Table of Contents

Homework 3, ECE283, Morten Lie. Collaboration with Sondre Kongsgård, Brage Sæther, Anders Vagle

```
clear all;
close all;
clc;
addpath('Functions');
```

Comments Problem 1

Looking at the emprical probabilities, we see that the K-means algorithm separates the dataset into two clusters with ease. The cetainty for a specific is assigned to the designated cluster with a probability higher than 0.8. Howerver, when we increase the number of clusters to be devided into, we observe that certain parts of the clusters in the dataset gets spread more out over the new clusters. Especially cluster 1 gets spread pretty uniformly out over two of the new clusters. The more clusters we allow for the K-means algorithm, the more uncertain is the probability for a dataset from a specific cluster to land in a new cluster. Much of the reason why this is is due to the dataset being very tight with a pretty similar mean value between the mixed gaussians.

Component definitions

```
N = 200;

% Component 1
theta_1 = 0;
```

```
m_1 = [0 0]';
pi_1 = 1/2;
lambda_1 = 2;
lambda_2 = 1;
u_1 = [cos(theta_1) sin(theta_1)]';
u_2 = [-sin(theta_1) cos(theta_1)]';
C_1 = [u_1 u_2]*diag([lambda_1,lambda_2])*inv([u_1 u_2]);

% Component 2
theta2_2 = -3*pi/4;
m_2 = [-2 1]';
pi_2 = 1/6;
lambda_a1 = 2;
lambda_a2 = 1/4;
u1_2 = [cos(theta2_2) sin(theta2_2)]';
u2_2 = [-sin(theta2_2) cos(theta2_2)]';
C_2 = [u1_2 u2_2]*diag([lambda_a1,lambda_a2])*inv([u1_2 u2_2]);

% Component 3
theta2_3 = pi/4;
m_3 = [3 2]';
pi_3 = 1/3;
lambda_b1 = 3;
lambda_b2 = 1;
u1_3 = [cos(theta2_3) sin(theta2_3)]';
u2_3 = [-sin(theta2_3) cos(theta2_3)]';
C_3 = [u1_3 u2_3]*diag([lambda_b1,lambda_b2])*inv([u1_3 u2_3]);


X = zeros(N,2);
Z = zeros(N,3);
for i = 1:N
    a = rand();
    if a < pi_1
        X(i,:) = mvnrnd(m_1,C_1);
        Z(i,:) = [1 0 0];
    elseif a < pi_1 + pi_2
        X(i,:) = mvnrnd(m_2,C_2);
        Z(i,:) = [0 1 0];
    else
        X(i,:) = mvnrnd(m_3,C_3);
        Z(i,:) = [0 0 1];
    end
end
```

K-means

```
N_rand_inits = 5;
K_max = 5;
m_opt = zeros(K_max,2,K_max);
C_opt = zeros(N,K_max);
for K = 2:K_max
    min_sme = inf;
```

```
    for i = 1:N_rand_inits
        C = randi(K,N,1);
        [m,C] = k_means(N,K,C,X);
        sme = SME(m,X,C);
        if sme < min_sme
            m_opt(1:K,:,K) = m;
            C_opt(:,K) = C;
            min_sme = sme;
        end
    end
end

% One-hot encoding
a = zeros(N,K_max,K_max);
for K = 2:K_max
    for i = 1:N
        a(i,C_opt(i,K),K) = 1;
    end
end
```

Generate empirical probability table

```
pk_kmeans = zeros(3,K_max,K_max);
for K = 2:K_max
    for l = 1:3
        for k = 1:K
            num_k_l = 0;
            num_l = 0;
            for i = 1:N
                if Z(i,l) == 1
                    num_l = num_l + 1;
                    if a(i,k,K) == 1
                        num_k_l = num_k_l + 1;
                    end
                end
            end
            pk_kmeans(l,k,K) = num_k_l/num_l;
        end
    end
end
plot_table(pk_kmeans,K_max,'K-means');
```

```
Plotting table of P(k|i) generated from K-means with K = 2

ans =

    0.1522    0.8478
         0    1.0000
    0.9000    0.1000

Plotting table of P(k|i) generated from K-means with K = 3

ans =
```

```
    0.3696    0.6304         0
    0.9792    0.0208         0
         0    0.4167    0.5833

Plotting table of P(k|i) generated from K-means with K = 4

ans =

    0.2935    0.5543    0.0326    0.1196
    0.7708         0         0    0.2292
         0    0.1500    0.7000    0.1500

Plotting table of P(k|i) generated from K-means with K = 5

ans =

    0.2609    0.1087    0.0978    0.5326         0
    0.5417    0.4583         0         0         0
         0    0.0167    0.5667    0.0833    0.3333
```

Comments Problem 2

In the figure one may observe that the contours of the gaussian distributions generated with the EM algorithm fits the clusters made by the K-means algorithm very well. Also, upon expection of the average values of the probabilities, one may see that they are pretty close to the values we got from the previous excercise. Allthough a bit weaker probabilities, but one would be able to classify the different clusters equally based on the probabilities from the different tables generated by K-means and EM.

EM

```
m_EM = zeros(K_max,2,K_max);
C_EM = zeros(2,2,K_max, K_max);
pi_EM = zeros(K_max,K_max);
pk_EM = zeros(3,K_max,K_max);
for K = 2:K_max
    m_init = m_opt(:,:,K);
    [m_,C_,pi_,pk_] = EM(m_init,N,K,X,Z);
    m_EM(:,:,K) = m_;
    C_EM(:,:,1:K,K) = C_;
    pi_EM(1:K,K) = pi_;
    pk_EM(:,1:K,K) = pk_;
end
plot_table(pk_EM,K_max,'EM');
```

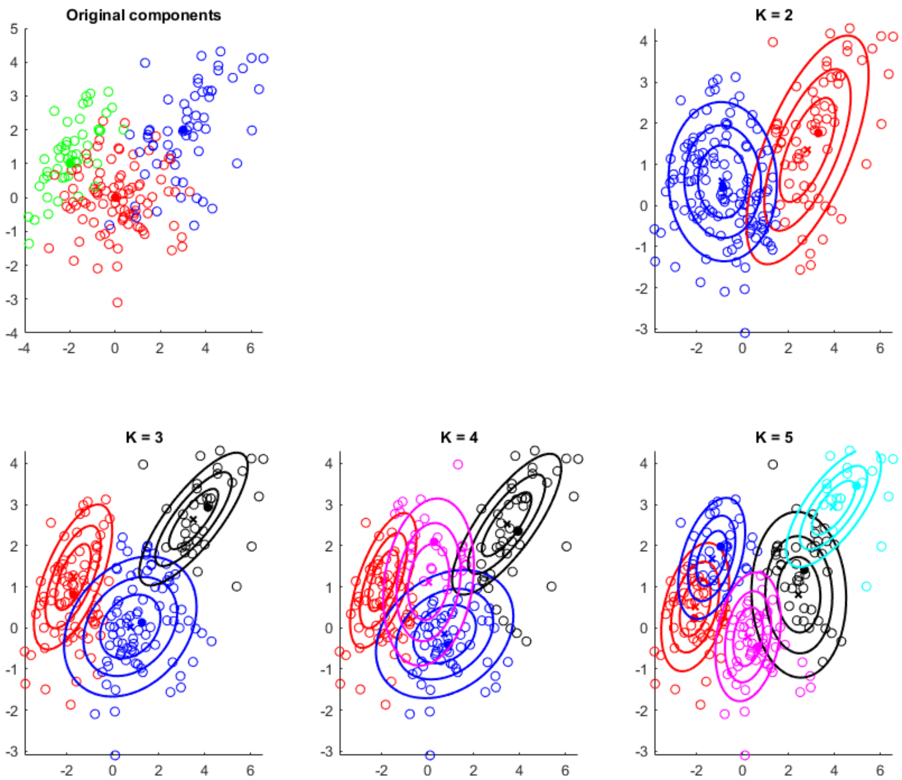
```
Plotting table of P(k|i) generated from EM with K = 2

ans =

    0.2694    0.7306
    0.0035    0.9965
```

```
    0.8592    0.1408

Plotting table of P(k|i) generated from EM with K = 3

ans =

    0.1652    0.8027    0.0321
    0.8820    0.1167    0.0014
    0.0191    0.4073    0.5735

Plotting table of P(k|i) generated from EM with K = 4

ans =

    0.1238    0.6801    0.0456    0.1505
    0.7943    0.0880    0.0012    0.1165
    0.0012    0.3031    0.6152    0.0806

Plotting table of P(k|i) generated from EM with K = 5

ans =

    0.1783    0.0769    0.1860    0.5494    0.0094
    0.5227    0.4397    0.0119    0.0254    0.0003
    0.0008    0.0055    0.4307    0.1024    0.4607
```

Comments Problem 3

One may observe in the figure that the means and the covariances for the example where we have K=3 clusters allign pretty well. For the other numbers of cluters, the means and the covariances allign well with the K-means algorithms.

```
plotData(X,Z,m_1,m_2,m_3,K_max,C_opt,m_opt,C_EM,m_EM,N);
```



Comments problem 4

A new vector is too correlated to another vector if the average absolute value of the product is greater than 0.2. (Based on experiements) The code is precented as follows from problem 5 and problem 6.

Generate a random vector u in d dimensions

```
d = 30;
u = zeros(d, 7);
for j = 1:7
    u(:,j) = generate_random_vector(d);
    while check_orthogonality(u,j) == 0
        u(:,j) = generate_random_vector(d);
    end
end
```

Generate d-dimensional data samples

```
sigma2 = 0.01;
N_d = 200;
[X_d, Z_d] = generate_sample_data(u,sigma2,N_d);
```

d-dimensional k-means

```
m_opt_d = zeros(K_max,size(X_d,2),K_max);
C_opt_d = zeros(N_d,K_max);
for K = 2:K_max
    min_sme = inf;
    for i = 1:N_rand_inits
        C_d = randi(K,N_d,1);
        [m_d,C_d] = k_means(N_d,K,C_d,X_d);
        sme = SME(m_d,X_d,C_d);
        if sme < min_sme
            m_opt_d(1:K,:,K) = m_d;
            C_opt_d(:,K) = C_d;
            min_sme = sme;
        end
    end
end
```

Generate empirical probability table for d-dimensional data

```
pk_kmeans_d = zeros(3,K_max,K_max);
for K = 2:K_max
    for l = 1:3
        for k = 1:K
            num_k_l = 0;
            num_l = 0;
            for i = 1:N_d
                if Z_d(i,l) == 1
                    num_l = num_l + 1;
                    if a(i,k,K) == 1
                        num_k_l = num_k_l + 1;
                    end
                end
            end
            pk_kmeans_d(l,k,K) = num_k_l/num_l;
        end
    end
end
plot_table(pk_kmeans_d,K_max,'d-dimensional K-means');
```

```
Plotting table of P(k|i) generated from d-dimensional K-means with K =
 2

ans =

    0.3649    0.6351
    0.2857    0.7143
    0.3651    0.6349

Plotting table of P(k|i) generated from d-dimensional K-means with K =
 3
```

```
ans =

    0.3919    0.3919    0.2162
    0.4286    0.4444    0.1270
    0.3968    0.4286    0.1746

Plotting table of P(k|i) generated from d-dimensional K-means with K =
 4

ans =

    0.3378    0.2703    0.2297    0.1622
    0.3175    0.3016    0.2063    0.1746
    0.3016    0.3333    0.2381    0.1270

Plotting table of P(k|i) generated from d-dimensional K-means with K =
 5

ans =

    0.2568    0.1622    0.1757    0.2568    0.1486
    0.2857    0.1587    0.2222    0.2857    0.0476
    0.2063    0.1746    0.2540    0.2698    0.0952
```

Comments problem 7

The cluster centers with d-dimentional dataset relates to the vector u in the way that the dimentions that u spans are also where the cluster centers are offset from the origin.

d-dimentional EM

```
m_EM_d = zeros(K_max,2,K_max);
C_EM_d = zeros(2,2,K_max, K_max);
pi_EM_d = zeros(K_max,K_max);
pk_EM_d = zeros(3,K_max,K_max);
for K = 2:K_max
    m_init = m_opt(:,:,K);
    [m_,C_,pi_,pk_] = EM(m_init,N,K,X,Z);
    m_EM_d(:,:,K) = m_;
    C_EM_d(:,:,1:K,K) = C_;
    pi_EM_d(1:K,K) = pi_;
    pk_EM_d(:,1:K,K) = pk_;
end
plot_table(pk_EM_d,K_max,'EM');
```

```
Plotting table of P(k|i) generated from EM with K = 2

ans =

    0.2694    0.7306
    0.0035    0.9965
```

```
    0.8592    0.1408

Plotting table of P(k|i) generated from EM with K = 3

ans =

    0.1652    0.8027    0.0321
    0.8820    0.1167    0.0014
    0.0191    0.4073    0.5735

Plotting table of P(k|i) generated from EM with K = 4

ans =

    0.1238    0.6801    0.0456    0.1505
    0.7943    0.0880    0.0012    0.1165
    0.0012    0.3031    0.6152    0.0806

Plotting table of P(k|i) generated from EM with K = 5

ans =

    0.1783    0.0769    0.1860    0.5494    0.0094
    0.5227    0.4397    0.0119    0.0254    0.0003
    0.0008    0.0055    0.4307    0.1024    0.4607
```

Comments problem 8

The EM algorithm is run for the same number of K's as earlier, and the eigen vectors of the covariance matrices relates to the mean and covariance of the gaussian mixtures such that higher mean and small variance for the gaussian mixtures yields smaller covariance from the EM algorithm.

Published with MATLAB® R2018a

```
function [ m,C ] = k_means( N,K,C,X )
convergence = 0;
m = zeros(K,size(X,2));
C_new = zeros(N,1);
while convergence == 0
    % Update mean
    for k = 1:K
        norm = 0;
        for i = 1:N
            if C(i) == k
                norm = norm + 1;
                m(k,:) = m(k,:) + X(i,:);
            end
        end
        m(k,:) = m(k,:)/norm;
    end

    % Update component vector C
    for i = 1:N
        min_dist = inf;
        for k = 1:K
            dist = sqrt((X(i,1)-m(k,1))^2 + (X(i,2)-m(k,2))^2);
            if dist < min_dist
                C_new(i) = k;
                min_dist = dist;
            end
        end
    end

    % Check for convergence
    if sum(abs(C_new - C),1) == 0
        convergence = 1;
    end
    C = C_new;
end
```

```
Not enough input arguments.

Error in k_means (line 3)
m = zeros(K,size(X,2));
```

Published with MATLAB® R2018a

Table of Contents


```
function [ m,C,pi,pk_ave ] = EM( m,N,K,X,Z )

max_iter = 20;
C = zeros(2,2,K);
for i = 1:K
    C(:,:,i) = eye(2);
end
pi = 1/K*ones(K,1);

for it = 1:max_iter
```

E-step

```
    pk = zeros(N,K);
    for i = 1:N
        x = X(i,:);
        piNormDist = zeros(K,1);
        for k = 1:K
            piNormDist(k) = pi(k)*mvnpdf(x,m(k,:),C(:,:,k));
        end
        for k = 1:K
            pk(i,k) = piNormDist(k)/sum(piNormDist(:),1);
        end
    end
```

M-step

```
    for k = 1:K
        sum_m = 0;
        sum_C = zeros(2,2);
        for i = 1:N
            sum_m = sum_m + pk(i,k) * X(i,:);
            sum_C = sum_C + pk(i,k) * (X(i,:)-m(k,:))'*(X(i,:)-
m(k,:));
        end
        m(k,:) = sum_m./sum(pk(:,k),1);
        C(:,:,k) = sum_C./sum(pk(:,k),1);
        pi(k) = sum(pk(:,k),1)/N;
    end

end
```

Not enough input arguments.

```
Error in EM (line 3)
C = zeros(2,2,K);
```

Calculate averages

```
pk_ave = zeros(3,K);
n = zeros(3,1);
for i = 1:N
    for l = 1:3
        if Z(i,l) == 1
            pk_ave(l,:) = pk_ave(l,:) + pk(i,:);
            n(l) = n(l) + 1;
        end

    end
end
pk_ave(1,:) = pk_ave(1,:)./n(1);
pk_ave(2,:) = pk_ave(2,:)./n(2);
pk_ave(3,:) = pk_ave(3,:)./n(3);
```

Published with MATLAB® R2018a