January 19 2020

ISyE 6740
Homework 1 Solution

1 Clustering

a Prove that using the squared Euclidean distance $||x^n - \mu^k||^2$ as the dissimilarity function and minimizing the distortion function, we will have

$$\mu^k = \frac{\sum_{n=1}^N r^{nk} x^n}{\sum_{n=1}^N r^{nk}} \tag{1}$$

That is, μ_k is the center of k-th cluster.

SOLUTION:

$$\begin{aligned} J &= \sum_{n=1}^N \sum_{k=1}^K r^{nk} ||x^n - \mu^k||^2 \\ &= \sum_{n=1}^N \sum_{k=1}^K r^{nk} (x^n - \mu^k)(x^n - \mu^k) \end{aligned} \tag{2}$$

The quadratic objective function J can be minimized by setting its derivative with respect to eack μ_k to zero.

$$\begin{aligned} \frac{\partial J}{\partial \mu^k} &= 0 \\ \sum_{n=1}^N 2 r^{nk} (x_n - \mu^k) &= 0 \\ \sum_{n=1}^N r^{nk} x^n - \mu^k \sum_{n=1}^N r^{nk} &= 0 \\ \mu^k &= \frac{\sum_{n=1}^N r^{nk} x^n}{\sum_{n=1}^N r^{nk}} \end{aligned} \tag{3}$$

b Prove that K-means algorithm converges to a local optimum in finite steps.

SOLUTION:
We are given N data points x^n $(n = 1, \ldots, N)$. The objective of k-means clustering is to partition the data set into k clusters, such that each cluster is as "tight" as possible. A clustering $\mathcal{C}$ $\{1, \ldots, N\} \rightarrow \{1, \ldots, k\}$ assigns one of k clusters to each point in the data set. Each cluster $k' \in \{1, \ldots, k\}$ is also associated with a center $\mu^{k'}$. If $\mathcal{C}(n)$ is the cluster in $\{1, \ldots, k\}$ to which $\mathcal{C}$ assigns input point n, the Euclidean distance between the point and its cluster center is $||x^n - \mu^{\mathcal{C}(n)}||^2$. The most common measure of the tightness (along with cluster centers μ) is the sum squared error (SSE), defined as

$$\sum_{n=1}^N ||x^n - \mu^{\mathcal{C}(n)}||^2 \tag{4}$$

Suppose that the algorithm proceeds from iteration t to iteration t+1. It suffices to show That $SSE\left(\mathcal{C}_{t+1}, \mu_{t+1}\right) < SSE\left(\mathcal{C}_t,\ \mu_t\right)$.

First, we show that $SSE\left(\mathcal{C}_{t+1}, \mu_t\right) < SSE\left(\mathcal{C}_t, \mu_t\right)$
and next, we show that $SSE\left(\mathcal{C}_{t+1}, \mu_{t+1}\right) \leq SSE\left(\mathcal{C}_{t+1}, \mu_t\right)$

The first step: $\mathcal{C}_t$ and $\mathcal{C}_{t+1}$ are different only if there is a point that finds a closer cluster center in μ_t than the one assigned to it by $\mathcal{C}_t$

$$SSE\left(\mathcal{C}_{t+1}, \mu_t\right) = \sum_{n=1}^{N} ||x^n - \mu_t^{\mathcal{C}_{t+1}(n)}||^2 < \sum_{n=1}^{N} ||x^n - \mu_t^{\mathcal{C}_t(n)}||^2 = SSE\left(\mathcal{C}_t, \mu_t\right) \tag{5}$$

The second step:

$$\begin{aligned} SSE\left(\mathcal{C}_{t+1}, \mu_{t+1}\right) &= \sum_{n=1}^{N} ||x^n - \mu_{t+1}^{\mathcal{C}_{t+1}(n)}||^2 \\ &= \sum_{k'=1}^{k} \sum_{n\in\{1,\ldots,N\}, \mathcal{C}_{t+1}(n)=k'} ||x^n - \mu_{t+1}^{\mathcal{C}_{t+1}(n)}||^2 \\ &\leq \sum_{k'=1}^{k} \sum_{n\in\{1,\ldots,N\}, \mathcal{C}_{t+1}(n)=k'} ||x^n - \mu_t^{\mathcal{C}_{t+1}(n)}||^2 \\ &= \sum_{n=1}^{N} ||x^n - \mu_t^{\mathcal{C}_t(n)}||^2 \\ &= SSE\left(\mathcal{C}_{t+1}, \mu_t\right) \end{aligned} \tag{6}$$

c For the following data (two moons), which of these three distance metrics below (if any) would successfully separate the two moons? Explain the reasoning

SOLUTION:
The **single linkage** distance metric, which measures the minimum distance between any two pairs of points from the two clusters will successfully separate the two moons.
Let L be the points in the left moon, and R be the points in the right moon, then from the image we can see that

$$\forall x \in L, \exists y \in L : ||x - y|| < ||x - z||, \forall z \in R \tag{7}$$

$$\forall x \in R, \exists y \in R : ||x - y|| < ||x - z||, \forall z \in L \tag{8}$$

Then, the single linkage will successfully separate the two moons.

2 Image compression using clustering

SOLUTION: (Any solution with reasonable results and explanations will be given credit)

1. **K-medoids Algorithm**
   For the K-medoids algorithm implementation, I chose K random data points (pixels here) as the initial cluster centers. Then I updated the cluster assignment for each pixel using the distance measure/dissimilarity parameter. I have used the following three distances metrics as the dissimilarity parameter for different implementations.
   Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{9}$$

Manhattan distance:

$$d(x,y) = \sum_{i=1}^{n} |x_i - y_i| \tag{10}$$

'inf'-distance/Chebychev Distance

$$d(x,y) = max_i|x_i - y_i| \tag{11}$$

Each pixel was then assigned to the cluster center to which it was least distant.
In the next step, I updated the center for each of the K clusters with the pixel value which was lying the closest (closeness calculated using the corresponding distance metric) to the arithmetically calculated centroid of the cluster. This was done only when the value of the distortion function with the new centers was less than its value with the previous centers. The iterations were stopped when either the $norm(c_{new} - c_{old}) \leq 10^{-6}$ or the distortion function ceased improving.

After trying out the above three dissimilarity parameters for different values of K, I made the following observations which is tabulated below.

K-medoids						
	Euclidean Distance		Manhattan Distance		Inf-Distance	
K	Time (s)	No of iterations	Time (s)	No of iterations	Time (s)	No of iterations
2	0.8932	8	0.4436	4	0.5613	5
3	1.1628	10	0.2375	2	0.591	5
4	1.1418	9	0.6348	5	0.2599	2
8	3.0033	21	0.5776	4	1.4353	10
10	2.7498	18	2.1348	14	1.819	12
16	3.9046	21	1.6605	9	0.5703	3
25	2.8379	12	3.1022	13	0.7228	3
32	5.5915	21	2.7184	10	1.3948	5

Figure 1: Dissimilarity parameter comparison

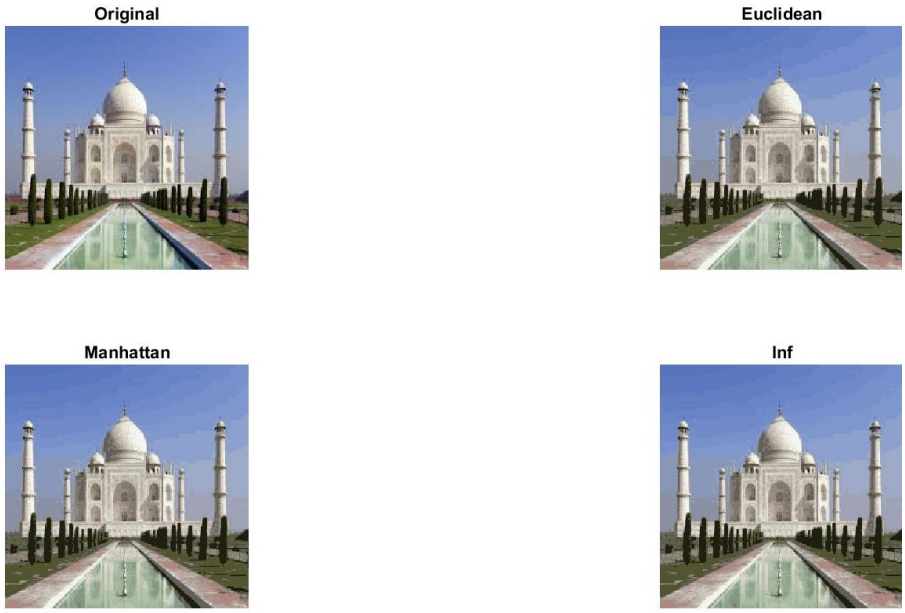

Figure 2: Output Pictures

We can see that the inf-distance metric outperformed both euclidean and manhattan distance metrics in terms of both the time and the number of iterations taken for convergence. Also there was no significant difference in the quality of the output generated by the three metrics. Therefore, I chose inf-distance as the dissimilarity parameter for the *mykmedoids.m* code. For reference purposes, I have attached an image with the list of all outcomes for $K = 32$, when there is a significant difference in the run time for the implementation of these three metrics and not any discernible differences in the output quality.

2. **Image Used**
Please find the image used for running the implementations below.

Figure 3: Image Used

3. **Implementation of K-medoids with different Ks**
Please find the table with the time taken for implementing K-medoids with different values of K in Figure 1. Although I observed that it usually takes longer time to run the implementation with

Eucledian and Manhattan distances when the K values were large, it cannot be strictly generalized. I could not infer any pattern for the dependence of time on K in the implementation using inf-distance metric.

4. **Effect of initial centroids**
Although there were minor changes in the pixel composition of the output picture for multiple implementations of K-medoids algorithm with different initial centroids, the differences were not very significant. However, we could see that the run time and the number of iterations varied with the change in initial centroids.
Also, we observed that initializing with an intentional poor assignments such as centroids which are very similar to each other (for eg.: $c0$=[255 254 255; 248 251 255] for $K = 2$ case), actually led to a considerable increase in the run time and the number of iterations needed for convergence.

5. **Implementation of K-means with different Ks and comparison with K-medoids**

K-means		
K	Time (s)	No of iterations
2	1.1674	12
3	2.345	23
4	4.2107	40
8	3.743	30
10	5.9611	44
16	12.1969	74
25	8.3431	40
32	34.4388	142

Figure 4: K-means

Please find the table with the time taken for implementing K-means with different values of K in Figure 4.
For smaller K values (such as $K = 2$), k-medoids algorithm was more effective in capturing the essential features than k-means. One can observe this in the beach figure attached below, where k-means failed to differentiate the water from the sand.

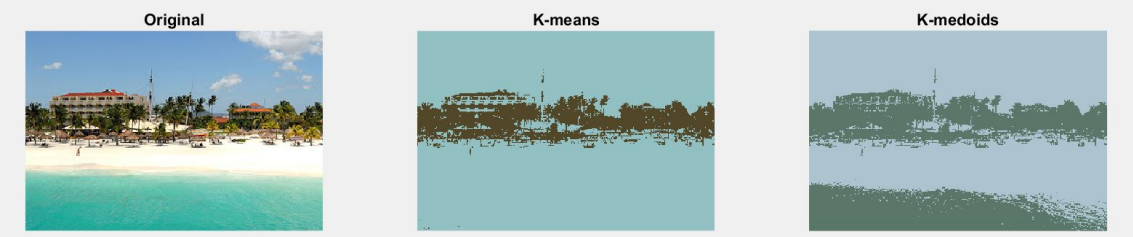

Figure 5: Output pictures for K=2

Also, the colors in the K-medoid outputs with higher K values looked more similar to the actual picture compared to the K-means output.
Further, I observed that K-medoids implementation took significantly lesser time (and iterations) than K-means implementation when larger K values were involved.