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

6. The goal of this problem is to investigate the influences of the ordering of the dataset in sequential clustering algorithm namely *Basic Sequential Algorithmic Scheme (BSAS)* and *Modified Basic Sequential Algorithmic Scheme (MBSAS)*. According to the literature, the main difference between BSAS and MBSAS is that the latter separates the clustering procedure into two parts. Firstly, the class determination phase is served as cluster discovery stage to find out the possible number of clusters with the constraints Θ : the distance threshold to subsume a data point to a cluster and q : the numbers of maximum allowed clusters. Secondly, the pattern classification stage where each point is being assigned to one of the already created clusters.

Plot of the vectors:

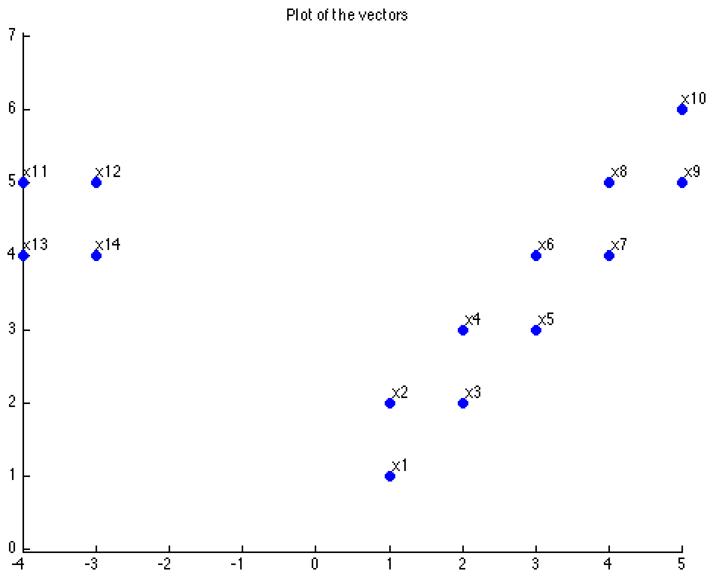

Figure 1: Plot of vectors

For case **a.** where the ordering of the dataset is $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{14}$ we got:

$$BSAS = \{\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}, \{\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\}, \{\mathbf{x}_7, \mathbf{x}_8, \mathbf{x}_9\}, \{\mathbf{x}_{10}\}, \{\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}\}\}$$
$$MBSAS = \{\{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_5\}, \{\mathbf{x}_7, \mathbf{x}_6, \mathbf{x}_8\}, \{\mathbf{x}_{10}, \mathbf{x}_9\}, \{\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}\}\}$$

The reason of the difference being MBSAS is trying to balance out the cluster assignment decision until all possible clusters are identified. This solves the problem with BSAS which decided a vector $\mathbf{x}$ to belong to an already created cluster or a new cluster prior to the final cluster formation.

For case **b.** where the ordering of the dataset is $\mathbf{x}_1, \mathbf{x}_{10}, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_{13}, \mathbf{x}_8, \mathbf{x}_{14}, \mathbf{x}_9$ and the clusters obtained:

$$BSAS = \{\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}, \{\mathbf{x}_{10}, \mathbf{x}_9\}, \{\mathbf{x}_4, \mathbf{x}_5, \mathbf{x}_6\}, \{\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}\}, \{\mathbf{x}_7, \mathbf{x}_8\}\}$$
$$MBSAS = \{\{\mathbf{x}_1, \mathbf{x}_2\}, \{\mathbf{x}_{10}, \mathbf{x}_9\}, \{\mathbf{x}_4, \mathbf{x}_3, \mathbf{x}_5\}, \{\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}\}, \{\mathbf{x}_7, \mathbf{x}_6, \mathbf{x}_8\}\}$$

For case **c.** where the ordering of the dataset is $\mathbf{x}_1, \mathbf{x}_{10}, \mathbf{x}_5, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_4, \mathbf{x}_6, \mathbf{x}_7, \mathbf{x}_{13}\mathbf{x}_{14}, \mathbf{x}_8, \mathbf{x}_9$ and the clusters obtained:

$$BSAS = \{\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}, \{\mathbf{x}_{10}, \mathbf{x}_9\}, \{\mathbf{x}_5, \mathbf{x}_4, \mathbf{x}_6\}, \{\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}\}, \{\mathbf{x}_7, \mathbf{x}_8\}\}$$
$$MBSAS = \{\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3\}, \{\mathbf{x}_{10}, \mathbf{x}_8, \mathbf{x}_9\}, \{\mathbf{x}_5, \mathbf{x}_4, \mathbf{x}_6, \mathbf{x}_7\}, \{\mathbf{x}_{11}, \mathbf{x}_{12}, \mathbf{x}_{13}, \mathbf{x}_{14}\}\}$$

In the last case MBSAS looked through the data and suggested only four clusters conversely BSAS purely depended on the ordering the data set and formed a total of five clusters.

7. In this task we are performing **k-mean** clustering with $k = 4$ on the GMD dataset from previous assignment. We started by initializing the centroids of all clusters randomly between -1 and 10 (empirically selected range and it doesn't carry out any knowledge-based implication) then run k-mean algorithm which consists of cluster assignment step in which a sample is assigned to closest cluster $C_{k_n^*}$.

$$k_n^* = \underset{k}{\operatorname{argmin}} \|x_n - \mu_k\|^2 \tag{1}$$

After all samples are assigned to clusters then update the cluster centroids by replacing it with the mean of the samples belong to each cluster k.

$$\hat{\mu_k} = \frac{1}{|C_k|} \sum_{x_n \in C_k} x_n \qquad ; k = 1, 2, 3, 4 \tag{2}$$

Finally, we compute the total distortion scores and check if it is unchanged(converged) then stop k-mean otherwise repeat the cluster assignment and centroid update until convergence.

7.1 The randomly initialization gave us:

$$U^{(0)} = \begin{bmatrix} \mu_{1,1} & \mu_{2,1} & \mu_{3,1} & \mu_{4,1} \\ \mu_{1,2} & \mu_{2,2} & \mu_{3,2} & \mu_{4,2} \end{bmatrix} \tag{3}$$

$$= \begin{bmatrix} 5.5742 & 8.8840 & 7.9968 & 7.1754 \\ 4.2734 & 9.2818 & 6.7980 & 8.8968 \end{bmatrix} \tag{4}$$

And the final cluster centroids is:

$$U = \begin{bmatrix} 1.3781 & 9.2459 & 13.1269 & 4.8156 \\ 1.7408 & 6.6790 & 2.8714 & 6.5621 \end{bmatrix} \tag{5}$$

7.2 The plot of samples with clusters:

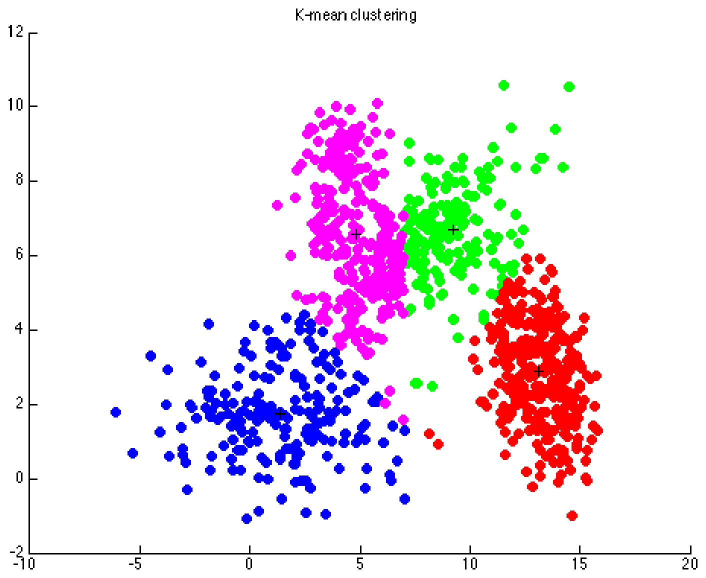

Figure 2: Plot of samples with clusters

7.3 The total distortion are [10154, 6105, 4869, 4675, 4652, 4648, 4647, 4647].

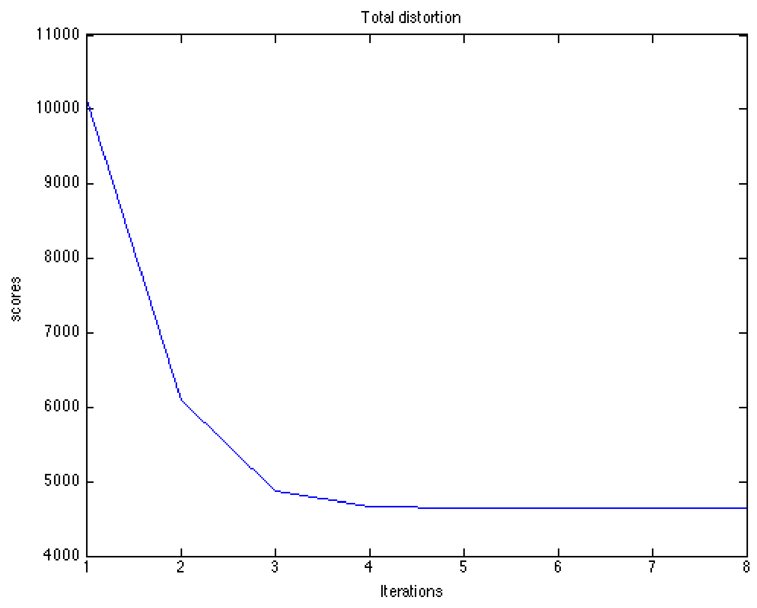

Figure 3: Total distortion

Epilogue: K-mean is simply a special case of EM algorithm with the assumption that the prior probability is uniform $\pi_k = \dfrac{1}{k}$ and all clusters have the same covariance matrix $\Sigma_k = \sigma^2 \cdot \mathbf{I} \quad ; k = 1, 2, ..., K$. Unlike EM it imposes hard clustering which make it converge faster and consume less computation than EM, yet k-mean and EM relay very strongly on good initialization and if k-mean somehow landed on a bad initialization it will also diverge faster than EM algorithm.

Appendix:

assignment_2.m

```
clc;
clear all;
close all;

problem_6
problem_7
```

problem_6.m

```
% load data
X = [[1 1]; [1 2]; [2 2]; [2 3]; [3 3]; [3 4]; [4 4]
     [4 5]; [5 5]; [5 6]; [-4 5]; [-3 5]; [-4 4]; [-3 4]];

% clustering parameters
q = 14;
Theta = sqrt(2);

% run clustering algorithm
% a.
seq = 1:14;
A = X(seq,:);
A_bsas = bsas(A, Theta, q);
A_mbsas = mbsas(A, Theta, q);
display(seq);
print_cluster(A_bsas, seq);
print_cluster(A_mbsas, seq);

% b.
seq = [1 10 2 3 4 11 12 5 6 7 13 8 14 9];
B = X(seq,:);
B_bsas = bsas(B, Theta, q);
B_mbsas = mbsas(B, Theta, q);
display(seq);
print_cluster(B_bsas, seq);
print_cluster(B_mbsas, seq);

% c.
seq = [1 10 5 2 3 11 12 4 6 7 13 14 8 9];
C = X(seq,:);
C_bsas = bsas(C, Theta, q);
C_mbsas = mbsas(C, Theta, q);
display(seq);
print_cluster(C_bsas, seq);
print_cluster(C_mbsas, seq);

% d. Plot the vectors
figure;
scatter(X(:,1), X(:,2), 'filled');
axis equal;
title('Plot␣of␣the␣vectors');

labels = num2str((1:size(X,1))','x%d');
text(X(:,1), X(:,2), labels, 'horizontal','left', 'vertical','bottom');
```

problem_7.m

```
% load data
X = load('../1/GMD.dat');

% initialization
[~, d] = size(X);
k = 4;
Mu = 11*rand(d, k) - 1;
display(Mu);

% run k-mean clustering
[I, Mu, D] = k_mean(X, Mu);

% print centroids and total distortion
```

```
display(Mu);
display(D);

% plot clusters
colors = 'bgrm';
figure;
hold on;
for i=1:k
    Xk = X(I==i,:);
    scatter(Xk(:,1), Xk(:,2), 'filled', colors(i));
end
plot(Mu(1,:), Mu(2,:), '+k');
hold off;
title('K-mean␣clustering');

% plot total distortion
figure;
plot(1:length(D), D);
title('Total␣distortion');
xlabel('Iterations');
ylabel('scores');
```

bsas.m

```
function [ C ] = bsas( X, Theta, q )
% bsas - Basic Sequential Clustering Algorithm
%
%       X       : sequential dataset
%       Theta   : clustering threshold
%       q       : maximum number of allowed clusters

[N,~] = size(X);

% numbers of cluster
m = 1;
C = {1};
% clusters representative
R = X(C{m},:);
for i=2:N
    [d, ck] = min_distance(X(i,:), R);
    if d > Theta && m < q
        m = m + 1;
        C{m} = i;
        R(m,:) = X(i,:);
    else
        nc = length(C{ck});
        C{ck} = [C{ck} i];
        % update representative
        % mc_new = (nc*mc+x) / (nc+1)
        R(ck,:) = (nc*R(ck,:) + X(i,:)) / (nc+1);
    end
end
```

mbsas.m

```
function [ C ] = mbsas( X, Theta, q )
% mbsas - Modified Basic Sequential Clustering Algorithm
%
%       X       : sequential dataset
%       Theta   : clustering threshold
%       q       : maximum number of allowed clusters

[N,~] = size(X);

% numbers of cluster
m = 1;
C = {1};
% clusters representative
R = X(C{m},:);

% class determination
for i=2:N
    [d,~] = min_distance(X(i,:), R);
    if d > Theta && m < q
        m = m + 1;
        C{m} = i;
        R(m,:) = X(i,:);
    end
end

% pattern classification
classified = cell2mat(C);
for i=1:N
   if ~nnz(classified == i)
       [~, ck] = min_distance(X(i,:), R);
       nc = length(C{ck});
       C{ck} = [C{ck} i];
       % update representative
       % mc_new = (nc*mc+x) / (nc+1)
       R(ck,:) = (nc*R(ck,:) + X(i,:)) / (nc+1);
   end
end
```

min_distance.m

```
function [ d, k ] = min_distance( x, C )
% distance - Compute min distance from x to all clusters = min d(x, C)
%   C - clusters representative

    [m,~] = size(C);
    D = zeros(1, m);
    for i=1:m
        D(i) = d2(x, C(i,:));
    end
    [d, k] = min(D);
end

% Euclidean distance between x and y
function [ d ] = d2(x, y)
    x_tilte = x - y;
    d = sqrt(x_tilte * x_tilte');
end
```

print_cluster.m

```
function print_cluster(C, seq)
% print_cluster - print cluster values
%   C   - set of clusters (cell array)
%   seq - sequence of x

disp([inputname(1) ' =']);
disp(' ');
m = length(C);
for i=1:m
    fprintf(['     {' sprintf(' x%d ', seq(C{i})) '}']);
end
disp(' ');
disp(' ');
```

k_mean.m

```
function [I, Mu, D] = k_mean( X, Mu )
% k_mean - run k-mean algorithm
%         X   - dataset
%         Mu  - clusters center (centroids)
%         D   - total distortion

    D = [];
    t = 1;

    while true
        I  = cluster_assignment(X, Mu);
        Mu = centroid_update(X, I);
        D(t) = distortion(X, Mu, I);
        if t > 1 && D(t) == D(t-1)
            break;
        end
        t = t + 1;
    end
end

function [ I ] = cluster_assignment( X, Mu )
% cluster_assignment in k-mean algorithm

    [~, m] = size(Mu);
    [N, ~] = size(X);
    D = zeros(N, m);
    for i=1:m
        mu = ones(N,1) * Mu(:,i)';
        D(:, i) = sum((X-mu).^2, 2);
    end
    [~, I] = min(D, [], 2);
end

function [ Mu ] = centroid_update( X, I )
% centroid_update in k-mean algorithm
%       X - dataset
%       I - clusters assignment

    [~, d] = size(X);
    K = unique(I)';
    Mu = zeros(d, length(K));

    for i=1:length(K)
        Mu(:,i) = mean(X(I == K(i), :))';
    end
end

function [ D ] = distortion( X, Mu, I )
% distortion - total distortion of the predicted clusters center
%       X  - dataset
%       Mu - clusters centroid
%       I  - clusters assignment

    [~, m] = size(Mu);
    [N, d] = size(X);

    S = zeros(N, d);
    for i=1:m
        S(I==i,:) = ones(nnz(I==i), 1) * Mu(:, i)';
    end
    D = sum( sum( (X-S).^2 ) );
end
```