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.

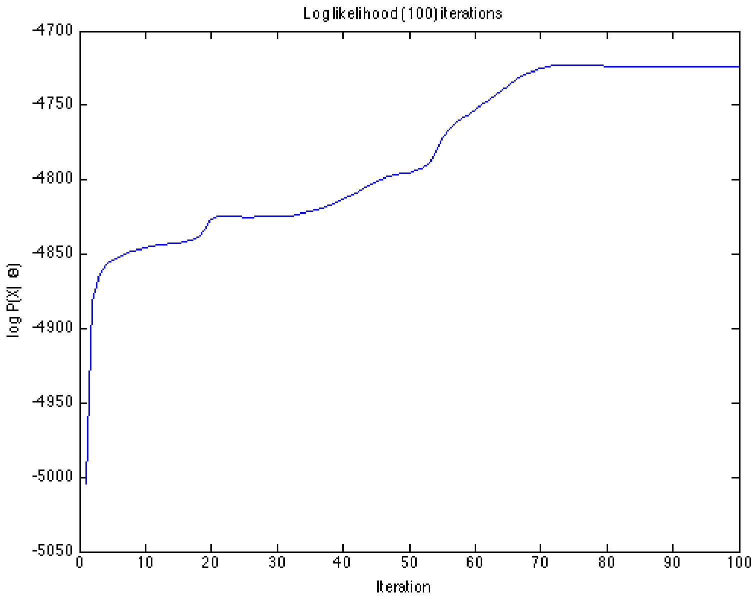

Figure 1: Log likelihood scores for case **a**

Appendix:

problem_1.m

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

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

%% EM algorithm
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] ];

tic;
[prior, Mu, Sigma, scores] = EM(X, T, prior, Mu, Sigma);
toc

% Estimated parameters
display(prior);
display(Mu);
display(Sigma);

% Plot of log likelihood scores
figure;
plot(1:T, scores);
title(['Log likelihood (' num2str(T) ') iterations']);
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, ~] = size(X);
% 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;
    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
```