Homework set on GMM, EM and Kmeans

Machine Learning – 361-1-3761

מגיש: עומר לוכסמבורג

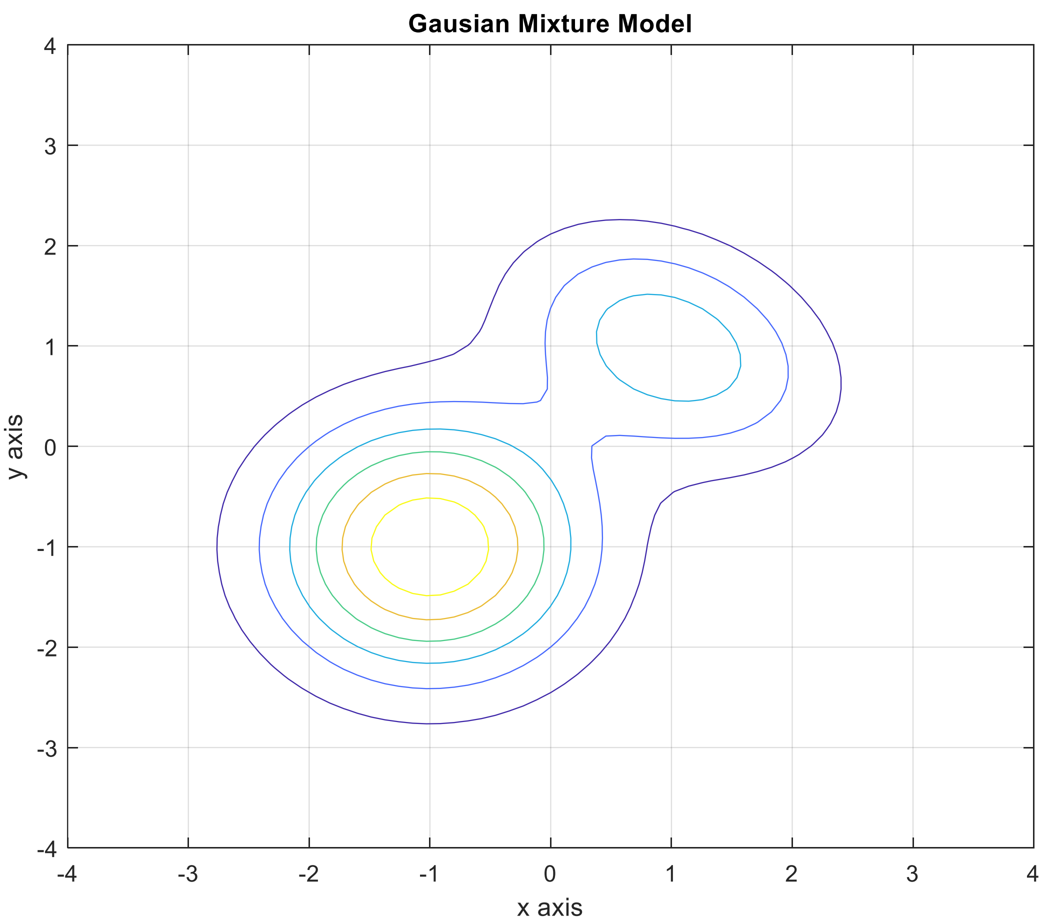
חלק 1 – יצירת המודל:

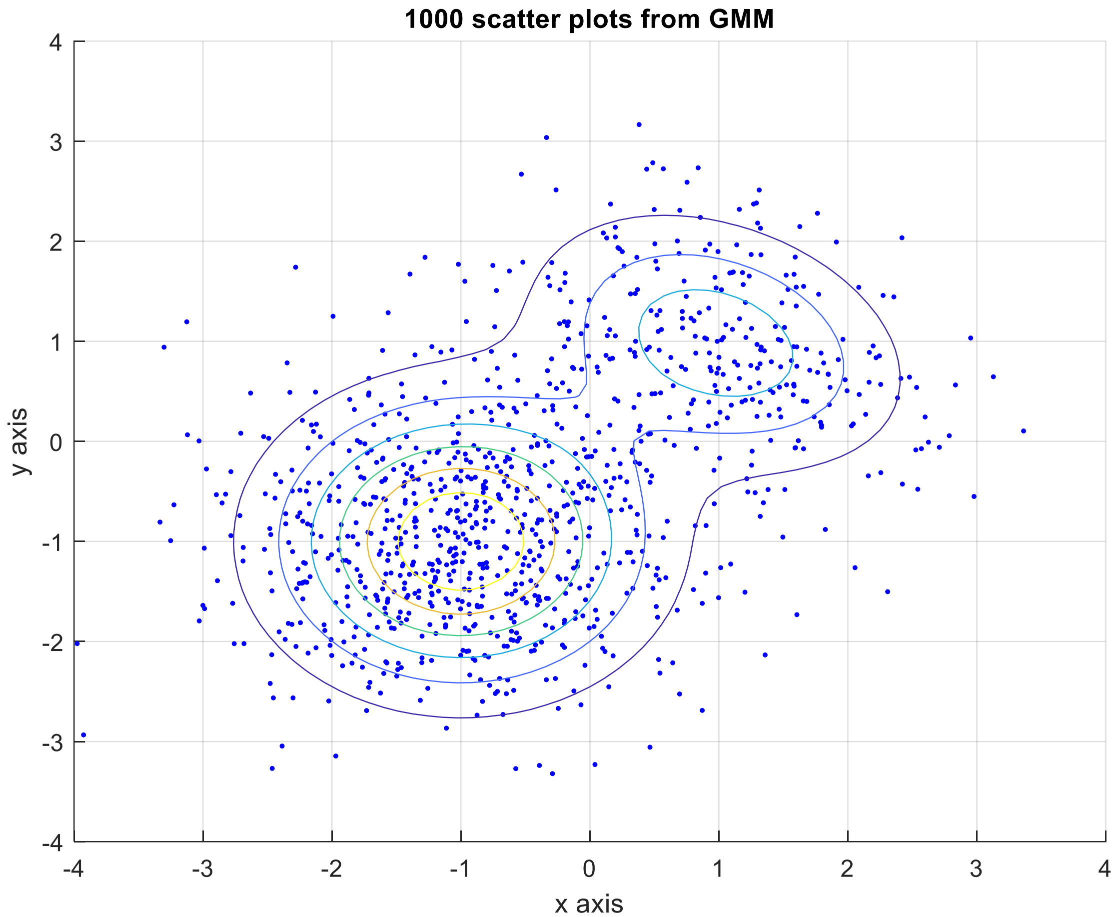
בחלק זה יצרנו את ה- Gaussian mixture model עם 2 גאוסיאנים, בעלי הפרמטרים הבאים:

$$\mu_1 = [-1, -1]^T,$$
$$\mu_2 = [1, 1]^T,$$
$$\Sigma_1 = \begin{pmatrix} 0.8 & 0 \\ 0 & 0.8 \end{pmatrix},$$
$$\Sigma_2 = \begin{pmatrix} 0.75 & -0.2 \\ -0.2 & 0.6 \end{pmatrix},$$
$$P_Z(z = 1) = 0.7.$$

השתמשתי בפונקצית gmdistribution ב-MATLAB.

גרף ראשון הוא רק ה-GMM והגרף השני הוא ה-GMM ו-1000 נקודות רנדומליות שנלקחו מהמודל.





1000 scatter plots from GMM
y axis
x axis
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4

חלק 2 – אלגוריתם Kmeans:

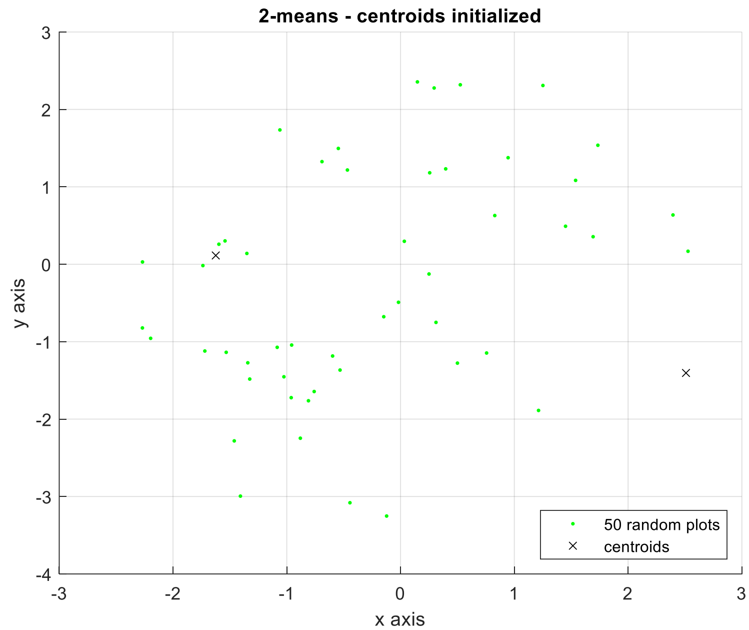
בחלק זה מימשנו את אלגוריתם *Kmeans*.

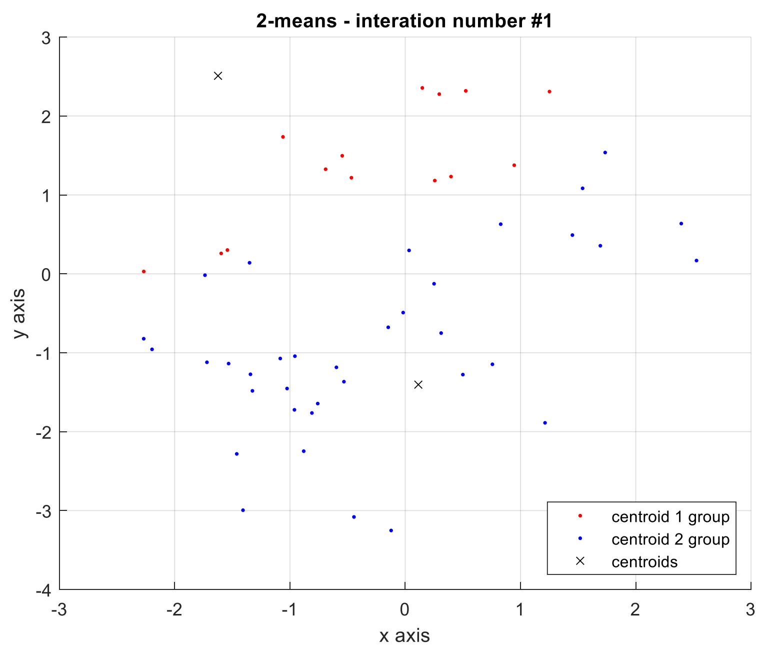
הגרלנו 50 נקודות מהתפלגות ה-GMM.

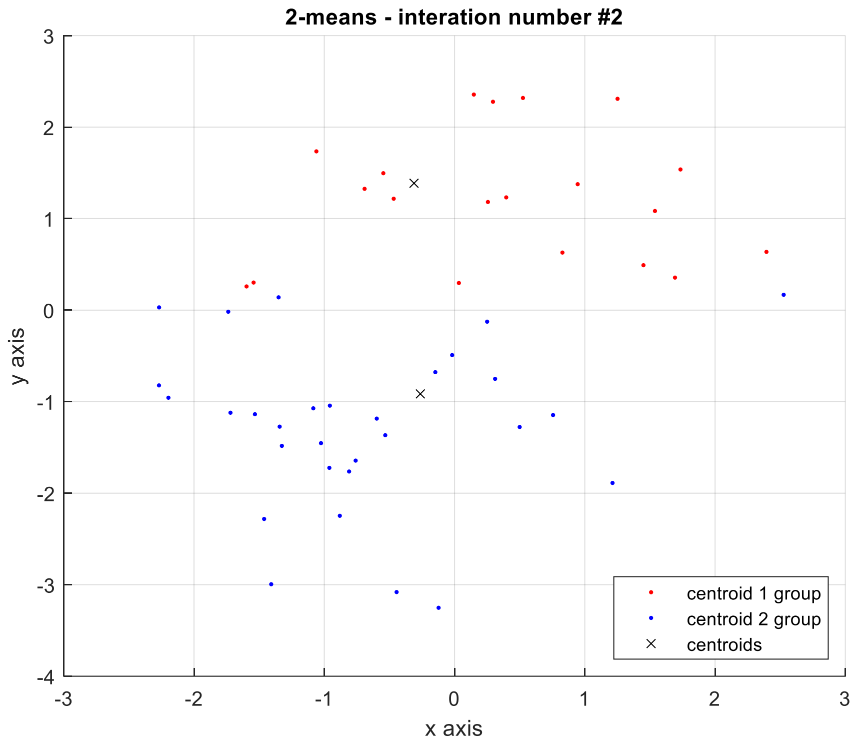
הגרלנו 2 *centroids* שיהוו את נקודות המיצוע ההתחלתיות (בוצע באופן אקראי במיקום הסמוך ל-GMM).

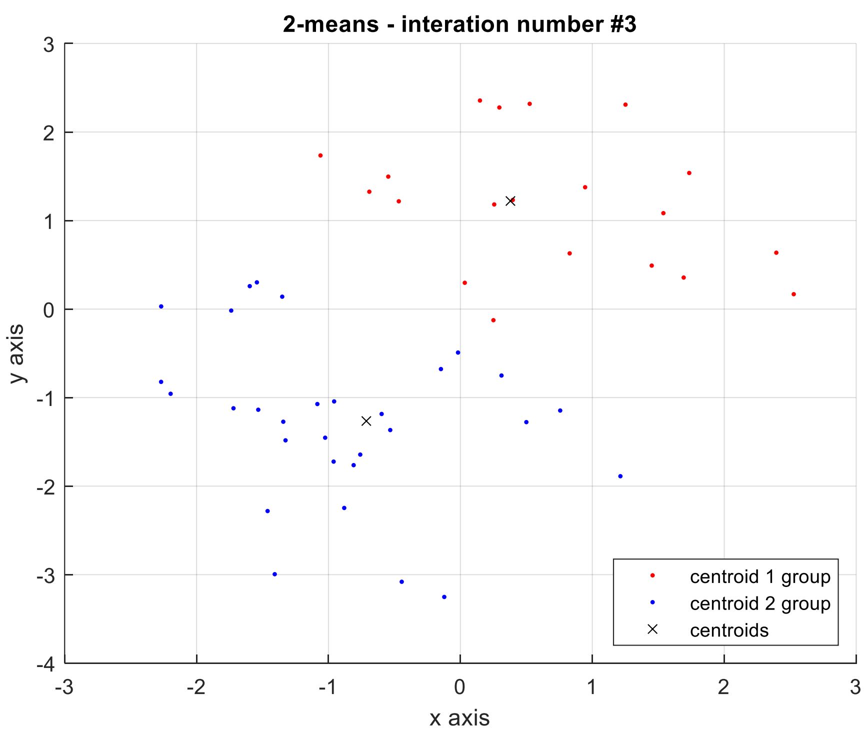
מספר האיטרציות המקסימלי שקבעתי הוא 20, ואפסילון (ההפרש בין השינויים) שקבעתי הוא 10^{-3}.

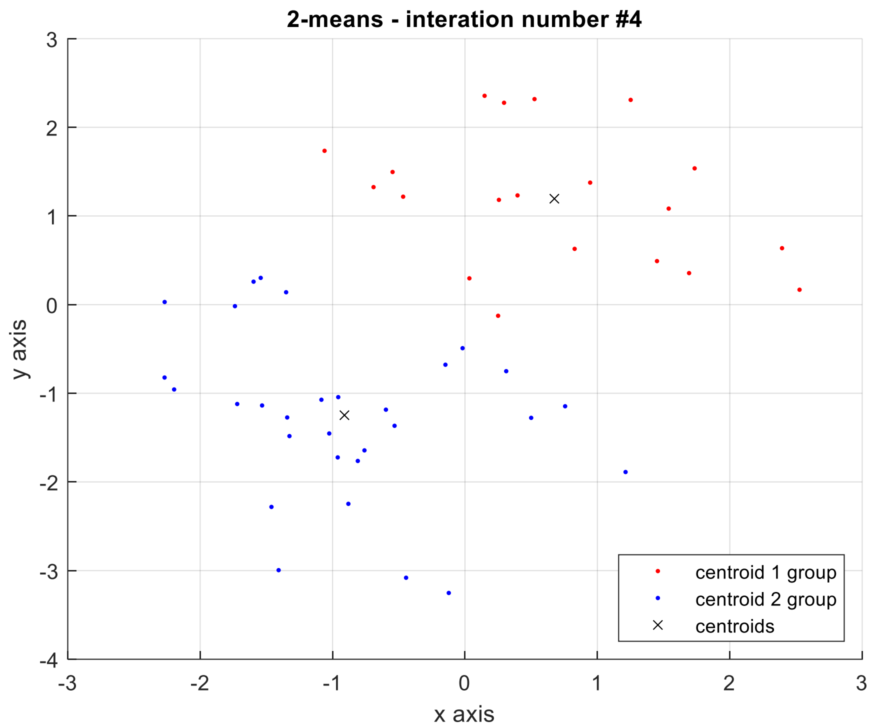
להלן התוצאות לפי סדר האיטרציה של האלגוריתם *Kmeans*:

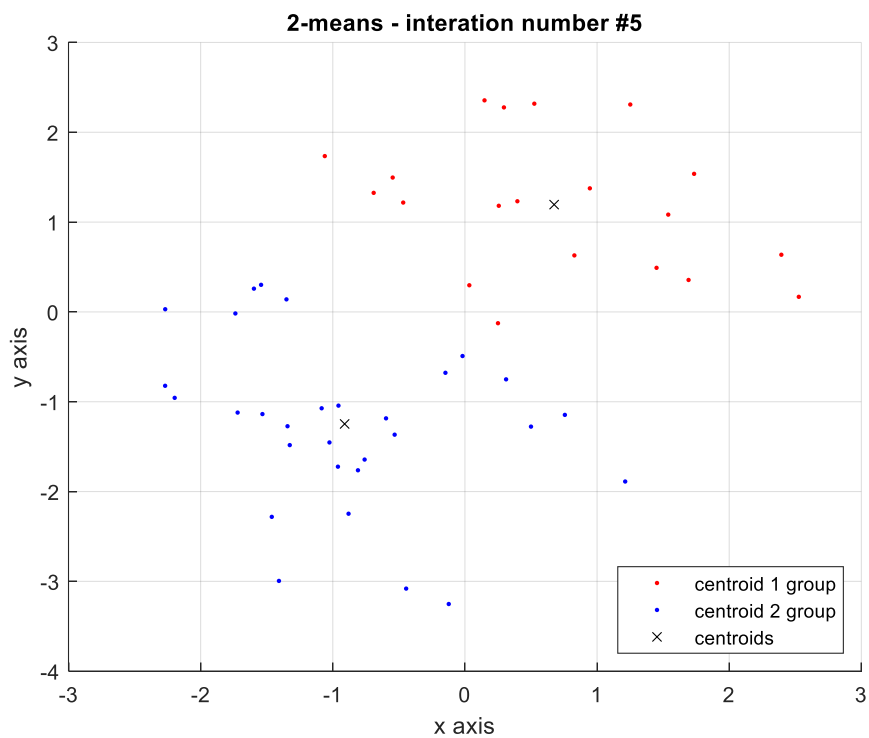





2-means - interation number #2
3
2
1
0
-1
-2
-3
-4
y axis
-3
-2
-1
0
1
2
3
x axis
centroid 1 group
centroid 2 group
centroids


2-means - interation number #3
3
2
1
0
-1
-2
-3
-4
y axis
-3
-2
-1
0
1
2
3
x axis
centroid 1 group
centroid 2 group
centroids





לאחר 5 איטרציות ניתן לראות כי ה'מרכזים' התאזנו בסביבת התוחלות של הפונקציות הגאוסיאניות מתוך ה-GMM.

כל החישובים בוצעו על פי אלגוריתם *Kmeans* ובפרט נוסחאות 33,34 בהרצאה "Lecture 2: GMM and EM".

להלן האלגוריתם:

<u>Kmeans Algorithem:</u>

1) Initialize centroids $\mu_1 \ldots \mu_k \in \mathbb{R}^n$
2) Repeat until convergence: $\{$
   a) for every i, set
   $$c^{(i)} := \arg\min_j \|x^{(i)} - \mu_j\|^2 \tag{33}$$
   b) for each j, set
   $$\mu_j := \frac{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} \mathbf{1}\{c^{(i)} = j\}} \tag{34}$$

$\}$

חלק 3 – אלגוריתם EM (Expectation maximization):

בחלק זה מימשנו את אלגוריתם EM עבור מציאת GMM מתאים למקבץ נקודות.

הגרלנו 1000 דגימות מתוך המודל שצוין בחלק 1.

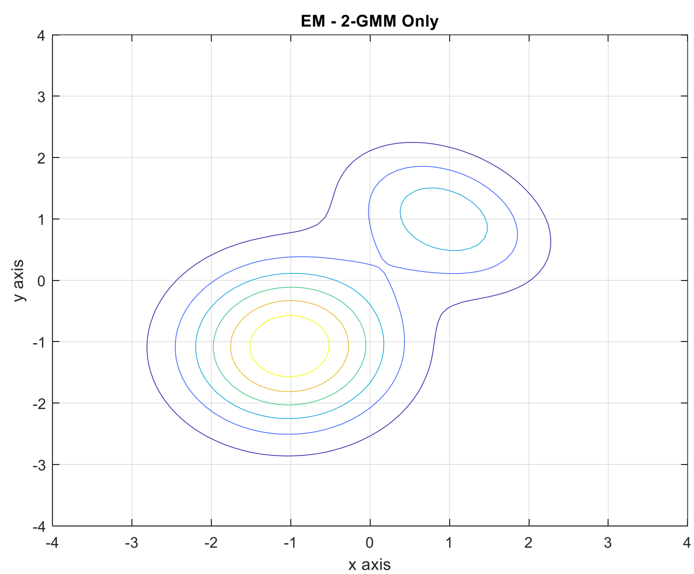
לאחר מכן ביצענו את אלגוריתם EM.

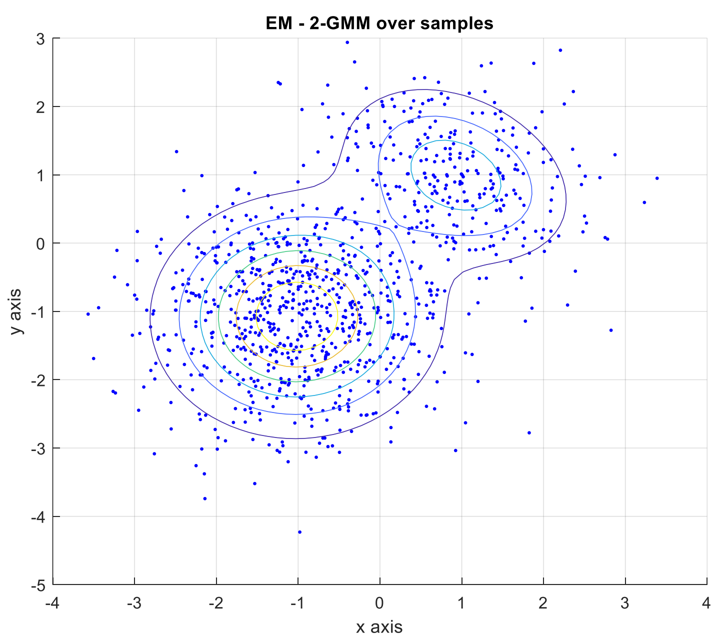
לכל איטרציה חישבנו את ה- Log Likelihood.

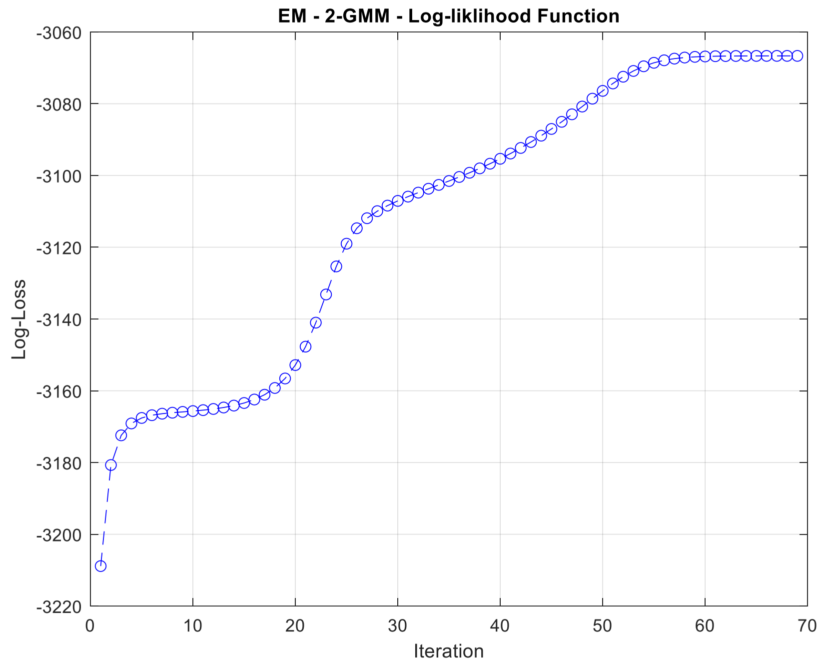
הגרלנו GMM אקראי במיקום הקרוב לנקודות. קבענו את מספר האיטרציות המקסימלי להיות 100 וכן את אפסילון 10^{-3} (כאשר כעת נבדוק בעזרתו את הפרשי ה-Log loss).

להלן התוצאות לאחר הפעלת האלגוריתם:

- 2 גאוסיאנים –



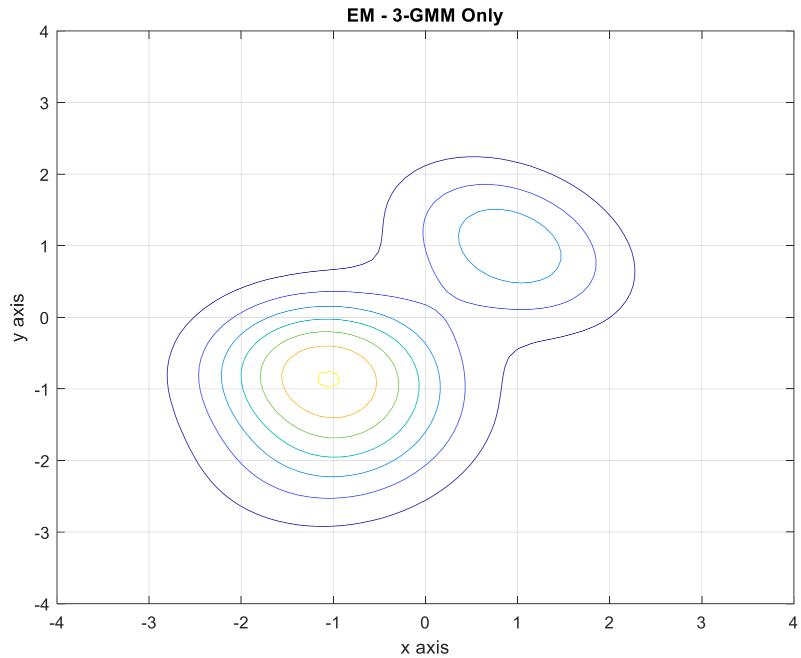




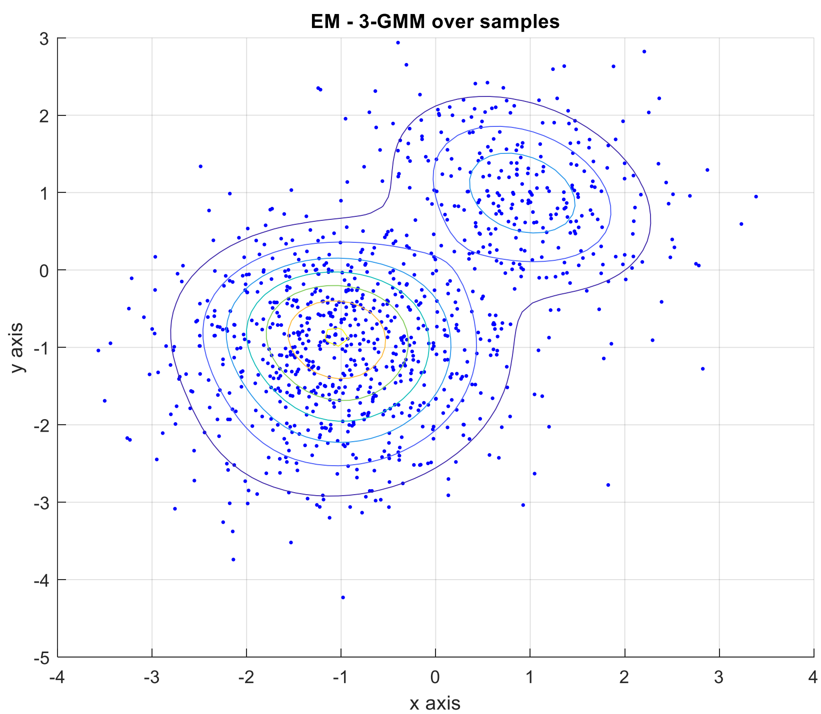
ניתן לראות כי לאחר כ-60 איטרציות הצלחנו לשחזר בדיוק את ה-*GMM* שהוצג בחלק 1. (בגרפים בחלק זה שוחזר ה-*Gaussian mixture model* על פי הנקודות).
ניתן לראות שאכן ערך הפונקצית ה-*Log-Likelihood* התכנס אל המקסימום.

בעמוד הבא נראה את התוצאה של חלק זה עבור 3 גאוסיאנים.

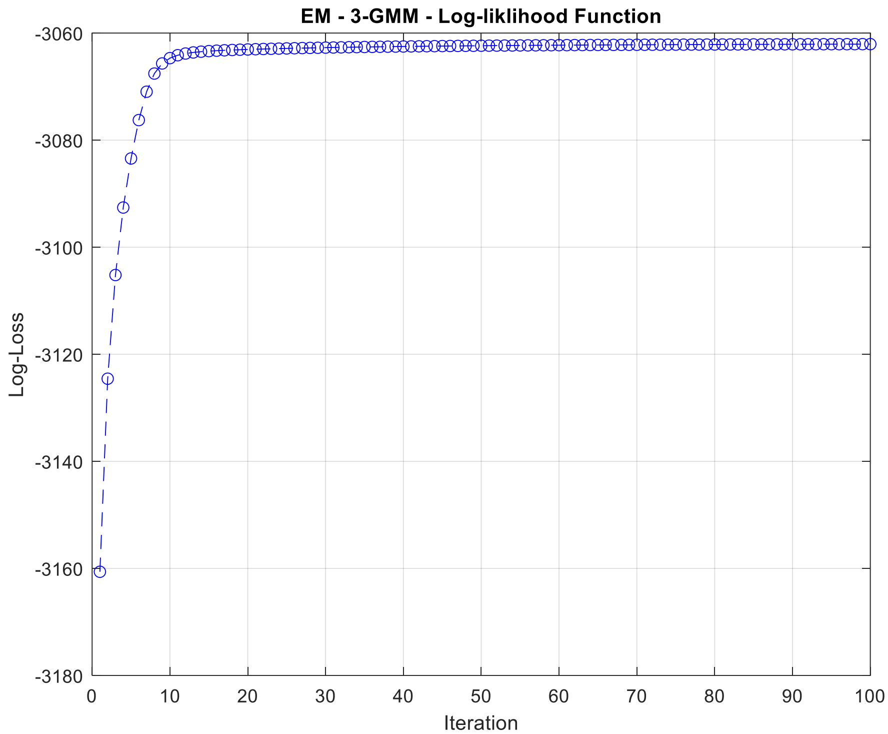
- 3 גאוסיאנים –



EM - 3-GMM Only
y axis
x axis
4
3
2
1
0
-1
-2
-3
-4
-4
-3
-2
-1
0
1
2
3
4



EM - 3-GMM over samples
y axis
x axis
3
2
1
0
-1
-2
-3
-4
-5
-4
-3
-2
-1
0
1
2
3
4



ניתן לראות כי למרות שהגדרנו כי קיימים 3 גאוסיאנים, באים לביטוי רק 2 מהם (למעשה מקדם הגאוסיאן השלישי שואף לאפס, $\phi(3) \approx 0$) ולכן קיבלנו גם כאן את אותו המודל שצויין בחלק 1, כפי שציפינו.

ניתן לשים לב לעיוותים קלים במודל זה יחסית למודל של 2 גאוסיאנים, וזאת מכיוון שעדיין קיים משקל לגאוסיאן השלישי שאינו 0.

בנוסף ניתן לראות כי לאחר כ-10 איטרציות יכלנו לעצור את התהליך, מכיוון שפונקצית ה-*logloss* לא השתנתה הרבה.

כל החישובים בוצעו על פי אלגוריתם *EM* ובפרט נוסחאות 29-32 בהרצאה "Lecture 2: GMM and EM".

להלן האלגוריתם:

<u>EM Algorithem:</u>

function EM($x^n, \theta^{(0)}$)

  for iteration $t \in 1, 2, ...$ **do**

    $Q^{(t)}(z_i) = P(z_i|x_i; \theta^{(t-1)}) \quad \forall i = 1, 2, ..., n \quad \forall z_i \in \mathcal{Z}$ **E-step**

    $\theta^{(t)} = \text{argmax}_\theta \, \mathbb{E}_{Q^{(t)}_{Z^n}}[\log(P(x^n, Z^n; \theta))]$ **M-step**

    if $\log P(x^n|\theta^{(t)}) - \log P(x^n|\theta^{(t-1)}) < \epsilon$ **then**

      return $\theta^{(t)}$

E-step: for each i, j

$$w(j,i) := P(z_i = j | x_i; \phi, \mu, \Sigma) \tag{29a}$$

$$= \frac{\phi(j) P(x_i|z_i = j; \mu_j, \Sigma_j)}{\sum_{l=1}^{k} \phi(l) P(x_i|z_i = l; \mu_l, \Sigma_l)} \tag{29b}$$

M-step: for each j

$$\phi(j) := \frac{1}{m} \sum_{i=1}^{m} w(j,i) \tag{30}$$

$$\mu_j := \frac{\sum_{i=1}^{m} w(j,i) x_i}{\sum_{i=1}^{m} w(j,i)} \tag{31}$$

$$\Sigma_j := \frac{\sum_{i=1}^{m} w(j,i)(x^{(i)} - \mu_j)(x_i - \mu_j)^T}{\sum_{i=1}^{m} w(j,i)} \tag{32}$$

<u>משוואת ה-*log-likelihood*:</u>

We define ℓ as the log-likelihood function. To estimate our parameters , we can write the log-likelihood of our data:

$$\ell(\phi, \mu, \Sigma) = \sum_{i=1}^{m} \log P(x_i; \phi, \mu, \Sigma) \tag{4a}$$

$$= \sum_{i=1}^{m} \log \sum_{j=1}^{k} P(x_i|z = j; \mu_j, \Sigma_j) P(z = j; \mu_j, \Sigma_j) \tag{4b}$$

<u>קוד מטלאב</u>

<u>hw4.m</u>

```matlab
%% Part 1 - Data Generation
mu1 = [-1 -1];
mu2 = [1 1];
mu = [mu1 ; mu2];                    % Means
sigma1 = [.8 0;0 .8];
sigma2 = [.75 -0.2;-0.2 .6];
sigma = zeros(2,2,2);
sigma(:,:,1) = sigma1;
sigma(:,:,2) = sigma2;               % Covariances
p = [0.7 0.3];                       % Mixing proportions p(z=1)=0.7

gmm = gmdistribution(mu,sigma,p);  % gmm is the gaussian mixture model

Y_1000 = random(gmm,1000);    % generates 1000 plots of the GMM

figure          % ploting only gmm
gmPDF = @(x1,x2)reshape(pdf(gmm,[x1(:) x2(:)]),size(x1));
fcontour(gmPDF,[[-4,4] [-4,4]])
grid on
xlabel('x axis')
ylabel('y axis')
title('Gausian Mixture Model')

figure         % ploting scatter and gmm
scatter(Y_1000(:,1),Y_1000(:,2),'.','b');
hold on
gmPDF = @(x1,x2)reshape(pdf(gmm,[x1(:) x2(:)]),size(x1));
g = gca;
fcontour(gmPDF,[g.XLim g.YLim])
title('1000 scatter plots from GMM')
grid on
xlabel('x axis')
ylabel('y axis')
hold off

%% Part 2 - K means
X_50 = random(gmm,50);      % generates 50 plots of the GMM
K = 20;
epsilon = 10^-3;            % to stop the iteration

centroids = zeros(2,2,K+1);
rng shuffle
centroids(:,:,1) = -3.5 + (3.5+3.5)*rand(2,2);  % initialized 2 centroids in a square
%centroids(:,:,1) = random(gmm,2);  % initialized 2 centroids

figure
scatter(X_50(:,1),X_50(:,2),'.','g');
hold on
scatter(centroids(1,:,1),centroids(2,:,1),'x', 'k');
grid on
xlabel('x axis')
ylabel('y axis')
title('2-means - centroids initialized')
legend({'50 random plots','centroids'},'Location','southeast')
hold off

for k = 1:K
```

```matlab
    cost = zeros(50,1);          % cost_i = arg(j)min || x_i - mu_j ||
    clear group1
    clear group2
    g1 = 1;
    g2 = 1;
    group1 = zeros(1,2);         % saving groups for different centroids
    group2 = zeros(1,2);         %
    for i = 1:50                 % for each xi - creating the argmin vector
        C1= norm((X_50(i,:)' - centroids(1,:,k)'),2);
        C2= norm((X_50(i,:)' - centroids(2,:,k)'),2);
        if C1 < C2
            cost(i) = 1;
            group1(g1,:) = X_50(i,:);
            g1 = g1 + 1;
        else
            cost(i) = 2;
            group2(g2,:) = X_50(i,:);
            g2 = g2 + 1;
        end
    end

    % ploting new centroids
    figure
    scatter(group1(:,1),group1(:,2),'.','r');
    hold on
    scatter(group2(:,1),group2(:,2),'.','b');
    hold on
    scatter(centroids(:,1,k),centroids(:,2,k),'x', 'k');
    grid on
    xlabel('x axis')
    ylabel('y axis')
    title(sprintf('2-means - interation number #%d', k))
    legend({'centroid 1 group','centroid 2 group','centroids'},'Location','southeast')
    hold off

    % Check lecture 6 to see the k-means algorithem
    upsum1 = zeros(1,2);
    dosum1 = 0;
    upsum2 = zeros(1,2);
    dosum2 = 0;
    for i = 1:50
        if cost(i)==1       % argmin is 1
            upsum1 = upsum1 + 1*X_50(i,:);
            dosum1 = dosum1 + 1;
        else                % argmin is 2
            upsum2 = upsum2 + 1*X_50(i,:);
            dosum2 = dosum2 + 1;
        end
    end
    centroids(1,:,k+1) = upsum1 ./ dosum1;  % calculating new centroids
    centroids(2,:,k+1) = upsum2 ./ dosum2;

    if norm(centroids(:,:,k+1) - centroids(:,:,k)) < epsilon
        break;  % change is too low
    end
end

% ploting new centroids
figure
scatter(group1(:,1),group1(:,2),'.','r');
hold on
scatter(group2(:,1),group2(:,2),'.','b');
```

```
hold on
scatter(centroids(:,1,k+1),centroids(:,2,k+1),'x', 'k');
grid on
xlabel('x axis')
ylabel('y axis')
title(sprintf('2-means - interation number #%d', k+1))
legend({'centroid 1 group','centroid 2 group','centroids'},'Location','southeast')
hold off

%% Part 3 - EM
Data = random(gmm,1000);     % generates 1000 plots of the GMM
K=100;                        % maximum iteration
epsilon = 10^-3;             % to stop the iteration

%making initial guess
Param = struct();
rng shuffle
Param.mu = -3 + (3+3)*rand(1,2,2);

rng shuffle
side = -1 + (1+1)*rand(1);
rng shuffle
diag = abs(side) + 2*rand(1,2);
sig1 = [diag(1,1), side; side, diag(1,2)];

rng shuffle
side = -1 + (1+1)*rand(1);
rng shuffle
diag = abs(side) + 2*rand(1,2);
sig2 = [diag(1,1), side; side, diag(1,2)];

Param.sigma(:,:,1) = sig1;
Param.sigma(:,:,2) = sig2;
rng shuffle
p = rand(1);
Param.phi = [p,1-p];         % probabilities for each gaussian

logloss = [];
for k=1:K
    % Expectation step
    Qz = expectation(Data,Param, 2);  % 2 gaussians. Data_new has the labels
    % Maximization step
    Param = maximization(Qz,Data,Param, 2);% 2 gaussians. Param_new has the new
gaussians
    % Log likelihood
    logloss(k) = loglike(Data, Param, 2); % 2 gausians. logloss hopefuly maximized

    if k >= 2 && norm(logloss(k)-logloss(k-1))<= epsilon
        break;
    end
end

mu = [Param.mu(:,:,1);Param.mu(:,:,2)];
sigma = Param.sigma;
phi = Param.phi;
gmm = gmdistribution(mu,sigma,phi);  % gmm is the gaussian mixture model

figure          % ploting only gmm
gmPDF = @(x1,x2)reshape(pdf(gmm,[x1(:) x2(:)]),size(x1));
fcontour(gmPDF,[[-4,4] [-4,4]])
grid on
```

```
xlabel('x axis')
ylabel('y axis')
title('EM - 2-GMM Only')

figure          % ploting scatter and gmm
scatter(Data(:,1),Data(:,2),'.','b');
hold on
gmPDF = @(x1,x2)reshape(pdf(gmm,[x1(:) x2(:)]),size(x1));
g = gca;
fcontour(gmPDF,[g.XLim g.YLim])
title('EM - 2-GMM over samples')
xlabel('x axis')
ylabel('y axis')
grid on
hold off

figure           % ploting log likelihood
iter = 1:k;
plot (iter,logloss,'b--o');
title('EM - 2-GMM - Log-liklihood Function')
xlabel('Iteration')
ylabel('Log-Loss')
grid on


%% Part 3 - for 3
K=100;                        % maximum iteration
epsilon = 10^-3;             % to stop the iteration

%making initial guess
Param = struct();
rng shuffle
Param.mu = -3 + (3+3)*rand(1,2,3);

rng shuffle
side = -1 + (1+1)*rand(1);
rng shuffle
diag = abs(side) + 2*rand(1,2);
sig1 = [diag(1,1), side; side, diag(1,2)];

rng shuffle
side = -1 + (1+1)*rand(1);
rng shuffle
diag = abs(side) + 2*rand(1,2);
sig2 = [diag(1,1), side; side, diag(1,2)];

rng shuffle
side = -1 + (1+1)*rand(1);
rng shuffle
diag = abs(side) + 2*rand(1,2);
sig3 = [diag(1,1), side; side, diag(1,2)];

Param.sigma(:,:,1) = sig1;
Param.sigma(:,:,2) = sig2;
Param.sigma(:,:,3) = sig3;
rng shuffle
Param.phi = [1/3,1/3,1/3];        % probabilities for each gaussian

logloss = [];
for k=1:K
    % Expectation step
```

```
    Qz = expectation(Data,Param, 3);  % 3 gaussians. Data_new has the labels
    % Maximization step
    Param = maximization(Qz,Data,Param, 3);% 3 gaussians. Param_new has the new
gaussians
    % Log likelihood
    logloss(k) = loglike(Data, Param, 3); % 3 gausians. logloss hopefuly maximized

    if k >= 2 && norm(logloss(k)-logloss(k-1))<= epsilon
        break;
    end
end

mu = [Param.mu(:,:,1);Param.mu(:,:,2);Param.mu(:,:,3)];
sigma = Param.sigma;
phi = Param.phi;
gmm = gmdistribution(mu,sigma,phi);  % gmm is the gaussian mixture model

figure          % ploting only gmm
gmPDF = @(x1,x2)reshape(pdf(gmm,[x1(:) x2(:)]),size(x1));
fcontour(gmPDF,[[-4,4] [-4,4]])
grid on
xlabel('x axis')
ylabel('y axis')
title('EM - 3-GMM Only')

figure         % ploting scatter and gmm
scatter(Data(:,1),Data(:,2),'.','b');
hold on
gmPDF = @(x1,x2)reshape(pdf(gmm,[x1(:) x2(:)]),size(x1));
g = gca;
fcontour(gmPDF,[g.XLim g.YLim])
title('EM - 3-GMM over samples')
xlabel('x axis')
ylabel('y axis')
grid on
hold off

figure          % ploting log likelihood
iter = 1:k;
plot (iter,logloss,'b--o');
title('EM - 3-GMM - Log-liklihood Function')
xlabel('Iteration')
ylabel('Log-Loss')
grid on
```

פונקציות נוספות בהמשך

expectation.m

```
function Qz = expectation(Data, Param, numG)
% returns Qz - the weights Q which the gaussians will be computed by

mu1 = Param.mu(:,:,1);
mu2 = Param.mu(:,:,2);
sigma1 = Param.sigma(:,:,1);
sigma2 = Param.sigma(:,:,2);
phi = Param.phi;

if numG == 3
    mu3 = Param.mu(:,:,3);
    sigma3 = Param.sigma(:,:,3);
    Qz = zeros(size(Data,1),3);
else
    Qz = zeros(size(Data,1),2);
end



for i=1:size(Data,1)
    x = Data(i,1:2);  % taking 1 sample

    % mvnpdf will be the W_ij and the phi is the phi(j)
    p1 = phi(1,1) * mvnpdf (x, mu1, sigma1);   % check probability of gaussian 1
    p2 = phi(1,2) * mvnpdf (x, mu2, sigma2);   % check probability of gaussian 2

    % Qz is according to 29a,b
    if numG == 3
        p3 = phi(1,3) * mvnpdf (x, mu3, sigma3);   % check probability of gaussian 3
        Qz(i,1:3) = [p1/(p1+p2+p3), p2/(p1+p2+p3), p3/(p1+p2+p3)]; % creating Q -
weights
    else
        Qz(i,1:2) = [p1/(p1+p2), p2/(p1+p2)];      % creating Q - weights
    end
end
```

maximization.m

```
function Param = maximization(Qz, Data, Param, numG)
% Maximization step -
%returns the new parameters of the gaussians according to the labels
% we gathered from the expectation step

% Calculating phi - eq. 30
Param.phi = sum(Qz(:,:)) / size(Qz,1);

% Calculating mu - eq. 31
% mu1
Param.mu(:,:,1) = [sum(Qz(:,1) .* Data(:,1)), sum(Qz(:,1) .* Data(:,2)),] ./
sum(Qz(:,1));
% mu2
Param.mu(:,:,2) = [sum(Qz(:,2) .* Data(:,1)), sum(Qz(:,2) .* Data(:,2)),] ./
sum(Qz(:,2));
% mu3
if numG == 3
    Param.mu(1,:,3) = [sum(Qz(:,3) .* Data(:,1)), sum(Qz(:,3) .* Data(:,2)),] ./
sum(Qz(:,3));
end

% Calculating sigma - eq. 32
% sigma 1
sum_sigma1 = zeros(2,2);
for i=1:size(Data,1)
    sum_sigma1 = sum_sigma1 +  Qz(i,1).* ...
        ((Data(i,:)' - Param.mu(:,:,1)')*(Data(i,:)' - Param.mu(:,:,1)')');
end
Param.sigma(:,:,1) = sum_sigma1 ./ sum(Qz(:,1));
% sigma 2
sum_sigma2 = zeros(2,2);
for i=1:size(Data,1)
    sum_sigma2 = sum_sigma2 +  Qz(i,2).* ...
        ((Data(i,:)' - Param.mu(:,:,2)')*(Data(i,:)' - Param.mu(:,:,2)')');
end
Param.sigma(:,:,2) = sum_sigma2 ./ sum(Qz(:,2));
% sigma 3
if numG == 3
    sum_sigma3 = zeros(2,2);
    for i=1:size(Data,1)
        sum_sigma3 = sum_sigma3 +  Qz(i,3).* ...
            ((Data(i,:)' - Param.mu(:,:,3)')*(Data(i,:)' - Param.mu(:,:,3)')');
    end
    Param.sigma(:,:,3) = sum_sigma3 ./ sum(Qz(:,3));
end
end
```

likelog.m

```
function LogLike = loglike(Data, Param, numG)
%returning the log likelihood of the function - eq.10b
LogLike = 0;

if numG == 3
    for i = 1:size(Data,1)
        x = Data(i,1:2);  % taking 1 sample
        % gi is the prob of gaussian x sample
        g1 = Param.phi(1,1) * mvnpdf(x, Param.mu(:,:,1), Param.sigma(:,:,1));
        g2 = Param.phi(1,2) * mvnpdf(x, Param.mu(:,:,2), Param.sigma(:,:,2));
        g3 = Param.phi(1,3) * mvnpdf(x, Param.mu(:,:,3), Param.sigma(:,:,3));
        LogLike = LogLike + log(g1+g2+g3);
    end
else
    for i = 1:size(Data,1)
        x = Data(i,1:2);  % taking 1 sample
        % gi is the prob of gaussian x sample
        g1 = Param.phi(1,1) * mvnpdf(x, Param.mu(:,:,1), Param.sigma(:,:,1));
        g2 = Param.phi(1,2) * mvnpdf(x, Param.mu(:,:,2), Param.sigma(:,:,2));
        LogLike = LogLike + log(g1+g2);
    end
end
end
```