Python Homework 4

Name**: Mingyue Ouyang** StudentID**: 174002158**

Q: In your write up, you need to (1) describe your algorithm and do complexity analysis (time and space analysis in big O notation). (2) What is your metric to measure the distances among points? (3) How do you choose initial centroid locations? (4) How do you assign a data point to a centroid in your program iteration? And eventually (5) what criteria you use to choose the best K value to classify your data set.

A:

1) The algorithm implemented in this homework mainly include three parts. Details are as follows:

Initialize K Centroids

Normalization

Read Data

Assign Data Points to Clusters

Update K Centroids

Assign Data Points to Clusters

No

If rate < 1%

Yes

K Clustering Result

**Note:**

**rate = n / total**

* **n: Number of data points whose cluster change after k centroids update.**
* **total: Total number of data points.**

~~First, use the constructor to initialize the cluster and parameters. We ask users to input four parameters: initial k, filename(dataset), attributes and the tolerance for iteration (0.01 in this homework).~~

**Read Data:**

Create a dictionary, input the attributes, let attribute be the keys, and lists as values. Put value of each data point’s attributes sequentially in those lists.

**Normalization:**

To get rid of outlier’s effect, medium normalization is deployed.

**Initialize the centroids:**

Randomly select k different data points as centroids. A list called existRand is created to keep track of selected data points in case of the repetition.

**Assign data points to clusters:**

Loop through all data points and assign each data point to a cluster whose centroid is the nearest one to the data point in all k centroids. Euclidean distance is deployed to represent distance between a data point and a centroid.

**Update centroids and Re-assign data points to clusters:**

Update the centroid of each cluster by calculating mean value of data points in the same cluster. And then assign data points to clusters with the updated centroids. Keep implementing these two steps before 99% data points’ cluster don’t change anymore.

**Calculate SSE:**

SSE is updated during the process of assigning and re-assigning data points to clusters.

You can find Big O notations in comments of codes.

2) Euclidean distance is deployed to measure distance among data points.

3) The initial centroids are randomly selected from the input data points.

4) Loop through all data points and assign each data point to a cluster whose centroid is the nearest one to the data point in all k centroids. Euclidean distance is deployed to represent distance between a data point and a centroid.

5) We run this code for every k from 2 to 20 (code 记得改), and choose the k with the minimum SSE value which is the optimal solution. The criteria I use to choose the best K value is SSE. The best k value has the minimum SSE.