Homework 2

k-means on Python (100 points)

Note: This problem should be implemented in Python.

* * *

This problem will help you understand the nitty gritty details of implementing clustering algorithms on Python. In addition, this problem will also help you understand the impact of using various distance metrics and initialization strategies in practice. Let us say we have a set $\mathcal{X}$ of n data points in the d-dimensional space $\mathbb{R}^d$. Given the number of clusters k and the set of k centroids $\mathcal{C}$, we now proceed to define various distance metrics and the corresponding cost functions that they minimize.

Euclidean distance Given two points A and B in d dimensional space such that $A = [a_1, a_2 \cdots a_d]$ and $B = [b_1, b_2 \cdots b_d]$, the Euclidean distance between A and B is defined as:

$$\|a - b\| = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \tag{1}$$

The corresponding cost function ϕ that is minimized when we assign points to clusters using the Euclidean distance metric is given by:

$$\phi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} \|x - c\|^2 \tag{2}$$

Note, that in the cost function the distance value is squared. This is intentional, as it is the squared Euclidean distance the algorithm is guaranteed to minimize.

Manhattan distance Given two random points A and B in d dimensional space such that $A = [a_1, a_2 \cdots a_d]$ and $B = [b_1, b_2 \cdots b_d]$, the Manhattan distance between A and B is defined as:

$$|a - b| = \sum_{i=1}^{d} |a_i - b_i| \tag{3}$$

The corresponding cost function ψ that is minimized when we assign points to clusters using the Manhattan distance metric is given by:

$$\psi = \sum_{x \in \mathcal{X}} \min_{c \in \mathcal{C}} |x - c| \tag{4}$$

Iterative k-Means Algorithm: We learned the basic k-Means algorithm in class which is as follows: k centroids are initialized, each point is assigned to the nearest centroid and the centroids are recomputed based on the assignments of points to clusters. In practice, the above steps are run for several iterations. We present the resulting iterative version of k-Means in Algorithm 1.

Iterative k-Means clustering on Python: Implement iterative k-means using Python. Please use the dataset from *data* within the bundle for this problem.

The folder has 3 files:

1. *data.txt* contains the dataset which has 4601 rows and 58 columns. Each row is a document represented as a 58 dimensional vector of features. Each component in the vector represents the importance of a word in the document.

2. *c1.txt* contains k initial cluster centroids. These centroids were chosen by selecting $k = 10$ random points from the input data.

Algorithm 1 Iterative k-Means Algorithm

```
1: procedure ITERATIVE k-MEANS
2:     Select k points as initial centroids of the k clusters.
3:     for iterations := 1 to MAX_ITER do
4:         for each point p in the dataset do
5:             Assign point p to the cluster with the closest centroid
6:         end for
7:         Calculate the cost for this iteration.
8:         for each cluster c do
9:             Recompute the centroid of c as the mean of all the data points assigned to c
10:        end for
11:    end for
12: end procedure
```

3. *c2.txt* contains initial cluster centroids which are as far apart as possible, using Euclidean distance as the distance metric. (You can do this by choosing 1st centroid c1 randomly, and then finding the point c2 that is farthest from c1, then selecting c3 which is farthest from c1 and c2, and so on).

Set number of iterations (MAX_ITER) to 20 and number of clusters k to 10 for all the experiments carried out in this question. Your driver program should ensure that the correct amount of iterations are run.

(a) Exploring initialization strategies with Euclidean distance [50 pts]

1. **[25 pts]** Using the Euclidean distance (refer to Equation 1) as the distance measure, compute the cost function $\phi(i)$ (refer to Equation 2) for every iteration i. This means that, for your first iteration, you'll be computing the cost function using the initial centroids located in one of the two text

files. Run the k-means on *data.txt* using *c1.txt* and *c2.txt*. Generate a graph where you plot the cost function $\phi(i)$ as a function of the number of iterations $i = 1, ..., 20$ for *c1.txt* and also for *c2.txt*. You may use a single plot or two different plots, whichever you think best answers the theoretical questions we're asking you about.

2. **[25 pts]** What is the percentage change in cost after 10 iterations of the K-Means algorithm when the cluster centroids are initialized using *c1.txt* vs. *c2.txt* and the distance metric being used is Euclidean distance? Is random initialization of k-means using *c1.txt* better than initialization using *c2.txt* in terms of cost $\phi(i)$? Explain your reasoning.

   (Hint: to be clear, the percentage refers to (cost[0]-cost[10])/cost[0].)

(b) Exploring initialization strategies with Manhattan distance [50 pts]

1. **[25 pts]** Using the Manhattan distance metric (refer to Equation 3) as the distance measure, compute the cost function $\psi(i)$ (refer to Equation 4) for every iteration i. This means that, for your first iteration, you'll be computing the cost function using the initial centroids located in one of the two text files. Run the k-means on *data.txt* using *c1.txt* and *c2.txt*. Generate a graph where you plot the cost function $\psi(i)$ as a function of the number of iterations $i = 1, ..., 20$ for *c1.txt* and also for **c2.txt**. You may use a single plot or two different plots, whichever you think best answers the theoretical questions we're asking you about.

   (Hint: This problem can be solved in a similar manner to that of part (a). Also note that It's possible that for Manhattan distance, the cost do not always decrease. K-means only ensures monotonic decrease of cost for squared Euclidean distance. Look up K-medians to learn more.)

2. **[25 pts]** What is the percentage change in cost after 10 iterations of the K-Means algorithm when the cluster centroids are initialized using *c1.txt* vs. *c2.txt* and the distance metric being used is Manhattan distance? Is random initialization of k-means using *c1.txt* better than initialization using *c2.txt* in terms of cost $\psi(i)$? Explain your reasoning.

What to submit:
(i) Upload the code for (a) and (b) to gitee
(ii) A plot of cost vs. iteration for two initialization strategies [(a)]
(iii) Percentage improvement values and your explanation [(a)]
(iv) A plot of cost vs. iteration for two initialization strategies [(b)]
(v) Percentage improvement values and your explanation [(b)]