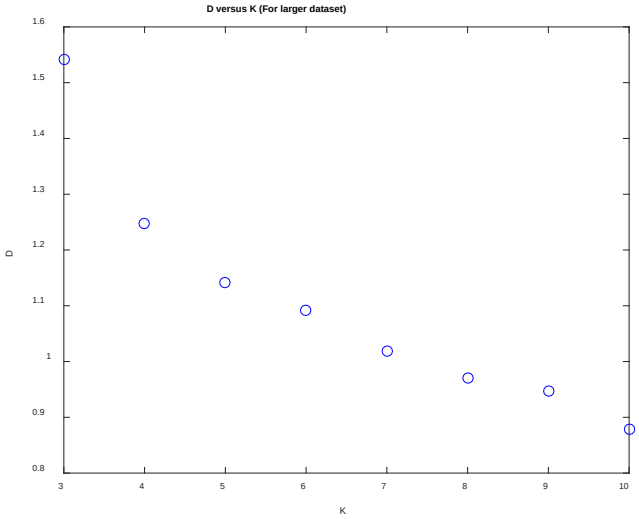
# Computational Assignment 2

Xiaotian Xie (xx5)

## I. K-means

1. Convergence: I selected the convergence criterion based on the iterates of *Y*, i.e., and $tol = 1*10^{-5}$.

1) "bigClusteringData.mat"




A. K = 3

I tried 17 initializations, among which the initialization with original centers: X1(2273,:), X2(237,:), X3(928,:) is the best since it only needs 11 iterations to converge. Besides, it is shown that the original centers are close to final centers.

C =[1;3;2;3;3;2;3;2;3;2;3;2;1;1;3;3;1;3;3;3;….] (first 20 entries)

B. K = 5

I tried 21 initializations, among which the initialization with original centers: X1(1145,:), X2(3284,:), X3(3174,:), X4(239,:) , X5(1578,:) is the best since it only needs 19 iterations to converge. Besides, it is shown that the original centers are close to final centers.

C =[5;2;3;2;2;3;2;3;2;3;2;4;5;5;5;2;4;1;2;2;4;….] (first 20 entries)

C. K = 7

I tried 18 initializations, among which the initialization with original centers: X1(3759,:), X2(3636,:), X3(209,:), X4(2915,:), X5(1064,:), X6(1671,:), X(2165,:) is the best since it only needs 20 iterations to converge. Besides, it is shown that the original centers are close to final centers.

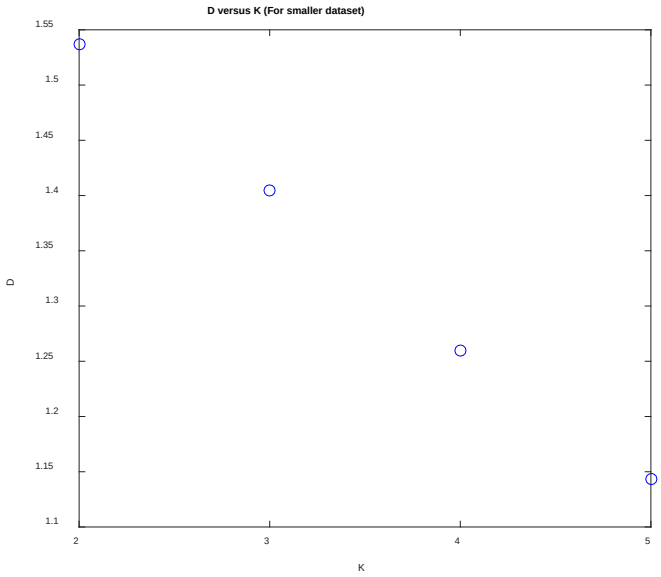
C =[4;1;6;1;3;6;7;6;1;6;1;3;5;5;1;7;2;1;1;7;….] (first 20 entries)

D. K = 9

I tried 18 initializations, among which the initialization with original centers: X1(3759,:), X2(3636,:), X3(209,:), X4(2915,:), X5(1064,:), X6(1671,:), X(2165,:) is the best since it only needs 20 iterations to converge. Besides, it is shown that the original centers are close to final centers.

C =[8;2;3;9;9;3;2;3;9;3;2;6;1;8;2;5;7;2;4;9;….] (first 20 entries)

2) “clustering.mat”




A. K = 2

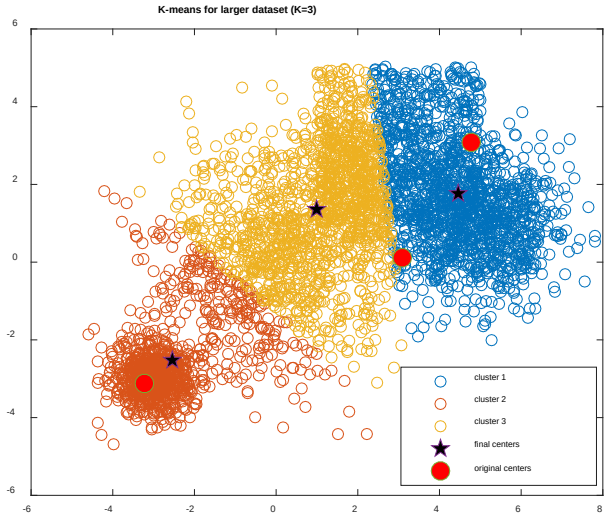
I tried 10 initializations, among which the initialization with original centers: X1(1538,:), X2(335,:) is the best since it only needs 5 iterations to converge. Besides, it is shown that the original centers are close to final centers.

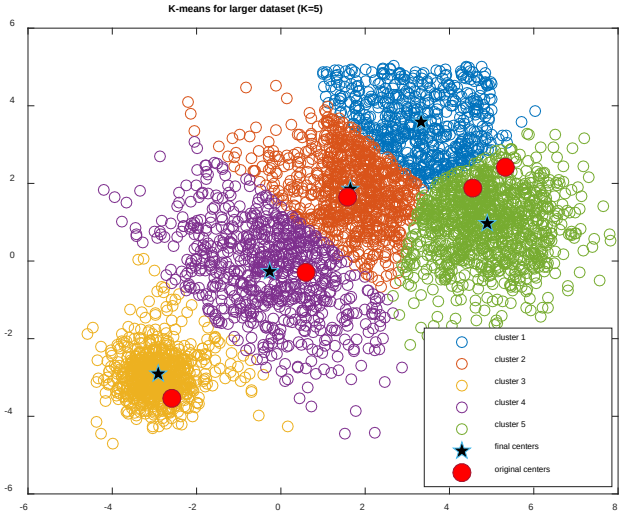
C =[2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;….] (first 20 entries)

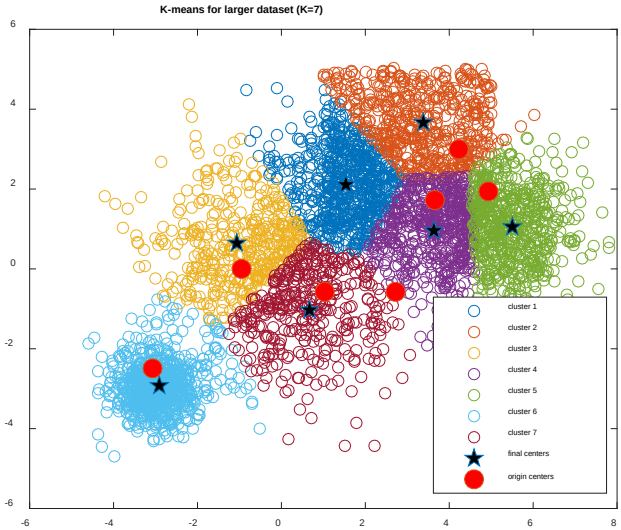
B. K = 4

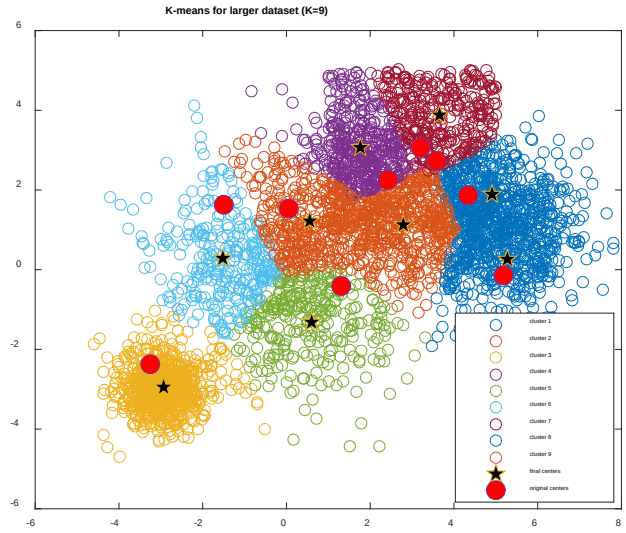
I tried 15 initializations, among which the initialization with original centers: X1(1603,:), X2(456,:), X3(997,:), X4(1802,:) is the best since it only needs 19 iterations to converge. Besides, it is shown that the original centers are close to final centers.

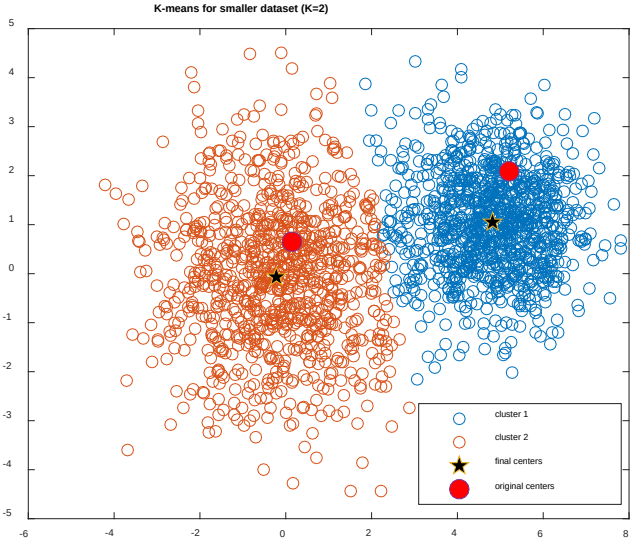
C =[2;2;3;2;3;3;3;2;3;2;3;2;3;4;2;3;3;2;2;3;….] (first 20 entries)

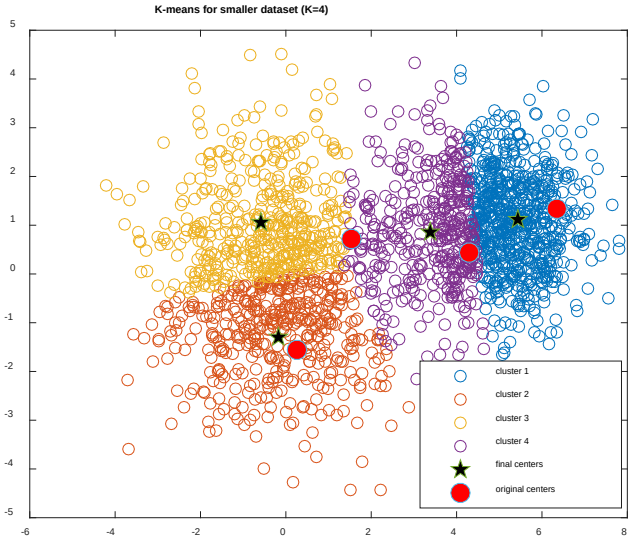

K-means for larger dataset (K=3)
cluster 1
cluster 2
cluster 3
final centers
original centers


K-means for larger dataset (K=5)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
final centers
original centers

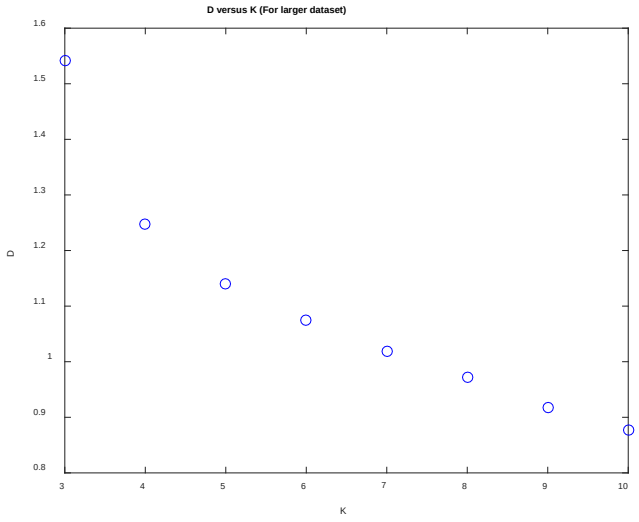

K-means for larger dataset (K=7)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
final centers
origin centers

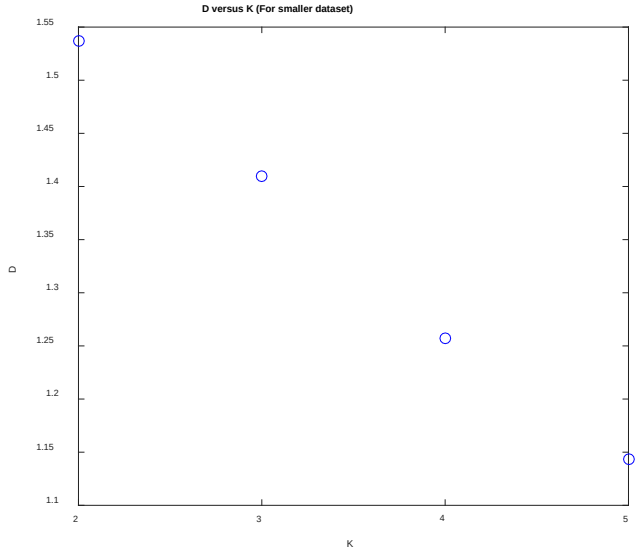

K-means for larger dataset (K=9)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
final centers
original centers


K-means for smaller dataset (K=2)
cluster 1
cluster 2
final centers
original centers


K-means for smaller dataset (K=4)
cluster 1
cluster 2
cluster 3
cluster 4
final centers
original centers

2. Convergence: I selected the convergence criterion based on the iterates of $Y$, i.e., and $tol = 1*10^{-7}$. Here I choose $1*10^{-7}$ to see if it could work better when $tol$ is smaller.
   The plots of D versus K for $tol = 1*10^{-7}$ are as follows:

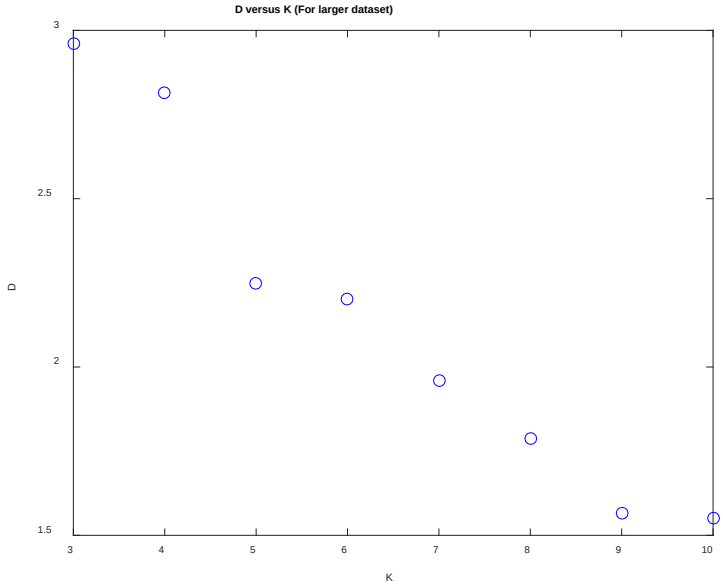


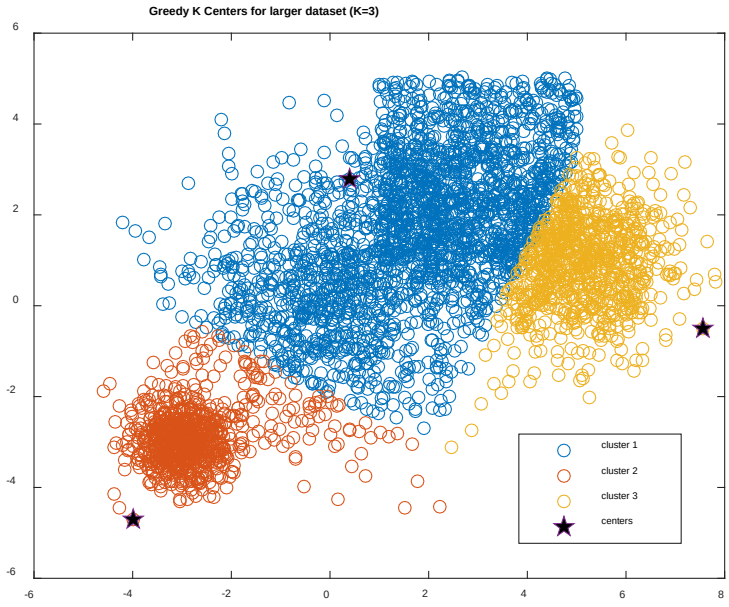

From the figures above, it is shown that these two $tol$ 's have no obvious difference.

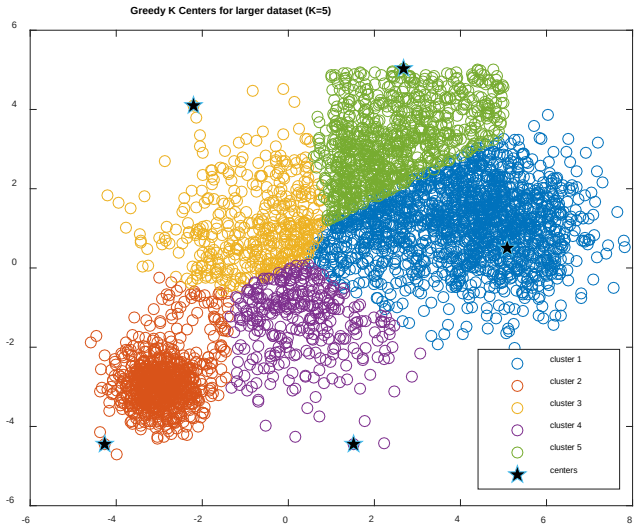
# II. Greedy K Centers & Single Swap

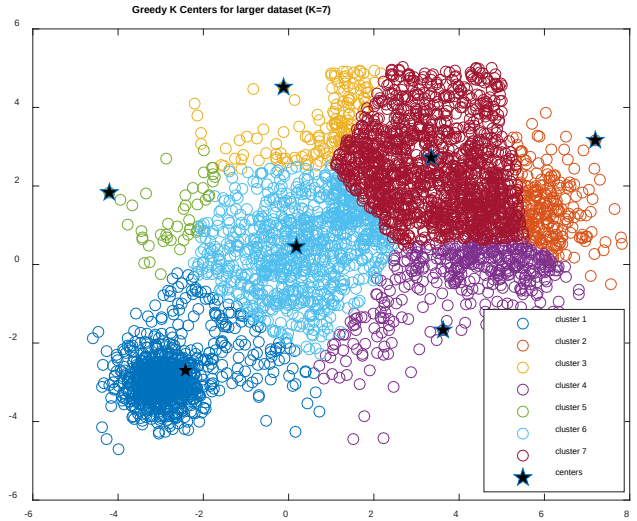
1) “bigClusteringData.mat”

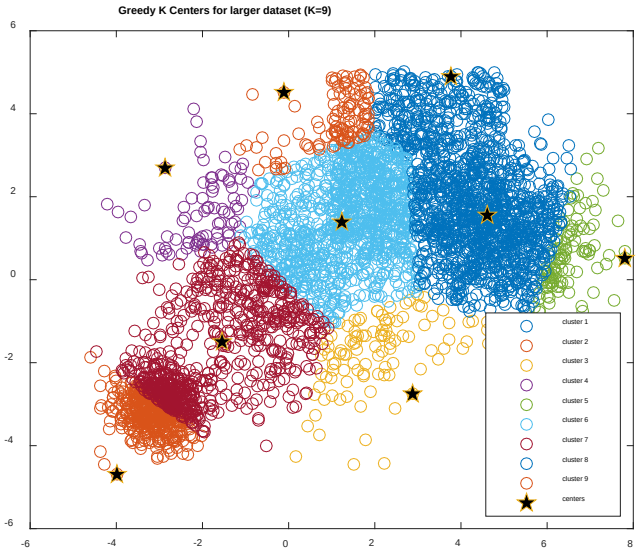



A. K = 3
   C1 = [3;1;2;1;1;2;1;2;1;2;1;1;3;3;1;1;1;1;1;1;…] (first 20 entries)
B. K = 5
   C1 = [1;1;2;1;3;2;1;2;1;2;1;3;1;1;1;4;5;1;5;4;… ] (first 20 entries)
C. K = 7
   C1 = [4;6;1;6;6;1;4;1;6;1;6;6;7;4;6;6;7;6;3;6;…] (first 20 entries)
D. K = 9
   C1 = [8;6;7;6;6;7;6;2;6;2;6;7;8;5;6;7;8;6;9;6;…] (first 20 entries)

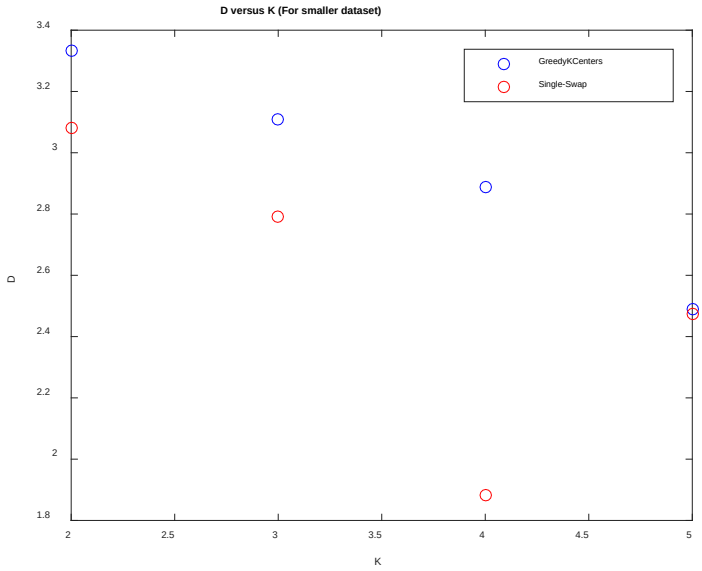



Greedy K Centers for larger dataset (K=5)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
centers


Greedy K Centers for larger dataset (K=7)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
centers


Greedy K Centers for larger dataset (K=9)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
centers

2) “clustering.mat”




C1 – C for “GreedyKCenters”  C2 – C for “Single-Swap”

A. K = 2

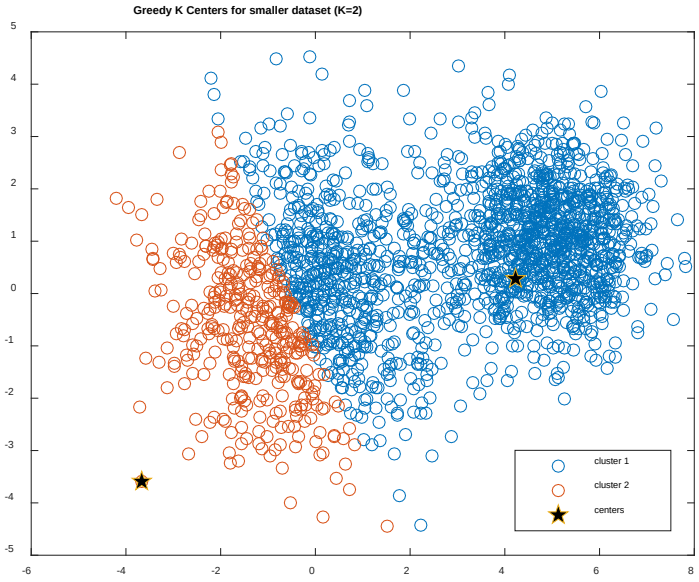
C1 = [1;2;2;2;1;1;1;2;1;1;1;1;1;2;1;1;1;2;1;…] (first 20 entries)

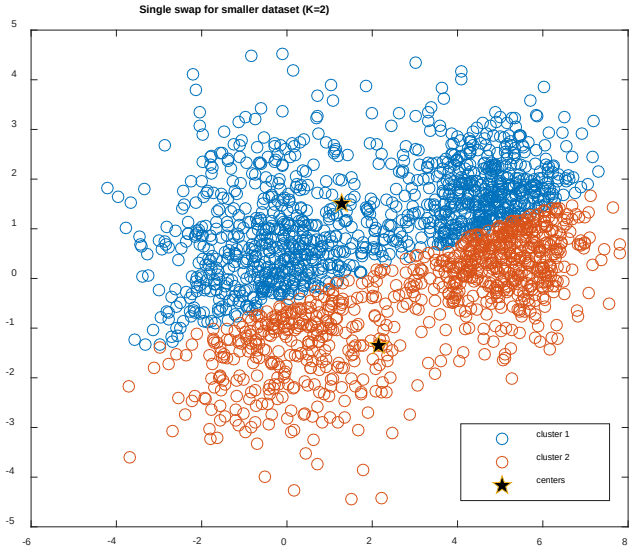
C2 = [2;1;1;2;1;1;1;2;1;1;1;2;1;1;2;1;1;2;1;…] (first 20 entries)

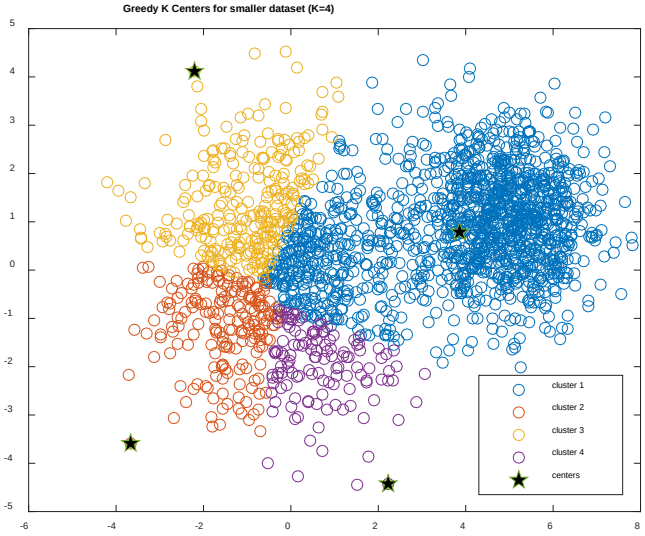
B. K = 4

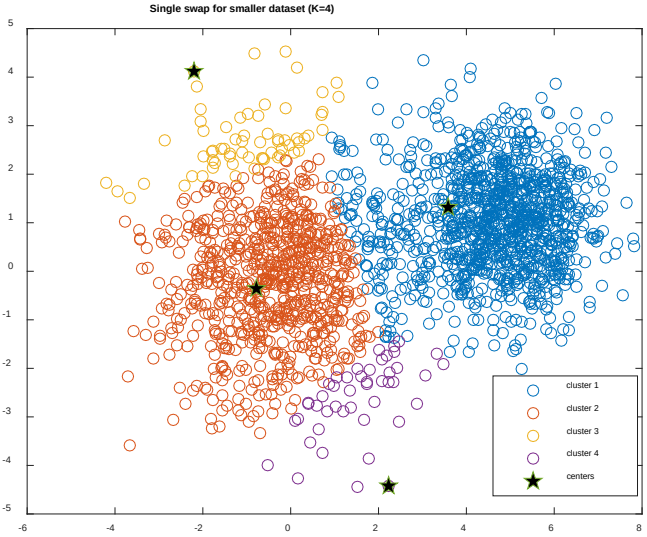
C1 = [1;1;1;2;1;3;4;1;1;1;4;1;3;1;3;2;3;2;1;…] (first 20 entries)

C2 = [2;2;2;2;2;2;2;2;1;2;2;2;1;1;2;2;2;4;2;…] (first 20 entries)



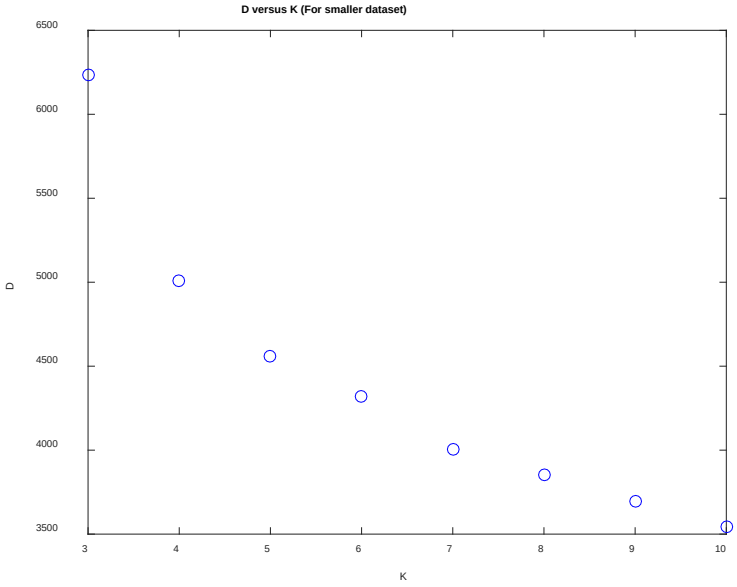

Single swap for smaller dataset (K=2)
cluster 1
cluster 2
centers


Greedy K Centers for smaller dataset (K=4)
cluster 1
cluster 2
cluster 3
cluster 4
centers

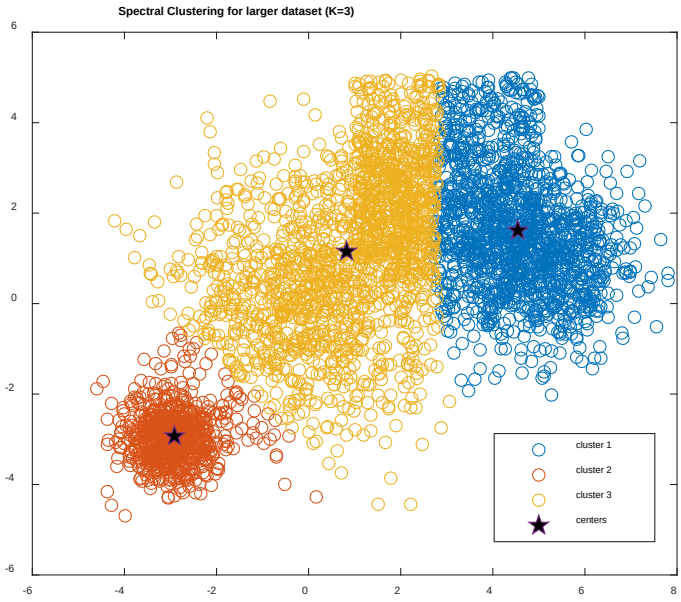

Single swap for smaller dataset (K=4)
cluster 1
cluster 2
cluster 3
cluster 4
centers

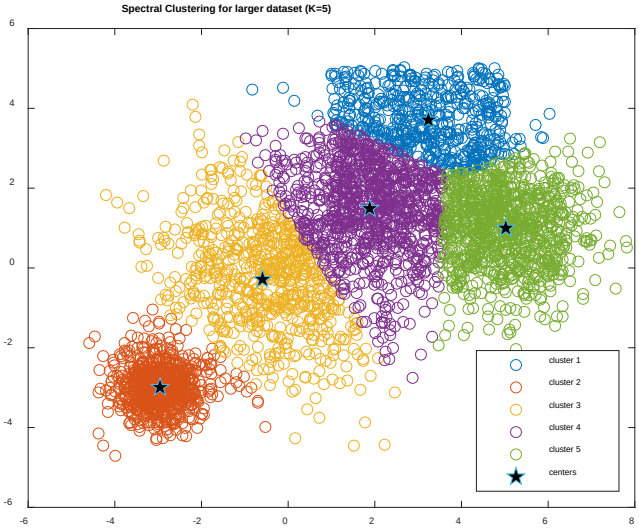
# III. Spectral Clustering

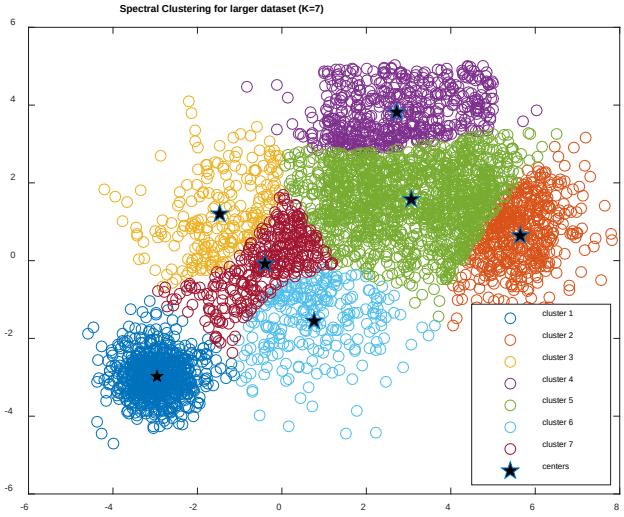
1) “bigClusteringData.mat”

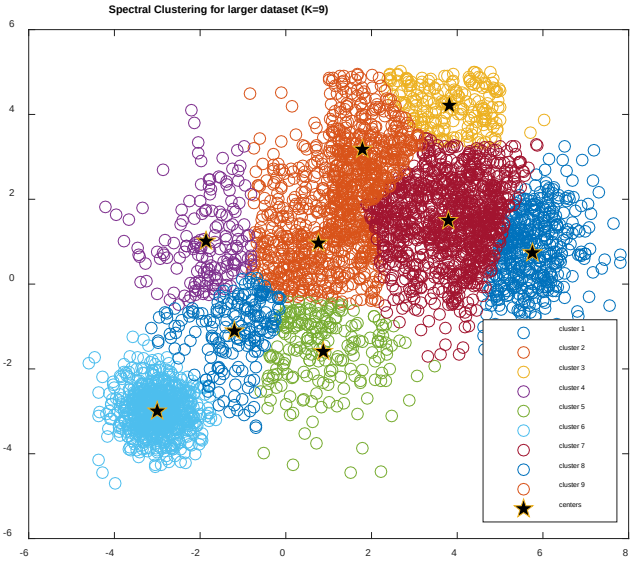



A. K = 3
   C = [1;3;2;3;3;2;3;2;3;2;3;3;1;1;3;3;1;3;3;3;…] (first 20 entries)
B. K = 5
   C = [3;1;2;1;1;2;1;2;1;2;1;4;3;3;1;4;5;1;5;4;…] (first 20 entries)
C. K = 7
   C = [2;5;1;7;7;1;5;1;5;1;5;3;5;2;5;6;5;5;4;7;…] (first 20 entries)
D. K = 9
   C = [7;7;6;2;2;6;2;6;2;6;2;4;7;1;2;5;7;2;9;2;…] (first 20 entries)

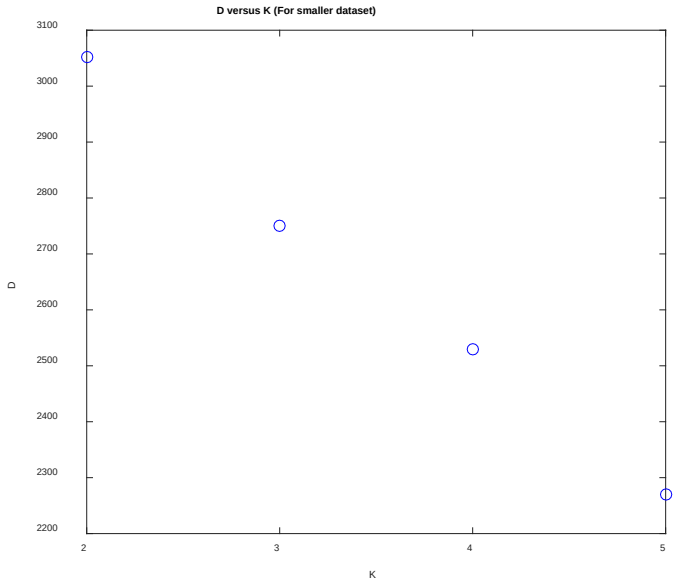



Spectral Clustering for larger dataset (K=5)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
centers

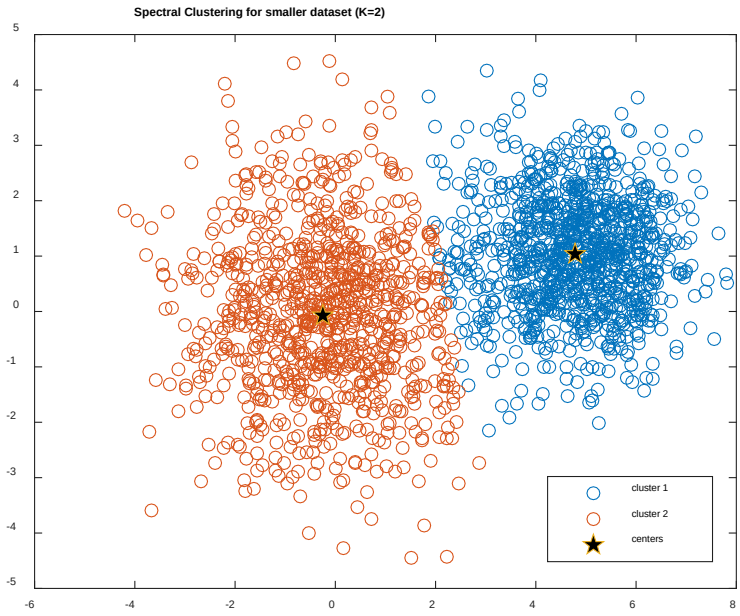

Spectral Clustering for larger dataset (K=7)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
centers

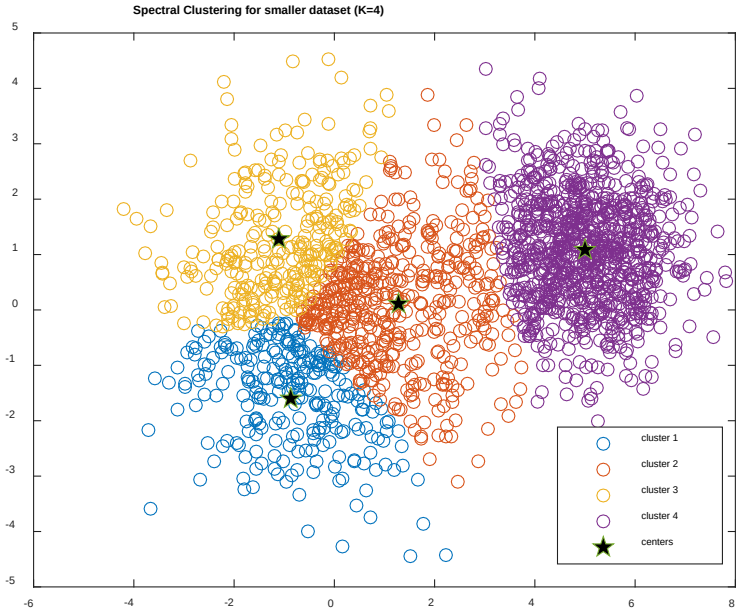

Spectral Clustering for larger dataset (K=9)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
centers

2) “clustering.mat”



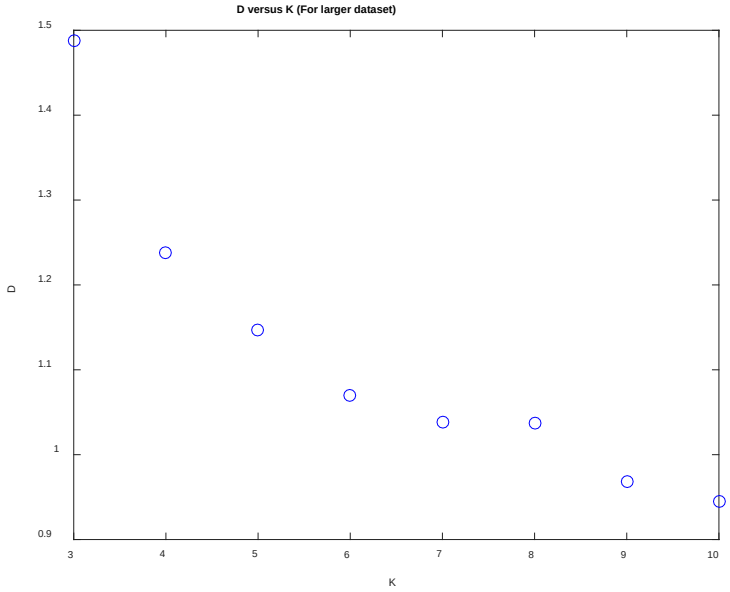

A. K = 2
   C = [2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;2;…] (first 20 entries)
B. K = 4
   C = [1;1;3;1;3;2;2;1;2;2;2;2;2;2;1;3;3;1;2;2;…] (first 20 entries)



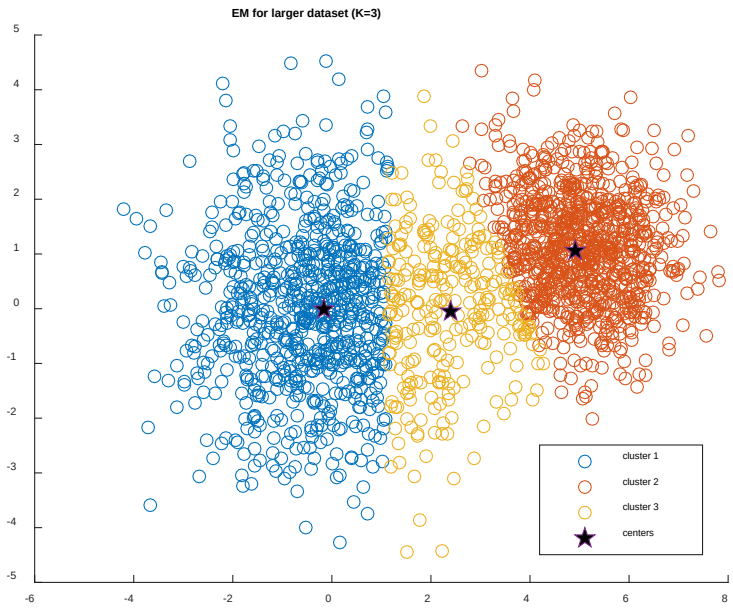

Spectral Clustering for smaller dataset (K=4)
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8
cluster 1
cluster 2
cluster 3
cluster 4
centers

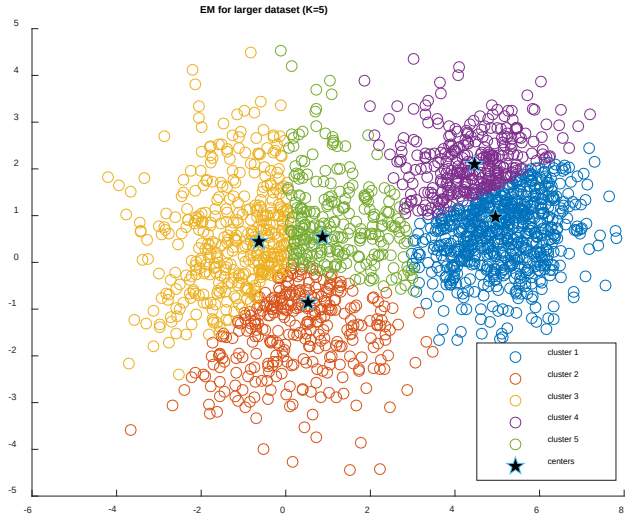
# IV. Expectation Maximization

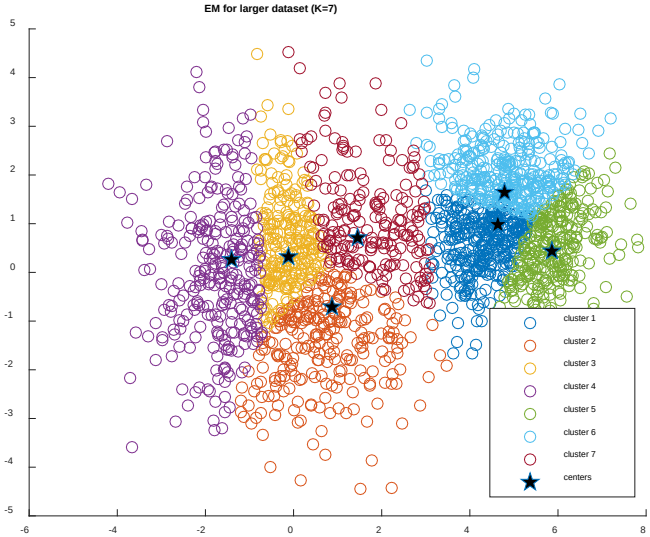
1) “bigClusteringData.mat”

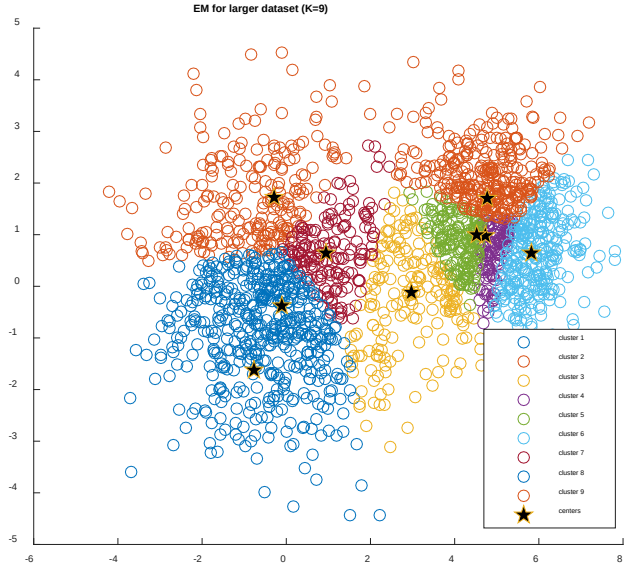



A. K = 3
   C = [1;2;3;2;2;3;2;3;2;3;2;2;1;1;1;2;2;1;2;2;…] (first 20 entries)
B. K = 5
   C = [4;3;1;2;2;1;2;1;3;1;3;2;4;4;3;2;5;3;3;2;…] (first 20 entries)
C. K = 7
   C = [5;1;3;4;4;2;4;3;1;3;1;4;5;5;1;4;6;1;1;4;…] (first 20 entries)
D. K = 9
   C = [9;8;6;3;3;6;3;6;8;68;3;5;4;8;3;5;8;7;3;…] (first 20 entries)



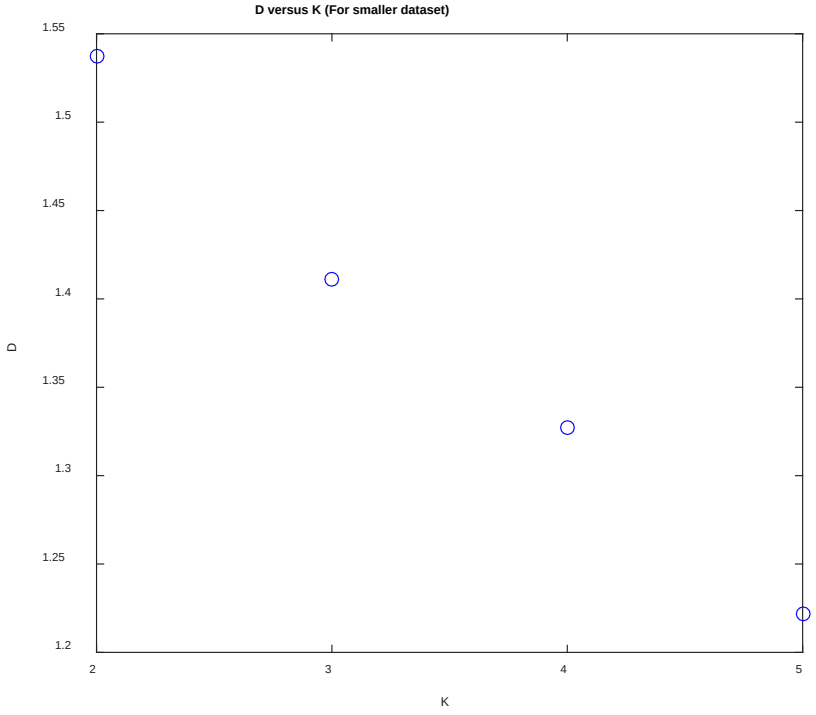

EM for larger dataset (K=5)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
centers

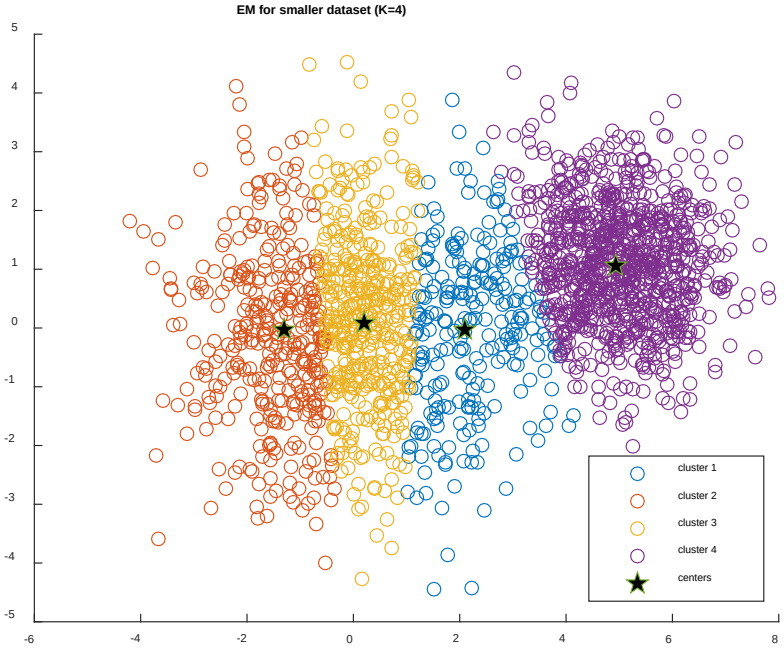

EM for larger dataset (K=7)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
centers


EM for larger dataset (K=9)
cluster 1
cluster 2
cluster 3
cluster 4
cluster 5
cluster 6
cluster 7
cluster 8
cluster 9
centers

2) “clustering”

A. K = 2
   C = [1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;1;…] (first 20 entries)
B. K = 4
   C = [3;2;2;2;2;3;3;3;1;3;3;3;3;1;2;3;3;1;2;3;…] (first 20 entries)





EM for smaller dataset (K=4)
5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8
cluster 1
cluster 2
cluster 3
cluster 4
centers

## V. Other questions

1. For the K-means algorithm (and the Spectral Clustering Algorithm since it should call your K-means algorithm) you will need to consider a series of initializations (with a minimum of 10), and then select the "best" results to report, for each K. State why these are `best', and note how many different initializations you tried.

   For each algorithm and K, the answers are provided in Part I.

2. Discuss how many `natural' clusters you think the test data contains, and why.

   From the figures above, I think dataset "bigClusteringData.mat" contains 3 natural clusters and dataset "clustering.mat" contains 2 natural clusters. Because when K=3 (for "bigClusteringData.mat") and K=2 (for "clustering.mat"), the corresponding figures show smoother edges among clusters. Or we can find the "elbow" from the figure "D versus K".

3. For the K-means algorithm, describe how you tested for convergence and why you choose this as a test.

   The answer is provided in Part I.

4. Provide a short analysis of the computational effort you found was required for each of the algorithms. Clearly state what measure you used to evaluate computational e ort, and how you observed this. Briefly discuss how these computational efforts compare to your expectations.

   Short analysis:
   Lloyd's algorithm: the computational complexity of each iteration is $O(NKd)$, complexity of the worst case is $O(\mathrm{n}^{Kd+1}\log n)$ .
   Greedy K Centers algorithm: the computational complexity of each iteration is $O(NKd)$
   Single-swap algorithm: the algorithm will terminate within $\binom{N}{K}$ steps.
   Spectral clustering: the computational complexity of the similarity matrix is $O(d^2N^2)$, and the complexity of eigenvalues and eigenvectors is $O(N^3)$
   Expectation Maximization: the computational complexity should be $O(mN^3)$, where m is the number of iterations.

   Here I evaluate the computational efforts of each algorithm via the running time. The testing computer is with i5-600U CPU and 8.00 GB memory.
   For example, when K = 5, the computational times for the lager dataset for each algorithm are:
   Lloyd: 0.3310s
   GreedyKCenters: 0.2280s
   SingleSwap: 204.6790s
   Spectral Clustering: 13.6630s
   EM algorithm: 55.6080s

5. Discuss whether your `Swap' algorithm improved the original solution found with your K-Centers algorithm (Since you were only asked to do swap on the smaller dataset, please ignore this question for the larger dataset). For each current set of centers Q, how many points in your data set did you evaluate to determine if there was a swap that would improve the outcome from your existing solution? If not all points, describe how/why you selected the points you tested. Discuss the computational effort needed to run your "Swap" algorithm and how you could further improve the performance of the algorithm.

   From the figures in Part II, it seems that "Single-swap" algorithm could improve the solution found by "GreedyKCenters" algorithm. But from the results of more runs, "Single-swap" algorithm sometimes may fail to improve. In my "Single-swap" algorithm, for every center in C, it is swapped as long as the objective value is reduced by the factor of $1-\tau = 0.95$, which means I actually do not evaluate all points. When K=5, for the larger dataset, the algorithm need 204.6790s, which is much more than other algorithms.

6. Write a brief summary paragraph of your findings for the test data set.
   Basically, I think the dataset "bigClusteringData.mat" contains 3 natural clusters and dataset "clustering.mat" contains 2 natural clusters. And Single-swap is the most time-consuming one. The results of spectral Clustering and EM algorithm seem to make more sense.
7. Source code

**Lloyd Algoritm**

```
function [Y, C, D, C0] = Lloyd_algm (X, K)
%% Lloyd Algorithm
[N,d] = size(X);
C0 = randi([1 N],K,1);
Y = X(C0,:);
cri = 1;
maxtest = 500;
ntest = 0;
% Start iteration
while cri > 1e-05
dis_bf = zeros(N,K); % distance between every point and every centroid
dis_af = zeros(K,N); % distances between the points assigned to centroid i and centroid i
C = zeros(N,1);% cluster index vector
num = zeros(K,1); % the number of points that were assigned to centroid i
D = zeros(K,1);
Y_test = zeros(K,d);
ntest = ntest +1;
for i = 1:N
      for j = 1:K
        dis_bf(i, j) = sqrt(sum((X(i,:)-Y(j,:)).^2));
      end
      [dis_min ,ind_min] = min(dis_bf(i,:));
      C(i) = ind_min;
      num(ind_min) = num(ind_min) + 1;
      dis_af( ind_min, num(ind_min)) = dis_min;
end

for p = 1:K
      D(p) = sum (dis_af(p,:))/num(p);
      Y_test(p,:) = sum(X(find(C==p),:),1)/num(p);
end

cri = norm (Y - Y_test);
if cri > 1e-05
      Y = Y_test;
end

if ntest >= maxtest
      text( 'no results, iteration ends')
      break
end
end
```

**Greedy K Centers**

```
function [maxmin, Q, C, CC, D] = GreedyKCenters(X, K)
[N,~] = size(X);
CC = zeros(K,1);
CC(1) = randi([1 N],1,1);
dis_org = zeros(N,1);
for i = 1:N
      dis_org(i) = sqrt(sum((X(CC(1),:)-X(i,:)).^2));
end
[~,maxind] = max(dis_org);
CC(2) = maxind;
for i = 2:K
      dis = zeros(N,i);
     for k = 1:i
      for j = 1:N
      dis(j,k) = sqrt(sum((X(CC(k),:)-X(j,:)).^2));
      end
     end
      dis_min = min (dis,[],2);
      [maxmin, ind_max] = max (dis_min);
      if i<K
      CC(i+1) = ind_max;
      end
end
Q = X(CC,:);
%% Calculate distances, C and D
dis_bf = zeros(N,K); % distance between every point and every centroid
dis_af = zeros(K,N); % distances between the points assigned to centroid i and centroid i
C = zeros(N,1);% cluster index vector
num = zeros(K,1); % the number of points that were assigned to centroid i
D = zeros(K,1);
for i = 1:N
      for j = 1:K
        dis_bf(i, j) = sqrt(sum((X(i,:)-Q(j,:)).^2));
      end
      [dis_min ,ind_min] = min(dis_bf(i,:));
      C(i) = ind_min;
      num(ind_min) = num(ind_min) + 1;
      dis_af( ind_min, num(ind_min)) = dis_min;
end
for p = 1:K
      D(p) = sum (dis_af(p,:))/num(p);
end
end
```

**Single Swap**

```
function [maxmin, Q, C, D] = SingleSwap(X, CC, K, maxmin)
[N,~] = size(X);
for i = 1:K
      NN = 1:N;
      C_test = CC;
      NN(CC)=[];

      for p = 1:N-K
            NN_test = NN;
            C_test(i) = NN_test(p);
            NN_test(p) = CC(i);
            dis = zeros(N-K,K);
      for k = 1: K
       for j = 1:N-K
             dis(j,k) = sqrt(sum((X(C_test(k),:)-X(NN_test(j),:)).^2));
       end
      end
       dis_min = min (dis,[],2);
      [maxmin_test,~] = max (dis_min);
      cost_rdc = maxmin - maxmin_test;
      if cost_rdc > 0.95
            CC = C_test;
            maxmin = maxmin_test;
           break
      end
      end
end
Q = X(CC,:);
%% Calculate distances, C and D
dis_bf = zeros(N,K); % distance between every point and every centroid
dis_af = zeros(K,N); % distances between the points assigned to centroid i and centroid i
C = zeros(N,1);% cluster index vector
num = zeros(K,1); % the number of points that were assigned to centroid i
D = zeros(K,1);
for i = 1:N
      for j = 1:K
       dis_bf(i, j) = sqrt(sum((X(i,:)-Q(j,:)).^2));
      end
      [dis_min ,ind_min] = min(dis_bf(i,:));
      C(i) = ind_min;
      num(ind_min) = num(ind_min) + 1;
      dis_af( ind_min, num(ind_min)) = dis_min;
end
```

```
for p = 1:K
      D(p) = sum (dis_af(p,:))/num(p);
end
end
```

**Spectral clustering**

```
%Normalized spectral clustering algorithm
function [W,U,C,Dis] = Spectral1(X,K)
    %Initialize
    N = size(X,1); S = zeros(N); D = zeros(N);
    %Similar matrix construction
    for i = 1:N
        for j = 1:N
            S(i,j) = norm(X(i,:)-X(j,:));
            S(j,i) = S(i,j);
          end
     end
    %Weighted adjacency matrix
    W = ones(N)./(exp(S));
    %degree matrix
    for i = 1:N
          D(i,i) = sum(W(i,:));
    end
    %Laplacian
    L = D - W;
    %Eigenvalue and Eigenvector
    [Eig_Vec,Eig_Val] = eig(L,D);
      %Order the eigenvalue
    [~,Ind] = sort(sum(Eig_Val));
      %Construct U
    U = Eig_Vec(:,Ind(1:K));
      %k-means algorithm
    [~,C] = Runkmeans(U,K,X,N);
      Dis = DistCal(X,C,K);
    end
  %Compute the total distortion of a certain clustering result C
function Dis = DistCal(X,C,K)
    %Define anonymous function for calculate norm for each row
    rowdist = @(dist_matrix,P) sum(abs(dist_matrix).^P,2).^(1/P);
    %Initialize distortion
    Dis = 0;
    %Compute distortion of each cluster
for i = 1:K
    %Identify points in cluster
      clus = find(C==i);
      %Locate center of cluster
      cent_cord = mean(X(clus,:));
      %Compute the distortion of each point
      Dist_Matrix = bsxfun(@minus,X(clus,:),cent_cord);
```

```
            Dist_Vector = rowdist(Dist_Matrix,2);
            %Distortion of cluster
            Dis = Dis + sum(Dist_Vector);
        end
    end

%A function to find the best instance of cluseters with certain K
function [dist_best,c_best] = Runkmeans(U,K,X,N)
    %Initialize
    dist_best = Inf; k = 1; C = zeros(N,1);
    %Run 50 instances
    while k<=50
        %k-means algorithm
        [~,Assig,Distortion,~] = Lloyd_algm(U,K);
        %Find new "best" instance
        if Distortion < dist_best
            %Update
            dist_best = Distortion;
            c_best = Assig;
        end
        %Increment counter
        k = k + 1;
    end
end
```

**Expectation Maximization**

```
function [mu,sigma,C,D]=EMalgo(X,K)
[m,n] = size(X);
indeces = randperm(m);
%initial means
mu = X(indeces(1:K), :);
%initial variance
sigma = [];
% Use the overal covariance of the dataset as the initial variance for each cluster.
for (j = 1 : K)
      sigma{j} = cov(X);
end
%initial phi
phi = ones(1, K) * (1 / K);
%%Run Expectation Maximization
W = zeros(m, K);
prevMu = mu + ones(K,n);
% Loop until convergence.
iter = 0;
while (norm(mu - prevMu)>1e-5);

      fprintf('    EM Iteration %d\n', iter);
%%=====================================================
%%Expectation Step
pdf = zeros(m,K);
for (j = 1 : K)
            iter = iter+1;
            % Evaluate the Gaussian for all data points for cluster 'j'.
            pdf(:, j) = gaussianND(X, mu(j, :), sigma{j});
end

pdf_w = bsxfun(@times, pdf, phi);
W = bsxfun(@rdivide, pdf_w, sum(pdf_w, 2));
%%
%maximization step
prevMu = mu;
for (j = 1 : K)

            % Calculate the prior probability for cluster 'j'.
            phi(j) = mean(W(:, j), 1);

            % Calculate the new mean for cluster 'j' by taking the weighted
            % average of all data points.
```

```
            mu(j, :) = weightedAverage(W(:, j), X);

            % Calculate the covariance matrix for cluster 'j' by taking the
            % weighted average of the covariance for each training example.

            sigma_k = zeros(n, n);

            % Subtract the cluster mean from all data points.
            Xm = bsxfun(@minus, X, mu(j, :));

            % Calculate the contribution of each training example to the covariance matrix.
            for (i = 1 : m)
                  sigma_k = sigma_k + (W(i, j) .* (Xm(i, :)' * Xm(i, :)));
            end

            % Divide by the sum of weights.
            sigma{j} = sigma_k ./ sum(W(:, j));
      end
   %% Calculate distances, C and D
dis_bf = zeros(m,K); % distance between every point and every centroid
dis_af = zeros(K,m); % distances between the points assigned to centroid i and centroid i
C = zeros(m,1);% cluster index vector
num = zeros(K,1); % the number of points that were assigned to centroid i
D = zeros(K,1);
for i = 1:m
      for j = 1:K
        dis_bf(i, j) = sqrt(sum((X(i,:)-mu(j,:)).^2));
      end
      [dis_min ,ind_min] = min(dis_bf(i,:));
      C(i) = ind_min;
      num(ind_min) = num(ind_min) + 1;
      dis_af( ind_min, num(ind_min)) = dis_min;
end
for p = 1:K
      D(p) = sum (dis_af(p,:))/num(p);
end
% End of Expectation Maximization
end
end

function [ pdf ] = gaussianND(X, mu, Sigma)
% Get the vector length.
n = size(X, 2);
% Subtract the mean from every data point.
```

```
meanDiff = bsxfun(@minus, X, mu);
% Calculate the multivariate gaussian.
pdf = 1 / sqrt((2*pi)^n * det(Sigma)) * exp(-1/2 * sum((meanDiff * inv(Sigma) .* meanDiff),
2));
end

function    val    = weightedAverage(weights, values)
%WEIGHTEDAVERAGE Calculate the weighted average of 'values' by applying
% the 'weights'
%
%      values - Data points to average, one per row.
%    weights - Weight to apply to each data point, one per row.
%
%    Returns:
%          val    - The weighted average of 'values'.
       % Apply the weights to the values by taking the dot-product between the
       % two vectors.
       val = weights' * values;
       % Divide by the sum of the weights.
       val = val ./ sum(weights, 1);
  end
```