randperm(size(samples,1)),:);
%kmeans
[y,C] = kmeans(samples,2);

%figure the clusters
figure
hold on
```

```
scatter(samples(y == 1,1),samples(y==1,2),'x')
scatter(samples(y == 2,1),samples(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')
xlabel('duration')
ylabel('waiting')
hold off

figure
GMM2d2popuEMcore(samples,0)
hold on
scatter3(samples(y == 1,1),samples(y==1,2),ones(length(samples(y==1,1)),1),'rx')
scatter3(samples(y == 2,1),samples(y==2,2),ones(length(samples(y==2,1)),1),'ro')
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 & EM GMM estimation')
xlabel('duration')
ylabel('waiting')
hold off
```

Using the EM implemented in last topic

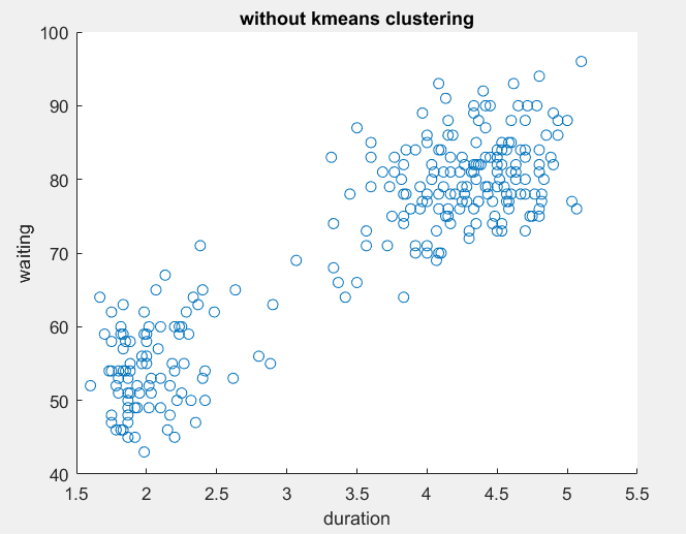

Figure 12: *samples from data*

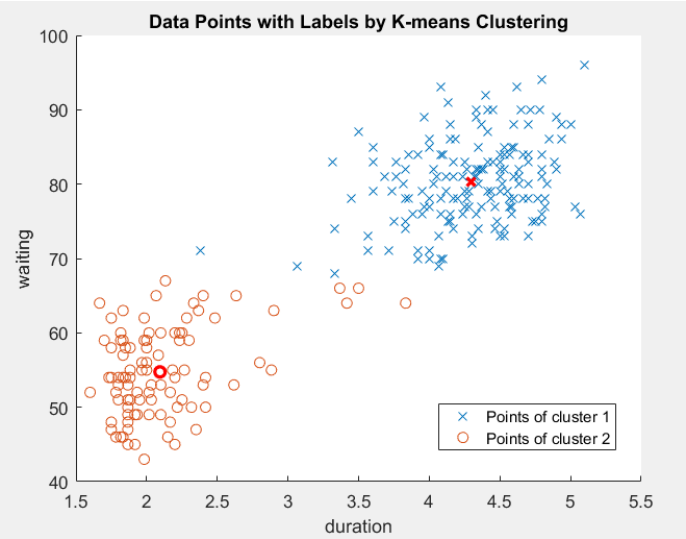

Figure 13: *roughly clusering using K-mean*

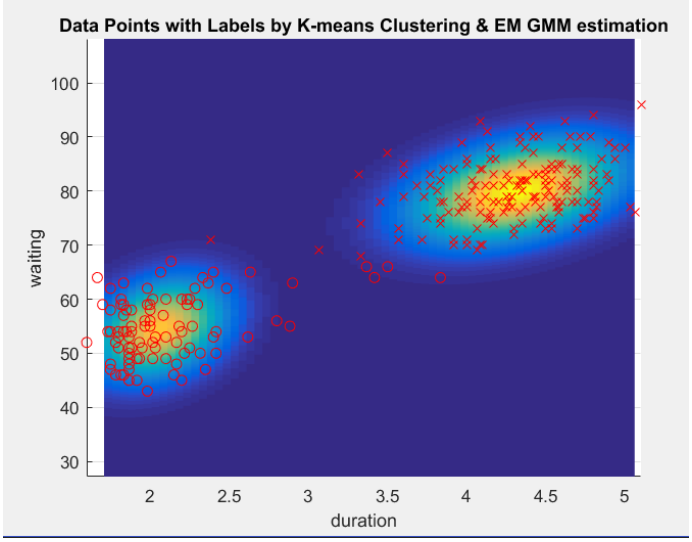

Figure 14: *EM result*

data obtained

mu(:,:,1) =

4.2902 79.9740

mu(:,:,2) =

2.0370 54.4843

covES(:,:,1) =

0.1693 0.9328 0.9328 35.9590

covES(:,:,2) =

0.0696 0.4400 0.4400 33.7325

Therefore, we can use the GMM model to estimate the behaviour of the geyser by sampling from the GMM model with parameters derived from above.

VI. Appendix

filename:GMM2d2popuEMcore.m

```
function GMM2d2popuEMcore(sample,option)
sampleAmount = size(sample,1);
k = 2;
w = [1/k,1/k];
[y,C] = kmeans(sample,k);
mu1 = C(1,:);
mu2 = C(2,:);
mu = zeros(2,1,k);
mu(:,:,1) = mu1';
mu(:,:,2) = mu2';
covES = zeros(2,2,k);
covES(:,:,1) = eye(2);
covES(:,:,2) = eye(2);
%compute initial likelihood
sumLH = 0;
for i = 1 : sampleAmount
    sumLH = sumLH + log(w(1) * mvnpdf(sample(i,:),mu(:,:,1)',covES(:,:,1)) + ...
                w(2) * mvnpdf(sample(i,:),mu(:,:,2)',covES(:,:,2)));
end
L = sumLH/sampleAmount;
L_new = 0;
ebsilon = 0.001;
%record data
disp('initial guess')
mu
covES
%plot the initial guess
plot2d2subpopuGMM(mu(:,:,1),mu(:,:,2),covES(:,:,1),covES(:,:,2),w,1,option);

%EM
stepCount = 1;
gamma = zeros(sampleAmount,k);
n = zeros(1,2);
while abs(L_new-L) >  ebsilon
%E-step
    L = L_new;
    for j = 1 : k
        for i = 1 : sampleAmount
```

```
            sumLHPoint = 0;
            for l = 1 : k
                sumLHPoint = sumLHPoint + w(l) * mvnpdf(sample(i,:),mu(:,:,l)',covES(:,:,
                    l));
            end
            gamma(i,j)= w(j) * mvnpdf(sample(i,:),mu(:,:,j)',covES(:,:,j))/sumLHPoint;
        end
    end
    n(1) = sum(gamma(:,1));
    n(2) = sum(gamma(:,2));

%M-step
    w = n/sampleAmount;
    for j = 1 : k
        mu(:,:,j) = (gamma(:,j)' * sample /n(j) )';
        sumCov = zeros(2,2);
        for i = 1 : sampleAmount
            sumCov  = sumCov + gamma(i,j) * (sample(i,:)' - mu(:,:,j))*...
                    (sample(i,:)' - mu(:,:,j))';
        end
        covES(:,:,j)=sumCov / n(j);
    end
%the new likelihood

    stepCount = stepCount + 1;
    plot2d2subpopuGMM(mu(:,:,1),mu(:,:,2),covES(:,:,1),covES(:,:,2),w,stepCount,option);
    disp(strcat('step:',num2str(stepCount)))
    mu
    covES

%Convergence check
    sumLH = 0;
    for i = 1 : sampleAmount
        sumLH = sumLH + log(w(1) * mvnpdf(sample(i,:),mu(:,:,1)',covES(:,:,1)) + ...
            w(2) * mvnpdf(sample(i,:),mu(:,:,2)',covES(:,:,2)));
    end
    L_new = sumLH/sampleAmount;
end
```

filename:plot2d2popuGMM.m

```
function plot2d2subpopuGMM(mu1,mu2,cov1,cov2,w,k,option)
%plot2d2subpopuGMM(mu1,mu2,cov1,cov2,w,k,option)
if option == 0
    figure
else
    hold on
    pause(0.5)
end
mu1 = reshape(mu1,[2,1]);
```

```
mu2 = reshape(mu2,[2,1]);
alpha = 4.7; %scale the plot
if mu1(1) < mu2(1)
    x1rangeLB = mu1(1)- alpha * sqrt(cov1(1,1));
    x1rangeUB = mu2(1)+ alpha * sqrt(cov2(1,1));
else
    x1rangeLB = mu2(1)- alpha * cov2(1,1);
    x1rangeUB = mu1(1)+ alpha * cov1(1,1);
end
x1 = x1rangeLB:0.05:x1rangeUB;

if mu1(2) < mu2(2)
    x2rangeLB = mu1(2)- alpha * sqrt(cov1(2,2));
    x2rangeUB = mu2(2)+ alpha * sqrt(cov2(2,2));
else
    x2rangeLB = mu2(2)- alpha * sqrt(cov2(2,2));
    x2rangeUB = mu1(2)+ alpha * sqrt(cov1(2,2));
end
x2 = x2rangeLB: 0.05 : x2rangeUB;

[X1,X2] = meshgrid(x1,x2);
f = w(1) * mvnpdf([X1(:) X2(:)],mu1',cov1)+w(2) * mvnpdf([X1(:) X2(:)],mu2',cov2);
f = reshape(f,length(x2),length(x1));
h = surf(x1,x2,f);
if k == 0
    title(' true GMM density');
else
    if k == 1
        title('initial guess');
    else
        title(strcat(num2str(k),' step estimation'));
    end
end
axis tight
view(0,90)
set(h,'LineStyle','none')
```