CS 8735: Report for assignment 1

Chanmann Lim

September 17, 2015

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 100^{th} iterations.

The Matlab code for the experiment is in the **Appendix** section.

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

$$\hat{\pi}_1 = 0.3457, \hat{\pi}_2 = 0.1401, \hat{\pi}_3 = 0.1847, \hat{\pi}_4 = 0.3295 \tag{1}$$

$$\hat{\mathbf{U}} = \begin{bmatrix}\hat{\mu}_1 & \hat{\mu}_2 & \hat{\mu}_3 & \hat{\mu}_4\end{bmatrix} \tag{2}$$
$$= \begin{bmatrix}13.0263 & 4.0619 & 1.6026 & 6.9183 \\ 3.0455 & 7.9674 & 1.5717 & 5.9843\end{bmatrix} \tag{3}$$

$$\hat{\mathbf{\Sigma}} = \begin{bmatrix}\hat{\Sigma}_1 & \hat{\Sigma}_2 & \hat{\Sigma}_3 & \hat{\Sigma}_4\end{bmatrix} \tag{4}$$
$$= \begin{bmatrix}1.6470 & 0.8788 & 8.4468 & 6.2731 \\ -0.7471 & 0.2342 & -0.0635 & 2.6295 \\ 2.0688 & 1.1568 & 1.0938 & 1.9615\end{bmatrix} \tag{5}$$

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 at around the 80^{th} iteration.

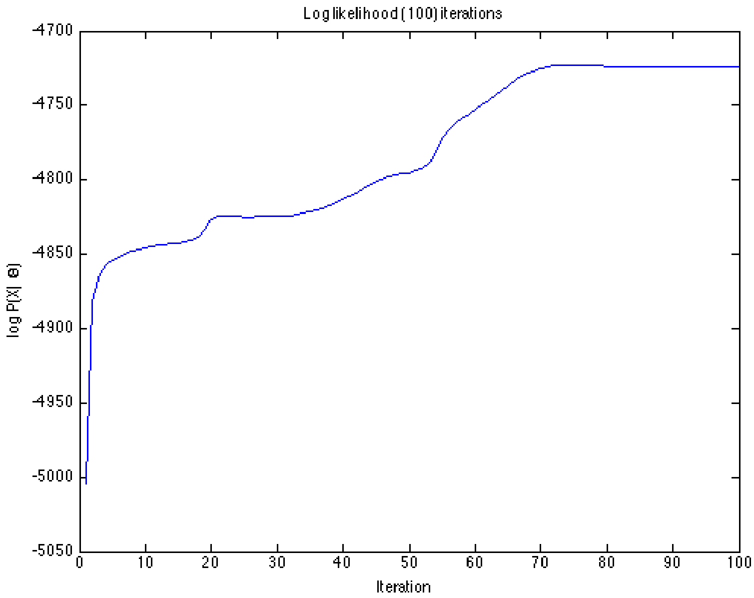

Figure 1: Log likelihood scores for case **a**

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 $t = 10, 50, 100$.

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

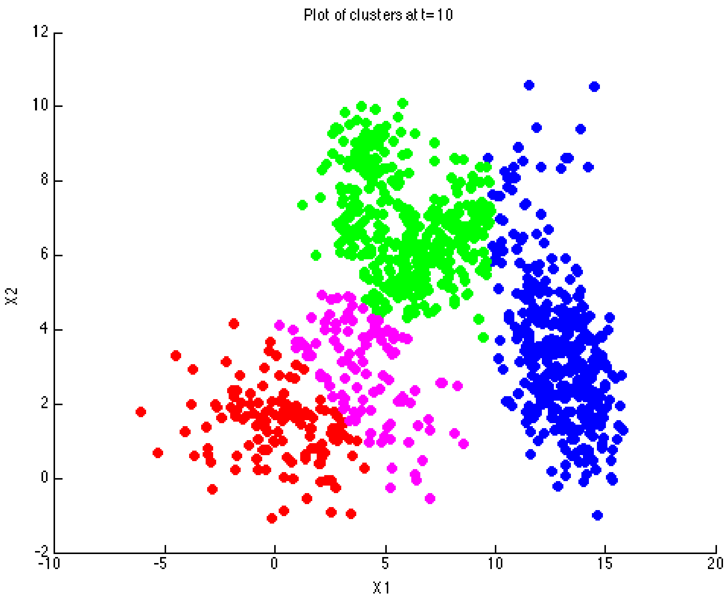

Figure 2: Plot of the four clusters at t=10

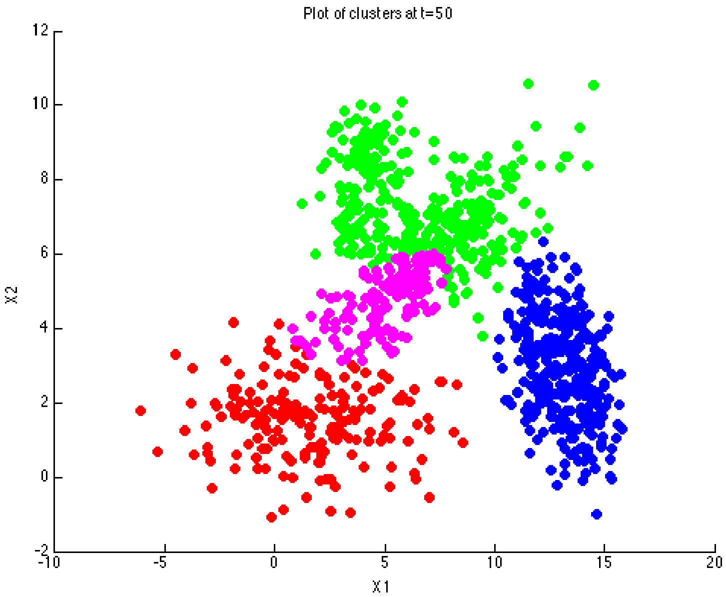

Figure 3: Plot of the four clusters at t=50

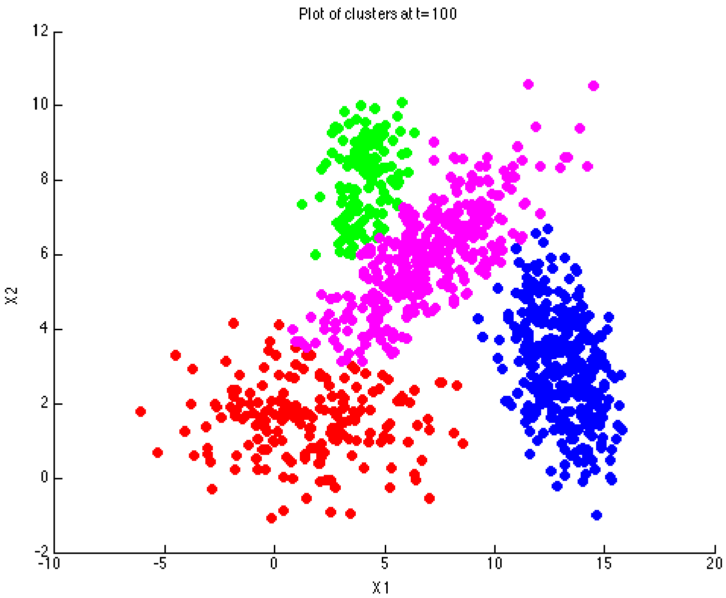

Figure 4: Plot of the four clusters at t=100

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.

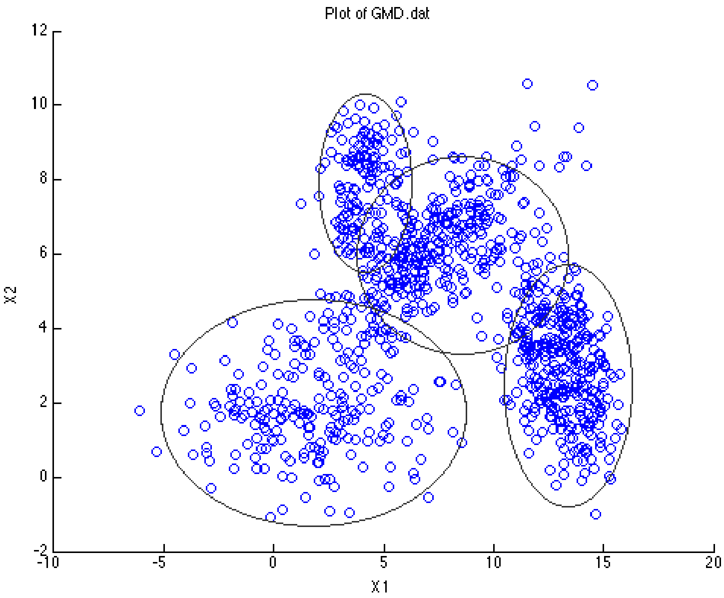

Figure 5: Plot of GMD.dat

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

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

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

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

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

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

$$\begin{aligned} \hat{\mathbf{U}} &= \begin{bmatrix} \hat{\mu}_1 & \hat{\mu}_2 & \hat{\mu}_3 & \hat{\mu}_4 \end{bmatrix} && (7) \\ &= \begin{bmatrix} 1.6026 & 4.0619 & 6.9182 & 13.0263 \\ 1.5717 & 7.9675 & 5.9843 & 3.0455 \end{bmatrix} && (8) \end{aligned}$$

$$\begin{aligned} \hat{\mathbf{\Sigma}} &= \begin{bmatrix} \hat{\Sigma}_1 & \hat{\Sigma}_2 & \hat{\Sigma}_3 & \hat{\Sigma}_4 \end{bmatrix} && (9) \\ &= \begin{bmatrix} 8.4468 & 0.8788 & 6.2733 & 1.6470 \\ -0.0635 & 0.2342 & 2.6295 & -0.7471 \\ 1.0938 & 1.1568 & 1.9615 & 2.0688 \end{bmatrix} && (10) \end{aligned}$$

Problem 2.

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_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(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(Prior{T});
display(MU{T});
display(SIGMA{T});
```

EM.m

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

[~, K] = size(prior);
[N, ~] = 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
    for k=1:K
        % Expectation step
        g = gamma_nk(X, k, prior, Mu, Sigma);
        Nk = sum(g);

        % Maximization step
        Mu(:,k) = 1/Nk * sum(g*ones(1, 2) .* X)';
        X_tilde = X' - Mu(:,k)*ones(1,N);
        Sigma(:,k) = vectorize_sigma( 1/Nk * (ones(2,1)*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 [ g ] = gamma_nk( X, k_i, prior, mu, Sigma )
% GAMMA_NK - gamma n,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);
    denominators = zeros(N, K);
    for k=1:K
        S = sigma_d(Sigma(:,k), d);
        denominators(:, k) = prior(k) * mvnpdf(X, mu(:,k), S);
    end
    g = denominators(:, k_i) ./ sum(denominators, 2);
end
```

mvnpdf.m

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

    [N, d] = size(X);
    y = zeros(N, 1);
    denominator = sqrt((2*pi)^d*det(Sigma));
    for n=1:N
        x = X(n, :)';
        x_tilde = x - mu;
        y(n) = 1/denominator * exp(-0.5 * x_tilde'/Sigma*x_tilde);
    end
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');
```