CS 8735: Report for assignment 1

Chanmann Lim

September 17, 2015

The Matlab code for all experiments is in the **Appendix** section.

Problem 1. In this task, we are given a dataset generated from a mixture density and the job is to implement EM algorithm to learn the parameters of the model. Based on the assumption that the Gaussian Mixture Model has four component Gaussian PDFs with each having a full covariance matrix we will terminate the our EM estimation at the 100th iterations.

a) For the first experiment which we named it case **a**, we run EM procedure with the initialization suggested in the assignment.

$$\pi_k^{(0)} = 1/4 \qquad 1 \le k \le 4$$

$$\mu_1^{(0)} = [10\ 2]^T, \mu_2^{(0)} = [5\ 6]^T, \mu_3^{(0)} = [0\ 1]^T, \mu_4^{(0)} = [4\ 3]^T$$

$$\Sigma_k^{(0)} = \mathbf{I}_{2\times 2} \qquad 1 \le k \le 4$$

After the EM procedure terminated, we got

$$\hat{\pi}_1 = 0.3459, \hat{\pi}_2 = 0.1412, \hat{\pi}_3 = 0.1850, \hat{\pi}_4 = 0.3280 \tag{1}$$

$$\begin{aligned} \hat{\mathbf{U}} &= \begin{bmatrix} \hat{\mu}_1 & \hat{\mu}_2 & \hat{\mu}_3 & \hat{\mu}_4 \end{bmatrix} && (2) \\ &= \begin{bmatrix} 13.0253 & 4.0666 & 1.6031 & 6.9285 \\ 3.0467 & 7.9557 & 1.5747 & 5.9848 \end{bmatrix} && (3) \end{aligned}$$

$$\begin{aligned} \hat{\mathbf{\Sigma}} &= \begin{bmatrix} \hat{\Sigma}_1 & \hat{\Sigma}_2 & \hat{\Sigma}_3 & \hat{\Sigma}_4 \end{bmatrix} && (4) \\ &= \begin{bmatrix} 1.6491 & 0.8845 & 8.4402 & 6.2493 \\ -0.7494 & 0.2297 & -0.0623 & 2.6326 \\ 2.0717 & 1.1715 & 1.0987 & 1.9615 \end{bmatrix} && (5) \end{aligned}$$

Where, $\hat{\Sigma_k}$ is the upper triangular values for covariance matrix of the k^{th} Gaussian component.

$$1 \le k \le 4$$

Figure 1 shows that EM has converged after the 97th iteration.

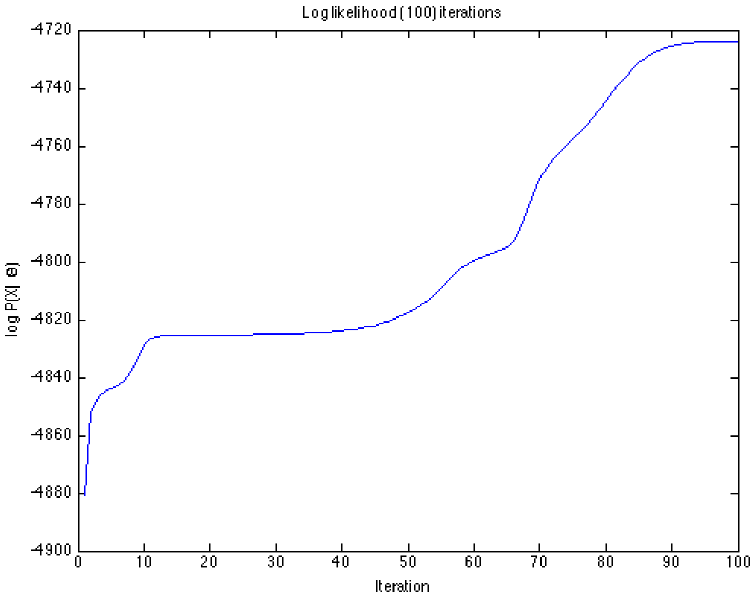

Figure 1: Log likelihood scores for case **a**

To see the effect of EM algorithm visually we assign each data point to one of the four clusters $k = 1, 2, 3, 4$ using the maximum posterior probability rule then plot three separate graphs for $t = 10, 50, 100$.

$$k^* = \underset{1 \le k \le 4}{\operatorname{argmax}} P(z_n = k | x_n; \Theta^{(t)})$$

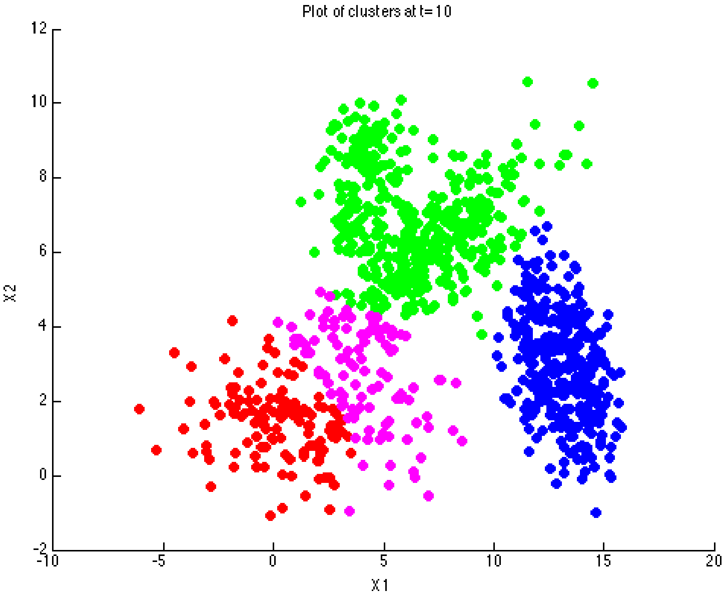

Figure 2: Plot of the four clusters at t=10

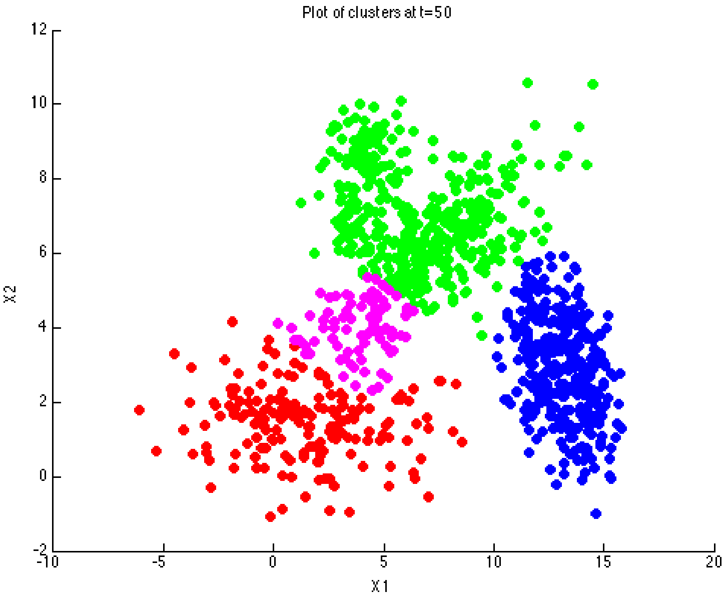

Figure 3: Plot of the four clusters at t=50

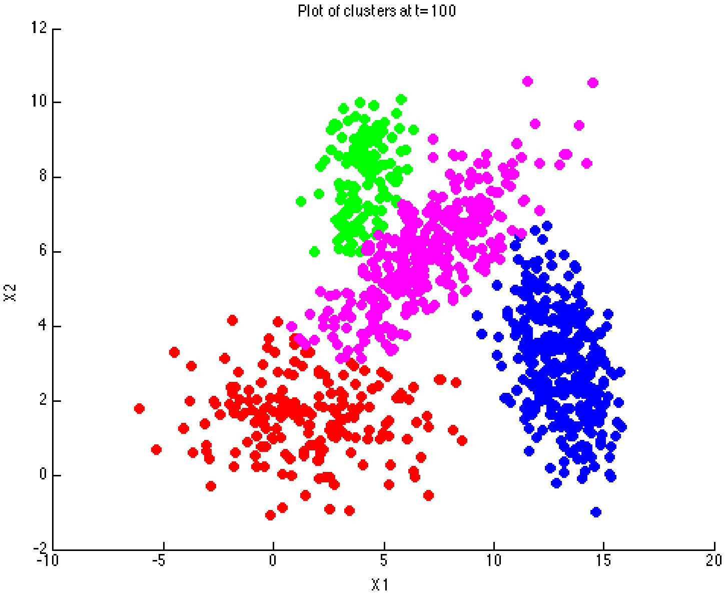

Figure 4: Plot of the four clusters at t=100

b) For the second experiment(case **b**) with the same dataset we are going to use a different initialization for the parameters $\Theta^{(0)} = \{\pi^{(0)}, \mu^{(0)}, \Sigma^{(0)}\}$ under the same assumption that the data comes from four components gaussian mixture model and EM procedure will converge at the 100^{th} iterations.

The plot of the data will actually help reveal its natural grouping to some extent before our blind guess and this is especially true for two dimensional dataset like in this problem.

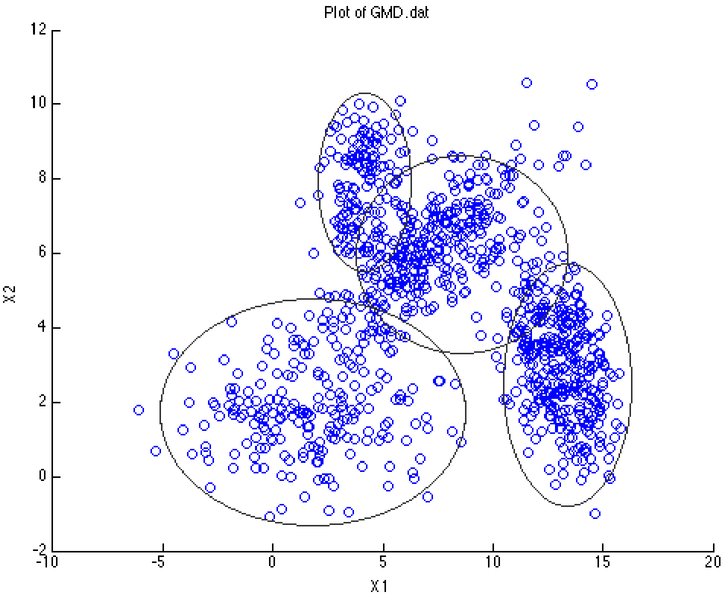

Figure 5: Plot of GMD.dat

And from Figure 5 we comes up with $\Theta^{(0)}$ as the following:

$$\pi_1^{(0)} = 0.25, \pi_2^{(0)} = 0.2, \pi_3^{(0)} = 0.25, \pi_4^{(0)} = 0.3$$

$$\mu_1^{(0)} = [1 \quad 2]^T, \mu_2^{(0)} = [4 \quad 8]^T, \mu_3^{(0)} = [8 \quad 6.5]^T, \mu_4^{(0)} = [13.5 \quad 3]^T$$

$$\Sigma_k^{(0)} = \mathbf{I}_{2\times 2} \qquad 1 \leq k \leq 4$$

Empirically we can select several points closed to each already chosen $\mu_k^{(0)}$ at random to compute for the covariance matrix Σ however that wouldn't guarantee to give measurable accuracy then any purely random guess covariance matrix than using the same covariance matrix $\Sigma_k^{(0)} = \mathbf{I}_{2\times 2}$ as in case **a** will be as satisfactory.

And the EM procedure terminated with

$$\hat{\pi}_1 = 0.1847, \hat{\pi}_2 = 0.1401, \hat{\pi}_3 = 0.3295, \hat{\pi}_4 = 0.3457 \tag{6}$$

$$\hat{\mathbf{U}} = \begin{bmatrix}\hat{\mu}_1 & \hat{\mu}_2 & \hat{\mu}_3 & \hat{\mu}_4\end{bmatrix} \tag{7}$$

$$= \begin{bmatrix}1.6026 & 4.0619 & 6.9182 & 13.0263\\ 1.5717 & 7.9675 & 5.9843 & 3.0455\end{bmatrix} \tag{8}$$

$$\hat{\mathbf{\Sigma}} = \begin{bmatrix}\hat{\Sigma}_1 & \hat{\Sigma}_2 & \hat{\Sigma}_3 & \hat{\Sigma}_4\end{bmatrix} \tag{9}$$

$$= \begin{bmatrix}8.4468 & 0.8788 & 6.2733 & 1.6470\\ -0.0635 & 0.2342 & 2.6295 & -0.7471\\ 1.0938 & 1.1568 & 1.9615 & 2.0688\end{bmatrix} \tag{10}$$

As shown in Figure 6 good initialization will lead to faster convergence.

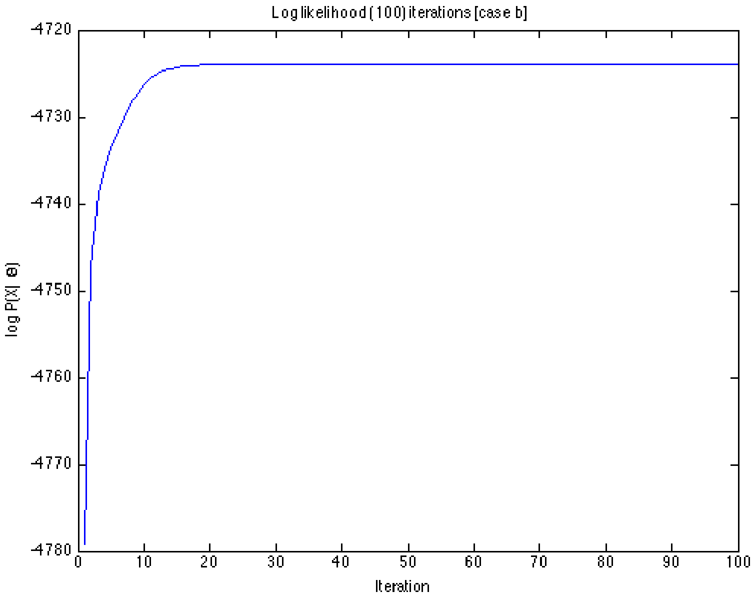

Figure 6: Log likelihood scores for case **b**

Problem 2. From the coin-tossing example discussed in class we know that there are two coins(A and B) equally likely to be selected at random to perform the tossing experiment which produced the following outcome.

	Coin A	Coin B
$\mathbf{x_1}$		5H, 5T
$\mathbf{x_2}$	9H, 1T	
$\mathbf{x_3}$	8H, 2T	
$\mathbf{x_4}$		4H, 6T
$\mathbf{x_5}$	7H, 3T	

And

$$P(z = A) = P(z = B) = 0.5$$

$$\theta_A = P(H|z = A), \quad \theta_B = P(H|z = B)$$

For this task we begin with $\theta^{(0)} = (\theta_A^{(0)}, \theta_B^{(0)}) = (0.6, 0.4)$ and terminate the EM procedure at the 10th iterations. We obtain the estimate of the parameters for $t = 1, 2, \cdots, 10$ as the following:

t	1	2	3	4	5	6	7	8	9	10
$\theta_A^{(t)}$	0.7261	0.7680	0.7852	0.7923	0.7951	0.7961	0.7965	0.7967	0.7968	0.7968
$\theta_B^{(t)}$	0.5020	0.5194	0.5208	0.5203	0.5199	0.5197	0.5196	0.5196	0.5196	0.5196

Now with the estimated parameters $\theta_A^{(t)}$ and $\theta_B^{(t)}$ we can compute the posterior probability $P(z^n = A|x^n; \theta^{(t)})$ and $P(z^n = B|x^n; \theta^{(t)})$ using Bayes rule:

$$P(z^n = A|x^n; \theta^{(t)}) = \frac{P(z^n = A) \cdot P(x^n|z^n = A; \theta^{(t)})}{P(x^n; \theta^{(t)})} \tag{11}$$

And $P(x^n; \theta^{(t)})$ can be obtained by marginalizing over z^n using the sum rule:

$$P(x^n; \theta^{(t)}) = P(z^n = A) \cdot P(x^n|z^n = A; \theta^{(t)}) + P(z^n = B) \cdot P(x^n|z^n = B; \theta^{(t)}) \tag{12}$$

	x^n	$P(z^n = A \vert x^n; \theta^{(t)})$	$P(z^n = B \vert x^n; \theta^{(t)})$
t=1	5H, 5T	0.2416	0.7584
	9H, 1T	0.9384	0.0616
	8H, 2T	0.8528	0.1472
	4H, 6T	0.1080	0.8920
	7H, 3T	0.6878	0.3122
t=10	5H, 5T	0.1030	0.8970
	9H, 1T	0.9520	0.0480
	8H, 2T	0.8455	0.1545
	4H, 6T	0.0307	0.9693
	7H, 3T	0.6015	0.3985

We get

$$\log P(X; \theta^{(t)}) = \sum_{n=1}^{5} \log P(x^n; \theta^{(t)}) \tag{13}$$

The plot of $\log P(X; \theta^{(t)})$ is shown in Figure 7.

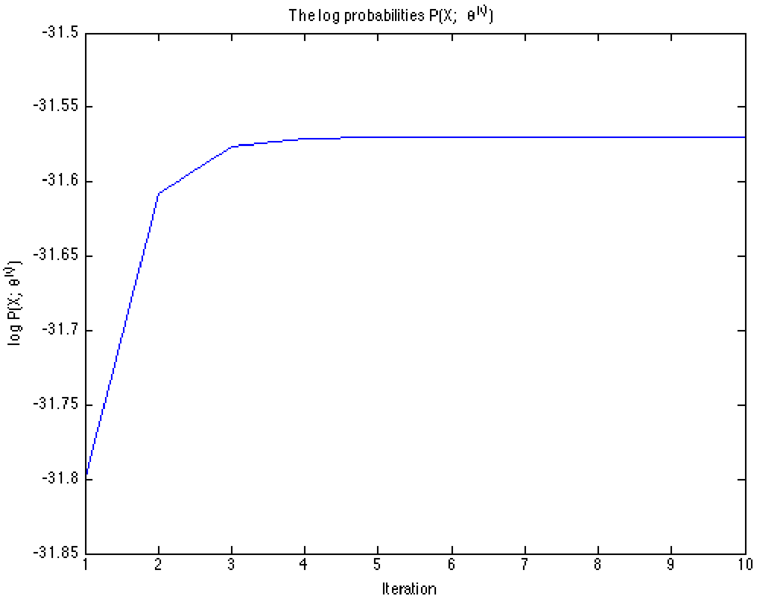

Figure 7: Plot of $\log P(X; \theta^{(t)})$

Problem 3. Derive the EM estimation for two-dimensional Gaussian Mixture Density parameters $\hat{\pi}_k, \hat{\mu}_k, \hat{\Sigma}_k$ for $k = 1, 2, \cdots, K$ given that $\Sigma_1 = \Sigma_2 = \cdots = \Sigma_K = \Sigma$. In EM algorithm we are maximizing $Q(\Theta, \Theta') = \mathbf{E}[\log P(\mathbf{X}, Z; \Theta) | \mathbf{X}, \Theta']$ with respect to π_k, μ_k, and Σ_k and according to the discussion in class and the constraints given here we get:

$$Q(\Theta, \Theta') = \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma'_{n,k} \left(\log \pi_k - \log 2\pi - \frac{1}{2} \log |\Sigma| - \frac{1}{2} (x_n - \mu_k)^T \Sigma^{-1} (x_n - \mu_k) \right) \tag{14}$$

Where $\gamma'_{n,k} = P(z_n = k | x_n; \Theta')$

$$\gamma'_{n,k} = \frac{\pi'_k N(x_n; \mu'_k, \Sigma')}{\sum_{j=1}^{K} \pi'_k N(x_n; \mu'_k, \Sigma')} \tag{15}$$

Then

$$Q_\mu = -\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma'_{n,k}(x_n-\mu_k)^T\Sigma^{-1}(x_n-\mu_k) \tag{16}$$

$$\frac{\partial Q_\mu}{\partial \mu_k} = \Sigma^{-1}\sum_{n=1}^{N}\gamma'_{n,k}(x_n-\mu_k) \tag{17}$$

$$\sum_{n=1}^{N}\gamma'_{n,k}x_n - \sum_{n=1}^{N}\gamma'_{n,k}\hat{\mu}_k = 0 \tag{18}$$

$$\hat{\mu}_k = \frac{1}{\sum_{n=1}^{N}\gamma'_{n,k}}\sum_{n=1}^{N}\gamma'_{n,k}x_n \tag{19}$$

And

$$Q_\Sigma = -\frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma'_{n,k}\left(\log|\Sigma| + (x_n-\mu_k)^T\Sigma^{-1}(x_n-\mu_k)\right) \tag{20}$$

$$= -\frac{1}{2}\left(\log|\Sigma|\sum_{n=1}^{N}\sum_{k=1}^{K}\gamma'_{n,k} + \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma'_{n,k}(x_n-\mu_k)^T\Sigma^{-1}(x_n-\mu_k)\right) \tag{21}$$

$$= -\frac{1}{2}\left(N\log|\Sigma| + \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma'_{n,k}(x_n-\mu_k)^T\Sigma^{-1}(x_n-\mu_k)\right) \tag{22}$$

$$\frac{\partial Q_\Sigma}{\partial \Sigma^{-1}} = -\frac{1}{2}\left(-N\Sigma + \sum_{n=1}^{N}(x_n-\mu_k)(x_n-\mu_k)^T\right) \tag{23}$$

$$-N\hat{\Sigma} + \sum_{n=1}^{N}(x_n-\mu_k)(x_n-\mu_k)^T = 0 \tag{24}$$

$$\hat{\Sigma} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_k)(x_n-\mu_k)^T \tag{25}$$

And

$$Q_\pi = \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma'_{n,k}\log\pi_k + \lambda(\sum_{k=1}^{K}\pi_k - 1) \tag{26}$$

$$\frac{\partial Q_\pi}{\partial \pi_k} = \sum_{n=1}^{N}\gamma'_{n,k}\frac{1}{\pi_k} + \lambda \tag{27}$$

$$\sum_{n=1}^{N}\gamma'_{n,k} + \lambda\hat{\pi}_k = 0, \quad \sum_{n=1}^{N}\sum_{k=1}^{K}\gamma'_{n,k} + \lambda\sum_{k=1}^{K}\pi_k = 0 \tag{28}$$

$$\hat{\pi}_k = \frac{1}{N}\sum_{n=1}^{N}\gamma'_{n,k} \tag{29}$$

Appendix:

assignment_1.m

```
% -----------------------------------------------------------
% CS 8735: Supervised Learning Fall (2015)
%        Unversity of Missouri-Columbia
%               Chanmann Lim
%              September 2015
% -----------------------------------------------------------
clc;
clear;
close all;

%% Problem 1
% Load data
X = load('GMD.dat');

% EM algorithm
problem_1_a
problem_1_b
problem_2
```

problem_1_a.m

```
T = 100; % 100 iterations
% Initialization
prior = 1/4 * ones(1, 4);
Mu = [ [10; 2], [5; 6], [0; 1], [4; 3] ];
Sigma = [[1; 0; 1], [1; 0; 1], [1; 0; 1], [1; 0; 1] ];

[Prior, MU, SIGMA, scores] = EM(X, T, prior, Mu, Sigma);

% Estimated parameters
display('Final estimated parameter:');
display(Prior{T});
display(MU{T});
display(SIGMA{T});

% Plot of log likelihood scores
figure;
plot(1:T, scores);
title(['Log likelihood (' num2str(T) ') iterations']);
xlabel('Iteration');
ylabel('log P(X|\Theta)');

% classification
for t=[10 50 100]
    [~, K] = size(Prior{t}); % The number of components assumed
    k = classify(1:K, X, Prior{t}, MU{t}, SIGMA{t});
    clusters_plot(X, k, t);
end
```

problem_1_b.m

```
T = 100; % 100 iterations
% Initialization
prior = [0.25 0.2 0.25 0.3];
Mu = [ [1; 2], [4; 8], [8; 6.5], [13.5; 3] ];
Sigma = [[1; 0; 1], [1; 0; 1], [1; 0; 1], [1; 0; 1] ];

[Prior, MU, SIGMA, scores] = EM(X, T, prior, Mu, Sigma);

% Estimated parameters
display('Second initialization:');
display('Final estimated parameter:');
display(Prior{T});
display(MU{T});
display(SIGMA{T});

% Plot of log likelihood scores
figure;
plot(1:T, scores);
```

```
title(['Log likelihood (' num2str(T) ') iterations [case b]']);
xlabel('Iteration');
ylabel('log P(X|\Theta)');
```

EM.m

```
function [ Prior, MU, SIGMA, scores ] = EM( X, T, prior, Mu, Sigma )
%EM - run EM algorithm for T iterations

[~, K] = size(prior);
[N, d] = size(X);
% Theta(t=1..T)
Prior = cell(1, T);
MU = cell(1, T);
SIGMA = cell(1, T);
% Log likelihood scores
scores = zeros(1, T);

t = 0;
while t < T
    Gamma = gamma_nk(X, prior, Mu, Sigma);
    for k=1:K
        % Expectation step
        g = Gamma(:,k) ./ sum(Gamma, 2);
        Nk = sum(g);

        % Maximization step
        Mu(:,k) = 1/Nk * X' * g;
        X_tilde = X' - Mu(:,k)*ones(1,N);
        Sigma(:,k) = vectorize_sigma( 1/Nk * (ones(d,1)*g' .* X_tilde * X_tilde') );
        prior(k) = Nk / N;
    end

    % Check for convergence
    % We're assuming that EM algorithm will converge in T iteration
    t = t + 1;
    % Store Theta(t=1..T)
    Prior{t} = prior;
    MU{t} = Mu;
    SIGMA{t} = Sigma;

    scores(t) = log_P(X, prior, Mu, Sigma);
end
```

gamma_nk.m

```
function [ Gamma ] = gamma_nk( X, prior, mu, Sigma )
% GAMMA_NK - Compute gamma n,k for all K in the E-Step of EM algorithm
%            is defined as P(z_n = k|x_n, Theta)
%   where
%       Theta =< prior, mu, Sigma >

    [~, K] = size(prior);
    [N, d] = size(X);
    Gamma = zeros(N, K);
    for k=1:K
        S = sigma_d(Sigma(:,k), d);
        Gamma(:, k) = prior(k) * mvnpdf(X, mu(:,k), S);
    end
end
```

mvnpdf.m

```
function [ y ] = mvnpdf( X, mu, Sigma )
% NORMAL - Multivariate normal density N(x; mu, Sigma)

    [N, d] = size(X);

    denominator = sqrt((2*pi)^d*det(Sigma));
    X_tilde = X' - mu * ones(1, N);
    y = 1/denominator * exp(-0.5 .* diag(X_tilde'/Sigma*X_tilde));
end
```

log_P.m

```
function [ score ] = log_P( X, prior, Mu, Sigma )
% LOG_P( X, prior, Mu, Sigma ) - Compute the log likelihood scores
%   log P( X| Theta ).

[~, K] = size(prior);
[N, d] = size(X);
P = zeros(N, K);

for k=1:K
    S = sigma_d(Sigma(:,k), d);
    P(:,k) = prior(k) * mvnpdf(X, Mu(:,k), S);
end
score = sum( log( sum(P, 2)));
```

sigma_d.m

```
function [ Sigma ] = sigma_d ( v, d )
% SIGMA_D( v, d ) - Convert a vector into d * d symmetric matrix

if d*(d+1)/2 ~= length(v)
    error('The␣required␣elements␣mismatch␣with␣the␣dimensionalty.');
end

Sigma = zeros(d, d);

index = 1;
for i=1:d
    for j=i:d
        Sigma(i, j) = v(index);
        index = index + 1;
    end
end

Sigma = Sigma + triu(Sigma, 1)';
```

vectorize_sigma.m

```
function [ v ] = vectorize_sigma( Sigma )
% VECTORIZE_SIGMA( Sigma ) - Vectorize covariance \Sigma for
%   memory efficiency.

% Get upper-triangle
S = triu(Sigma);
% Vectorize matrix S
v = S(:);
% Remove all zeros from v
v(v==0) = [];
```

classify.m

```
function [ k ] = classify( K, X, prior, Mu, Sigma )
% CLASSIFY - Hard boundary classification for X
%   so that each data point is belong to only one class.
%   Find k* = argmax_k P(z_n = k|x_n; \Theta').

[N, d] = size(X);
P = zeros(N, length(K));

for j=K
    S = sigma_d( Sigma(:,j), d );
    P(:,j) = prior(j) * mvnpdf(X, Mu(:,j), S);
end
% row-based max
[~, k] = max(P, [], 2);
```

clusters_plot.m

```
function clusters_plot( X, k, t )
% CLUSTERS_PLOT - Plot of clusters in X
%   Where
%       X - dataset
%       k - clusters
%        t - t variable for the plot title

colors = 'bgrm';
figure;
hold on;
for j=unique(k)'
    x1 = X(:,1); x1 = x1(k == j);
    x2 = X(:,2); x2 = x2(k == j);
    scatter(x1, x2, 'filled', colors(j));
end
hold off;
title(['Plot␣of␣clusters␣at␣t=' num2str(t)]);
xlabel('X1');
ylabel('X2');
```

problem_2.m

```
% Observations: X = [x_1 ... x_n]', x_n = [n_H, n_T]
X = [5 5; 9 1; 8 2; 4 6; 7 3];
T = 10; % 10 iterations
[N, ~] = size(X);

% Initialization
% theta = [theta_A, theta_B]
theta_0 = [0.6 0.4];
prior = 0.5;
[Theta, P] = EM_2(X, T, prior, theta_0);

% Learned parameters t=1..10
display(Theta);

% Posterior probabilities
for t=[1 10]
    p = P{t};
    p_zA = p(:,1) ./ sum(p, 2);
    p_zB = p(:,2) ./ sum(p, 2);

    display(['Posterior␣prob.␣at␣t=' num2str(t)]);
    display(p_zA);
    display(p_zB);
end

% log P(X; theta') for t=1..10
p_x = zeros(N, T);
for t=1:T
    p_x(:,t) = sum(P{t}, 2);
end

log_p_X = sum( log(p_x) );
display(log_p_X);

figure;
plot(1:T, log_p_X);
title('The␣log␣probabilities␣P(X;\theta^{(t)})');
xlabel('Iteration');
ylabel('log␣P(X;\theta^{(t)})');
```

EM_2.m

```
function [ Theta, P ] = EM_2( X, T, prior, theta )
% EM_2 - EM procedure for problem 2

% Paramaters for t = 1..T
Theta = zeros(T, length(theta));
% P(z,x;theta) - the joint probability of z and x
P = cell(1, T);

t = 0;
while t < T
    % Expectation
    g_A = 1./(1 + bernoulli(X, theta(2)) ./ bernoulli(X, theta(1)));
    g_B = 1./(1 + bernoulli(X, theta(1)) ./ bernoulli(X, theta(2)));
    % Maximization
    theta(1) = 1/10 * 1/sum(g_A) * g_A' * X(:,1);
    theta(2) = 1/10 * 1/sum(g_B) * g_B' * X(:,1);

    % Check for convergence
    % We're assuming that EM algorithm will converge in T iteration
    t = t + 1;

    % Store Theta(t=1..T)
    Theta(t,:) = theta;
    P{t} = prior * [bernoulli(X, theta(1)), bernoulli(X, theta(2))];
end
```

bernoulli.m

```
function [ p ] = bernoulli( X, theta )
% BERNOULLI - Bernoulli distribution density function

p = theta.^X(:,1) .* (1-theta).^X(:,2);
```