CSE6740: Computational Data Analysis **(Due: 05/24/2020)**

Submission Assignment #1

Instructor: Prof Xie *Name:* Student name(s):Jian Pang, *Netid:*

Problem 1 (10+5+10 points)

(1) Solution

$$J = \sum_{i=1}^{m}\sum_{j=1}^{k} r^{ij}||x^i - \mu^j||^2$$

$$\begin{aligned}\frac{dJ}{d\mu} &= \sum_{i=1}^{m}\sum_{j=1}^{k} r^{ij}|| - 2x^i + 2\mu^j|| \\ &= \sum_{i=1}^{m}\sum_{j=1}^{k}(-2r^{ij}x^i + 2r^{ij}\mu^j)\end{aligned}$$

To minimize J, let

$$\frac{dJ}{dx} = 0,$$

and for a specific cluster j,

$$\frac{dJ}{dx} = \sum_{i=1}^{m} 2r^{ij}x^i - \sum_{i=1}^{m} 2r^{ij}\mu^j = 0$$

Therefore,

$$\mu^j = \frac{\sum_{i=1}^{m} 2r^{ij}x^i}{\sum_{i=1}^{m} 2r^{ij}}$$

(2) Since we assign each x^i to a cluster j based on the criteria that

$$r^{ij} = \{x^i : ||x^i - \mu^j||^2 \leq ||x^i - \mu^h||^2 \; \forall \; h, 1 \leq h \leq k\}$$

so that it makes sure each point is assigned to a closest cluster. Then the cluster centroid will be updated based on the previous assignment. As the assigned points are closer to the cluster centroid, the cost function J in question 1 will always decrease monotonically, while still bounded by 0. This is the reason why it will finally converge. However, because the initialization process is subject to randomness, i.e, the number of the clusters and the number of the clusters. Therefore, for each initialization configuration, there is only local optimal guaranteed. The above process will keep iterating until we find that each point has been assigned to a closest cluster, and the centroid will not change. The above criteria shows that the distance will have a lower bound of zero, and thus, the algorithm will converge no matter how.

(3)

$$P = [(2,2),(3,1),(-1,1),(0,-1),(-2,-2)]; K_1 = [K_{B1} = (2,1), K_{A1} = (-3,1)];$$
$$D_1 = [1,1,3,3,2]; \sum D_1 = 10;$$
$$C_1 = [B_1, B_1, B_1, A_1, A_1];$$
$$K_2 = [K_{B2} = [4/3, 4/3], K_{A2} = [-1, -3/2]];$$
$$D_2 = [4/3, 2, 5/2, 3/2, 3/2]; \sum D_2 = 8.833;$$
$$C_2 = [B_2, B_2, A_2, A_2, A_2];$$
$$K_3 = [K_{B3} = (5/2, 3/2), (-3/2, -1)];$$
$$D_3 = [1,1,5/2,3/2,3/2]; \sum D_3 = 7.5;$$
$$C_3 = [B_3, B_3, A_3, A_3, A_3];$$

Based on C_3, the next iteration will generate the same cluster centroids as K_3, and therefore, the result will still be identical. This kmeans algorithm will generate the optimal solution in 3 steps. The final cluster is shown in K_3.

Problem 2 (5+10+10+5=30 points)

(1) First, I import the image and convert it to an array to process the data. To start with the clustering, I first write a function $init\_rep(X, n)$ to initialize the clusters representatives. The cluster takes the input of the original array and the number of initial clusters. Second, I write a function $get\_class(X, k, minkowski_order)$ to calculate the distance between each data point and the cluster representatives and assign the data points to the cluster to whom it is closest . Here I use the Minkowski distance, and allow the user to input the distance order. For example, Euclidean distance will be 2, and Manhattan distance will be 1. Third, I write a function $update\_cluster_means(data, unique_c, cluster)$ to update the cluster centroids by calculating the mean of the data points belonging to the cluster. Based on the newly-generated centroids, we still need to pick the new reprensentatives. Therefore, the fourth step is to design a function $find\_new\_rep(data, new_means, minkowski_order)$ to find the closest data point around the new cluster centroid and assign them as new representatives.
Next, I put everything together into one function $kmedoid(X, n, minkowski_order, threshold, steps)$. The inputs here are: the data array, the number of clusters needed, the Minkowski distance order, the threshold for the algorithm to converge, and last but not least, the maximum number of iterations.

(2)

cluster/img	my own image	beach	football
3	0.88s	0.55s	5.11s
16	1.90s	6.10s	23.00s
64	4.66s	10.08s	61.89s

The above table shows that as the number of clusters (k) increases, the processing time increases accordingly. However, the growth of the two is not proportional.

(3)
After my experiment with random initialization of representative assignment for 50 times (10 clusters, Manhattan distance, converging threshold = 1000, 50 max iterations), I find that the randomness will create variance in both the final converging results as well as processing time.
The longest processing time is 3.05s and the shortest is 0.52s. The standard deviation of the time spent within the 50 experiments is 0.71s.
The largest distortion is 903,779 and the smallest distortion is the 811,753. The standard deviation of the distortion within the 50 experiments is 22,209.

(4)
I use my own image to test the performance difference of K Means. First, I test the difference between different number of clusters.

cluster/img	time
3	0.80s
16	2.03s
64	7.44s

The result seems to show that when the number of clusters is small, the performance between K Means and K Medoids do not vary by much. As the number of clusters grows, K Medoids has an advantage in processing time over K Means.

Second, I compare the randomness in the initialization process in K Means with K Medoids. I also run the same configuration (10 clusters, Manhat-tan distance, converging threshold = 1000, 50 max iterations) 50 times. The minimum time 0.67s and the maximum time is 3.10s, and the standard deviation is

Problem 3 (10+10+5=25 points)

(1)

Spectral clustering with solve the question. First, these points are not distributed in space in a random way and there is a clear pattern as to how the points are clustered. In other words, the distance between points follow a certain pattern that will not work on ways like K Means which relies purely on the distance measures that is not capable of identifying the non-linear patterns in the distribution of the data, like the two torus shown in the image.

Spectral clustering will serve this purpose because it identifies the connection between points, rather than spatial distance. By identifying the connection between points, we can build a network graph with a degree of connectivity attached on each nodes and a weight to each edge. It's more effective to detect the special non-linear distribution of the data because the network graph will do a better job at fitting the pattern of the distribution by setting a threshold to disconnect the nodes that whose distance are beyond the threshold. The graph is a new space that's detached from the original data space, while maintaining the non-linear patterns of the data distribution within the graph, so that it avoids the hollow part/empty areas in the original Euclidean space, which will cause uncertainty and instability in classification because the centroid will swift drastically when these areas are involved.

First, we have to create a neighbor graph for all the points. The way to do it is to use a Euclidean distance threshold t. For data point i, the other data point j whose distance d_{ij} to that point falls within the threshold t will be assigned as its neighbors and with an edge that connects them . The we can construct the adjacency matrix A whose entries will be w_{ij} which is equivalent to d_{ij}

$$A_{ij} = \begin{cases} w_{ij}: & \text{weight of edge(i,j) if } d_{ij} \leq t \\ 0: & \text{if no edge between i, j} \end{cases} \tag{0.1}$$

Then, we can sum the all the weights w_{ij} for one data points, and construct the degree matrix by placing the sum value on the diagonal and keep the rest of the matrix zero.

$$D = \begin{bmatrix} d_1 = \sum_{j=1}^{m} w_{1j} & 0 & \cdots & 0 \\ 0 & d_2 = \sum_{j=1}^{m} w_{2j} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n = \sum_{j=1}^{m} w_{nj} \end{bmatrix} \tag{0.2}$$

Next, we construct the Laplacian matrix L by subtracting A from D. Finally, by performing the Eigenvalue decompostion, we will be able to tell the main distinguishers in the data, along which, the variance of the original graph is preserved as much as possible. These distinguishers will the eigenvectors corresponding to any of the smallest eigenvalues that's above zero.

(2)
In the original data, excluding those isolating websites, 636 (52.0%) are left-oriented 588 (48.0%) and right-oriented. The clustering result shows that the accuracy of the spectral clustering is only 60.3%. Among those, the ones that are "1" in the original data, which is left-oriented have a percentage of 72.6% being mis-categorized into being right-oriented. For the "0"s that are right-oriented in the original data, only 4% are mis-categorized.

One of the reason might be that I am using the unnormalized cut minimization instead of the normalized cut minimization. When we are trying to minimize the weighs loss when cutting the graph into clusters, it also creates some bias, as the more isolated the cluster is, the less is the weight that's associated to the edge that's connecting that cluster. So I assume what happened in the algorithm was that it finds a cut that separates a relatively isolated cluster within the left-oriented cluster, let's say they are the extreme leftism that even the moderate-left do not connect with them very well, and the algorithm turn those into the new left-oriented cluster, and put the original moderate-left into the new right-oriented cluster. This makes sense in real world as we see the extreme left ideas might not appeal to most people, not necessarily because they are really extreme but because they are more vocal, whereas the extreme right might be well-hidden within the right community and not capturing the attention of the general public. In terms of the algorithm, if I use a normalized cut minimization by taking the sum of the edge weights in each cluster into consideration, the result might be improved because the less-weighted isolated cluster will be scaled up.

Problem 4 (15+10=25 points)

(1) The data matrix has 21 columns and 16 rows (excluding the NA rows). So the matrix has a dimension of 16*21. Each dimension of the data corresponds to one food type. To perform PCA, first we need to center the data by extracting the mean value of each dimension from the data point. Then we need to generate a covariance matrix by doing the outer product of the centered data matrix. The point of PCA is to reduce the dimensions of the orignal data while still maintaining the variance of the data. This is why we need the covariance matrix.

Then we perform PCA on the covariance matrix by doing eigenvalue decomposition. The reason to do this is because in the direction where the data has the largest variance is where the largest eigenvalue will be. And that largest eigenvalue will correspond to the most major eigenvector. Each eigenvector is a linear combination of the orignal dimensions, which represents a combination of the multiple food categories. (eg. poultry/vegetable/red meat).

(2) Let C be the covariance matrix. We are trying to find eigenvectors w that $Cw = \lambda w$. Then we get $Cw - \lambda w = 0$. To solve for w, we need to find the $det(C - \lambda I) = 0$ where I is the identity matrix. As a result, we get $\lambda = [\lambda_1, \lambda_2, ..., \lambda_m]$. Then we sort the λ from the largest to smallest to filter out the major eigenvalues.

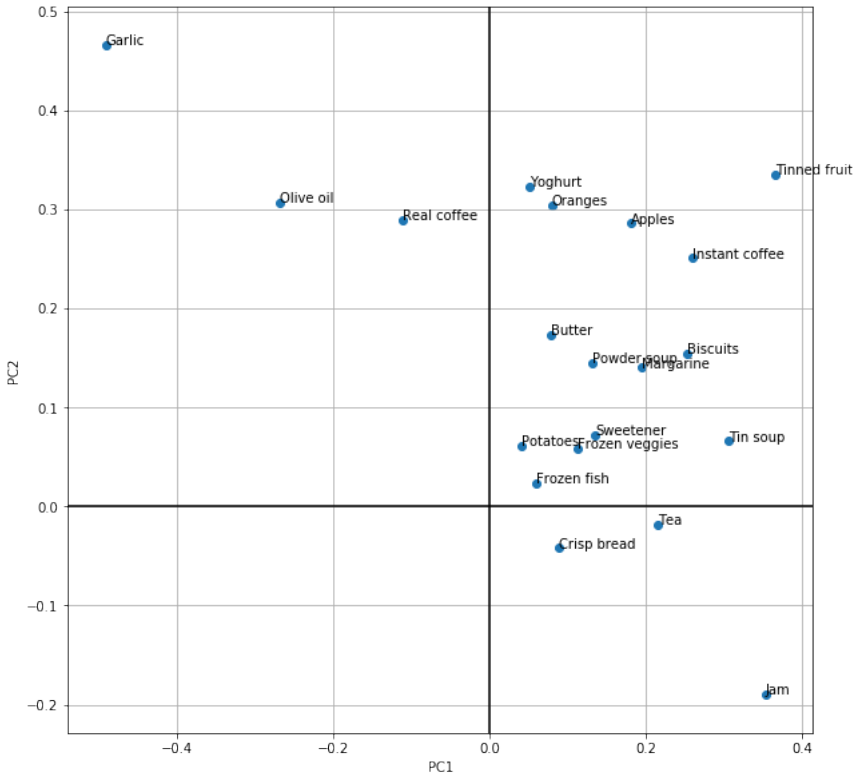
As the result of the eigenvalue decomposition, we will have a $16*m$ matrix where m is the number of eigenvectors, so naturally, each column vector is the eigenvector. Because we feed the covariance matrix, not the original data matrix into the eigenvalue decomposition, each cell is the new matrix is still the covariance but represented by the new principal component. We then find the k eigenvectors that correspond to the top k eigenvalues.

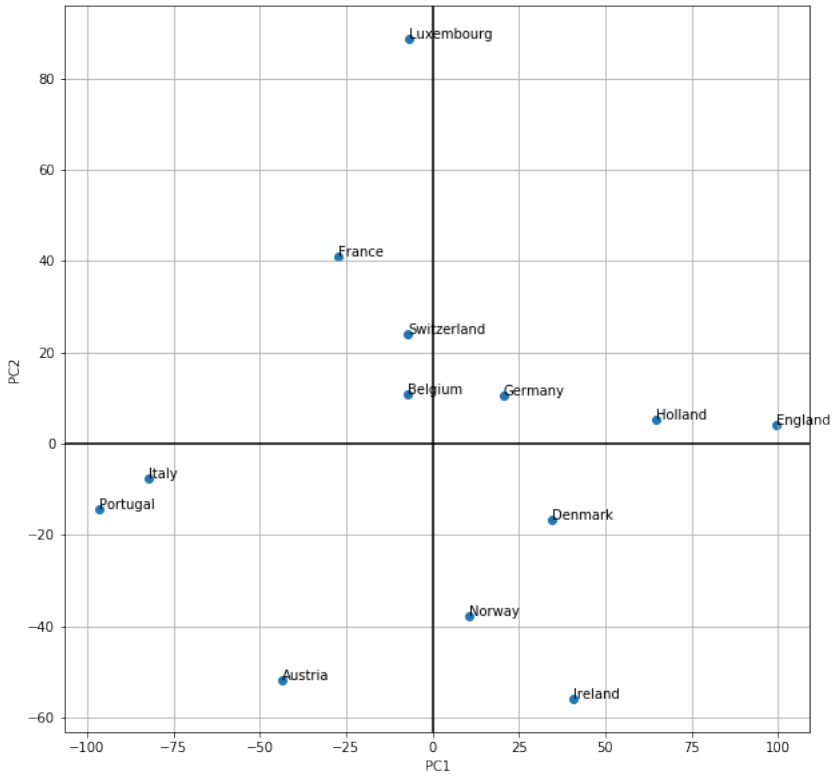
(3)

From the principal component loadings (Figure 1), we can see that PC1 is characterized by tinned fruit, jam and tin soup on one side, while garlic, olive oil and real coffee on the othe side. For PC2, it is characterized by garlic, tinned fruit and yoghurt on one side, while jam is on the other side.

(4)

From the new data plot (Figure 2), we can see that countries like England, Holland share similar food preference on tinned fruit, jam and tin soup. Italy and Portugal share food preferences on garlic, olive oil and real coffee. Luxembourg has very high preference on garlic, tinned fruit and yoghurt. Norway and Ireland share some preference on jam.


Garlic
Olive oil
Real coffee
Yoghurt
Oranges
Apples
Tinned fruit
Instant coffee
Butter
Biscuits
Powder soup
Margarine
Sweetener
Potatoes
Frozen veggies
Tin soup
Frozen fish
Tea
Crisp bread
Jam
PC1
PC2


Luxembourg
France
Switzerland
Belgium
Germany
Holland
England
Italy
Portugal
Denmark
Norway
Austria
Ireland
PC1
PC2