CSE 847 (Spring 2023): Machine Learning— Homework 5
Instructor: Jiayu Zhou
Balija Santoshkumar
balijasa@msu.edu

1 Clustering: K-means

1. Elaborate the relationship between k-means and spectral relaxation of k-means. Is it possible that we obtain exact k-means solution using spectral relaxed k-means?

2. Implementation of k-means. Submit all the source code to D2L along with a short report on your observation.

   - Implement the k-means in MATLAB using the alternating procedure introduced in the class (you will not get the credit if you use the build-in kmeans function in MATLAB).
   - Implement the spectral relaxation of k-means. Create a random dataset and compare the k-means and spectral relaxed k-means.

Sol:
We assume that we have n data points $\{x_i\}_{i=1}^n \in \mathbb{R}^m$, which we organize as columns in a matrix

$$X = [x_1, x_2, \cdots, x_n] \in \mathbb{R}^{m \times n}$$

The objective of K-means is reduce sum squared error(SSE)

$$q_j = \sum_{v \in \pi_j} \|x_v - c_j\|^2 \tag{1}$$

where c_j is the cluster center of the corresponding cluster Let e be the vector of all ones with appropriate length. It is easy to see that $c_j = X_j e/n_j$, where X_j is the data matrix of the j-th cluster.

SSE can transformed into

$$q_j = \sum_{j=1}^{k} \left(\text{trace}\left(X_j^T X_j\right) - \frac{e^T}{\sqrt{n_j}} X_j^T X_j \frac{e}{\sqrt{n_j}} \right) \tag{2}$$

Define the n-by- k orthogonal matrix Y as follows

$$Y = \begin{pmatrix} e/\sqrt{n_1} & & & \\ & e/\sqrt{n_2} & & \\ & & \vdots & \\ & & & e/\sqrt{n_k} \end{pmatrix}$$

Then

$$Q(\Pi) = \text{trace}\left(X^T X\right) - \text{trace}\left(Y^T X^T X Y\right).$$

The k-means objective, minimization of $Q(\Pi)$, is equivalent to the maximization of trace $\left(Y^T X^T X Y\right)$ with Y .

In spectral relaxation of k-means is instead of using this specific expression for Y , it is possible to use any arbitrary orthogonal matrix for Y . This leads to the relaxed maximization problem

$$\max_{Y^T Y = I_k} \text{trace}\left(Y^T X^T X Y\right) \tag{3}$$

The first k vectors in the left singular matrix of X can produce the Y that maximizes this expression in Eq.3 we can see that both k-means and spectral k-means are trying to minimize the same error function, but spectral k-means is first trying to project the dataset into a lower dimensional space which makes it easier to capture the complex clustering structures.

The spectral-relaxed k-means become completely equivalent to k-means when the expression for Y becomes equal to the matrix mentioned in Eq. 2

I have used fisheriris data set for K-means

Regular K-means

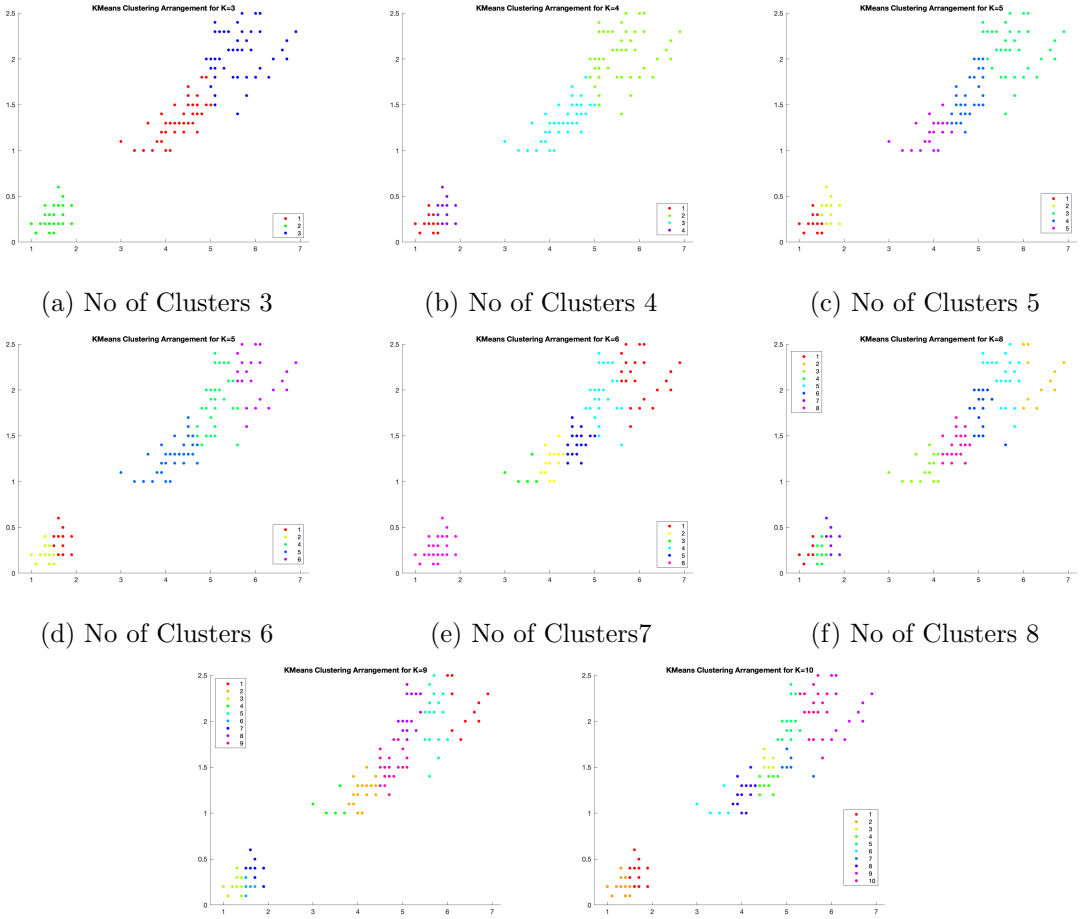
After implementation we see SSE as

```
SSE for k=3: 26.893170
SSE for k=4: 26.067859
SSE for k=5: 14.691709
SSE for k=6: 11.985356
SSE for k=7: 8.231259
SSE for k=8: 6.774484
SSE for k=9: 6.016505
SSE for k=10: 5.094189
```

spectral relaxation of k-means

After implementation we see Final SSE as

```
SSE for k=3: 145.287135
SSE for k=4: 107.839511
SSE for k=5: 127.624815
SSE for k=6: 82.031575
SSE for k=7: 106.387628
SSE for k=8: 70.359438
SSE for k=9: 69.009363
SSE for k=10: 43.252894
```



(a) No of Clusters 3 (b) No of Clusters 4 (c) No of Clusters 5

(d) No of Clusters 6 (e) No of Clusters7 (f) No of Clusters 8

(g) No of Clusters 9 (h) No of Clusters 10

Figure 1: Cluster assignments with K-means

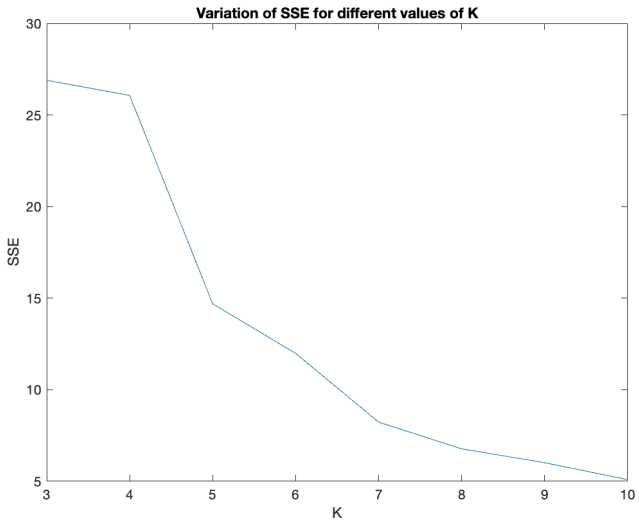
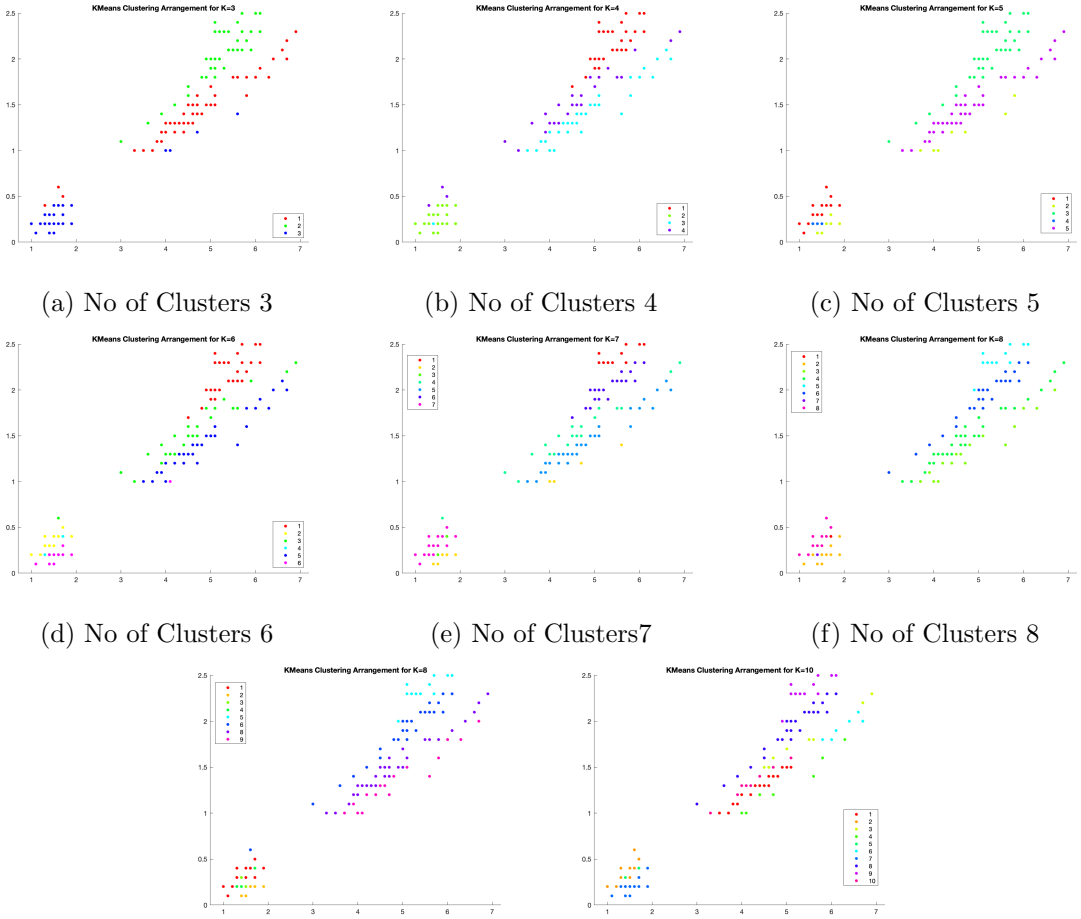


Figure 2: Convergence of SSE with Clusters variation in KMeans



(a) No of Clusters 3 (b) No of Clusters 4 (c) No of Clusters 5

(d) No of Clusters 6 (e) No of Clusters7 (f) No of Clusters 8

(g) No of Clusters 9 (h) No of Clusters 10

Figure 3: Cluster assignments with Spectral relaxation K-means

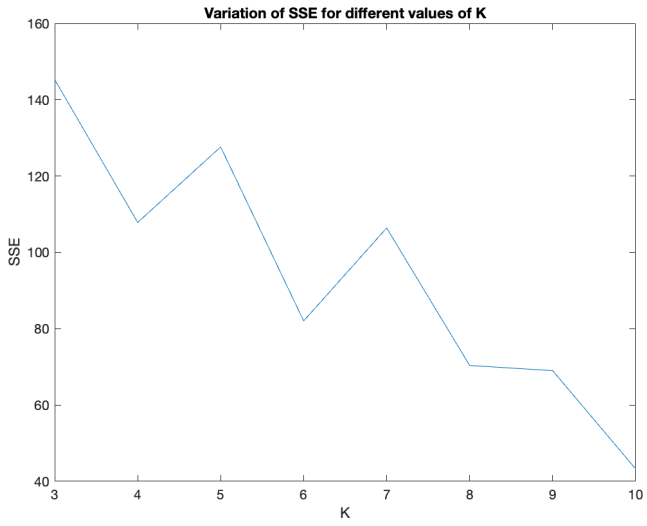


Figure 4: Convergence of SSE with Clusters variation in Spectral relaxation KMeans

MATLAB Code

```
clear all
clc
close all
load fisheriris
data = meas
data = data(:,3:4);
num_clusters = [3:10];
SSE = zeros(1, size(num_clusters,2));
for i=1:size(num_clusters,2)
[cluster_assignments, cluster_centers] = kmeans_cluster(data
   , num_clusters(i),1);
SSE(i) = compute_SSE(data, cluster_assignments);
show_plot(data, cluster_assignments);
saveas(gcf, strcat('Results/KMeans/Cluster_Arrangement_S',
   int2str(num_clusters(i)),'.png'));
end
fig = figure; plot(num_clusters, SSE); xlabel('K'); ylabel('
   SSE'); title('Variation of SSE for different values of K'
   );
saveas(gcf, strcat('Results/KMeans/SSE_Convergence_S.png'));


function [cluster_assignments, cluster_centers] =
   kmeans_cluster(raw_data, k, spectral)
%     Code to perform k-means clustering
%     INPUTS:
%         data     =  (n * m) matrix where n is the number
   of samples and m
%                     is the number of features
%         k        =  integer number of intended clusters
%         spectral =  boolean value representing spectral k-
   means if true,
%                     else standard k-means
%
%     OUTPUTS:
%         cluster_assignments = labels assigned to the each
   samples in
%                               [1,k]
%         cluster_centers     = final clsuter centers found
   in the process
%                               of k-means

% Assigning default values
if nargin < 3
if ~exist('spectral')
```

```
spectral=false;
end
end

if spectral
% for spectral relaxation, map the data samples to k-
   dimensional
% feature space
[U, ~, ~] = svd(raw_data);
projection = U(:, 1:k);
rand_mat = rand(k,k);
orth_mat = orth(rand_mat);
data = projection * orth_mat;
else
data = raw_data;
end

[num_samples, ~] = size(data);
cluster_assignments = zeros(num_samples, 1);
temp = randperm(num_samples);
cluster_center_idx = temp(1:k);
cluster_centers = data(cluster_center_idx, :);
change = inf;
count_iter = 0;

while(change ~= 0)
% change represents the number cluster assignments that got
   changed
% in the current iteration
count_iter = count_iter+1;
prev_assignments = cluster_assignments;

for cur_idx=1:num_samples
min_dist = inf;
min_idx = -1;

% for each sample, find the cluster center which is at min
% distance
for cluster_idx = 1:k
cur_dist = norm(data(cur_idx,:) - cluster_centers(
   cluster_idx,:));
if(cur_dist < min_dist)
min_dist = cur_dist;
min_idx = cluster_idx;
end
end
cluster_assignments(cur_idx,1) = min_idx;
```

```
end

for cluster_idx = 1:k
% get the mean of each cluster
cluster_centers(cluster_idx,:) = mean(data(
   cluster_assignments == cluster_idx,:));
end

change = sum(prev_assignments ~= cluster_assignments);
%         fprintf('Number of changes in iter %d: %d\n',
   count_iter, change);
%         show_plot(raw_data, cluster_assignments);   % plot
    the clusters
end

SSE = compute_SSE(raw_data, cluster_assignments);
fprintf('SSE for k=%d: %f\n', k, SSE);

end


function [SSE] = compute_SSE(data, cluster_assignments)
% Function to compute Sum of Squared Error
% INPUTS:
%   data = the dataset used for clustering
%   cluster_assignments = labels for each sample in the data
%   cluster_centers = the centers found for each cluster
%
% OUTPUT:
%   SSE = final sum of squared errors for the cluster config
   .

num_clusters = size(unique(cluster_assignments),1);
SSE = 0;
for cluster_no = 1:num_clusters
cluster_center = mean(data(cluster_assignments == cluster_no
   ,:));
SSE = SSE + norm(data(cluster_assignments==cluster_no,:)-
   cluster_center)^2;
end
end

function [] = show_plot(data, labels)
%   Function to plot the cluster config.
%   INPUTS:
%       data = dataset used for clustering
%       labels = the cluster label assigned to each sample
```

```
%
%   OUTPUT:
%       A plot representing the cluster config.

k = size(unique(labels),1);
[~, num_features] = size(data);


if num_features>2
pcs = pca(data);
reduced_data = data * pcs(:, 1:2);
else
reduced_data = data;
end

figure;
hold on;
gscatter(reduced_data(:,1),reduced_data(:,2),labels);
title(strcat('KMeans Clustering Arrangement for K=', int2str
   (k)));
hold off;
pause(2);
end
```

2 Principle Component Analysis

1. Suppose we have the following data points in 2 d space $(0,0),(-1,2),(-3,6),(1,-2),(3,-6)$.

   - Draw them on a 2-d plot, each data point being a dot.
   - What is the first principle component? Given 1-2 sentences justification. You do not need to run MATLAB to get the answer.
   - What is the second principle component? Given 1-2 sentences justification. You do not need to run MATLAB to get the answer.

2. Experiment: We apply data pre-processing techniques to a collection of handwritten digit images from the USPS dataset (data in MATLAB format: USPS.mat) [1]. You can load the whole dataset into MATLAB by load USPS.mat. The matrix A contains all the images of size 16 by 16. Each of the 3000 rows in A corresponds to the image of one handwritten digit (between 0 and 9). To visualize a particular image, such as the second one, first you need to convert the vector representation of the image to the matrix representation by $A2 = \text{reshape}(A(2,:),16,16)$, and then use imshow $(A2')$ for visualization.

   Implement Principal Component Analysis (PCA) using SVD and apply to the data using $p = 10, 50, 100, 200$ principal components. Reconstruct images using the selected principal components from part 1.

   - Show the source code links for parts 1 and 2 to your github account.
   - The total reconstruction error for $p = 10, 50, 100, 200$.
   - A subset (the first two) of the reconstructed images for $p = 10, 50, 100, 200$. Note: The USPS dataset is available at http://www. csie.ntu.edu.tw/ cjlin/libsvmtools/ datasets/multiclass.html#usps. The image size is 16 by 16 , thus the data dimensionality of the original dataset is 256 . We used a subset of 3000 images in this homework.

Sol: