Student: Kien D. Vu

Professor: Dr. Yao Xie

Computational Data Analytics

ISYE-6740-OAN

Homework 1

Problem a:

[a-b] Given N data points $\mathrm{x}^n (n = 1, \ldots, N)$, K-means clustering algorithm groups them into K clusters by minimizing the distortion function over $\{r^{nk}, \mu^k\}$

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r^{nk} \|\mathrm{x}^n - \mu^k\|^2,$$

where $r^{nk} = 1$ if x^n belongs to the k-th cluster and $r^{nk} = 0$ otherwise.

Since clusters are independent to each other, setting the derivative of the distortion fucntion w.r.t μ^k to find the minumum

Thus, $2 \sum_{n=1}^{N} r^{nk}(x^n - \mu^k) = 0$

$$\Leftrightarrow \quad \mu^{\boldsymbol{k}} = \frac{\sum_{\boldsymbol{n}} r^{nk} x^n}{\sum_{\boldsymbol{n}} r^{nk}}$$

Problem b: K-mean clustering algorithm is convergence in finite steps because

First, there are at max K^{N} ways to cluster N data points to K clusters, which is a finite number

Second, the distortion function is always decreases after each iteration of K-mean algorithm

Problem c: When we use the bottom-up hierarchical clustering to realize the partition of data, the complete linkage distance metrics would most likely result in cluster most similar to those given by K-mean with second-order Manhattan distance function. Because, in 2nd-order Manhattan distance function, the dominant term is the farthest distance between 2 data points, which is similar to complete linkage.

Problem d: In 2-moon dataset, the single linkage can successfully separate 2 clusters. (This was experimented in Python)

Question 4:

1. K-Medoids Implementation:
    - **Initialization:** Randomly choose k datapoints in n samples (n pixels of the image)
    - **Distance:** Euclidian distance
    - **Iteration:**
        - Assign each datapoints to each nearest medoid.
        - Calculate the sum of distance between each datapoint in the same cluster with its cluster medoid. Calculate the sum of distance for all clusters
        - Randomly choose another point within assigned clusters as new medoid, calculate the sum of distance between each data points in the same cluster with new medoid.
        - If the new sum of distance is less than the old sum of distance, assign medoid := new_medoid. Re-iterate from step 1
    - **Stopping criteria:**
        - Max iteration = 500
        - Each assigned cluster remains unchanged after iteration
2. As the number of clusters increases, the compressed image look more like the original image, but also the running time increases.
3. The K-medoid initialization does affect the final result. Choosing K medoids too close to each other causes the image to look bad and it takes longer to run

4. K-Mean Implementation:
    - **Initialization:** Randomly choose k datapoints in n samples (n pixels of the image)
    - **Distance:** Euclidian distance
    - **Iteration:**
        - Assign each datapoints to each nearest center.
        - Calculate the mean of each cluster and assign mean point as new cluster center
    - **Stopping criteria:**
        - Each assigned cluster center remains unchanged after iteration
5. As the number of clusters increases, the compressed image looks more like the original image, but also the running time increases.
6. The K-means initialization does affect the final result. Choosing K mean too close to each other causes the image to look bad and it takes longer to run