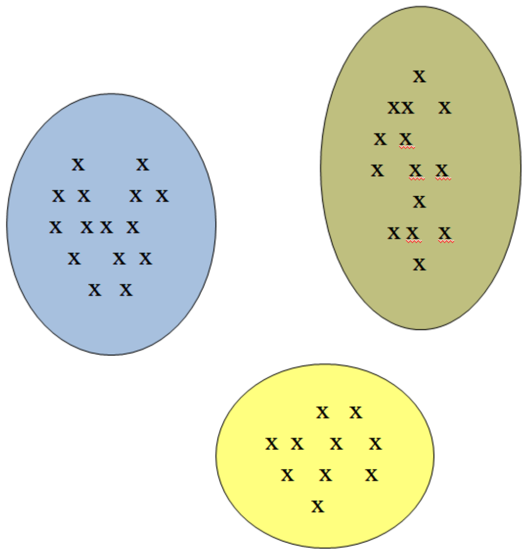
DATA-51100: Statistical Programming
Programming Assignment 2 – k-Means Clustering

Introduction

Clustering is a process of identifying groupings (i.e. *clusters*) within the data. For example, the figure below shows three clusters of two-dimensional data points (X's):



Clustering has many applications, including inferring population structures from genetic data, recognizing communities within social networks, or segmenting of customers for market research.

One of the most popular algorithms for performing clustering is the *k*-means method. The algorithm depends on the notion of *distance* between two points. For points with only one dimension (just single values), we can define the distance between two points p and q as

$$d(p, q) = |p - q|$$

The k-means algorithm will work by placing points into clusters and computing their *centroids*, which is defined as the average of the data points in the cluster. Specifically, the algorithm works as follows:

1. Pick k, the number of clusters.
2. Initialize clusters by picking one point (centroid) per cluster. For this assignment, you can pick the first *k* points as initial centroids for each corresponding cluster.
3. For each point, place it in the cluster whose current centroid it is nearest.
4. After all points are assigned, update the locations of centroids of the k clusters
5. Reassign all points to their closest centroid. This sometimes moves points between clusters.
6. Repeat 4,5 until convergence. Convergence occurs when points don't move between clusters and centroids stabilize.

Requirements

You are to create a program using Python that does the following:

1. Asks the user for the number of clusters. This is the parameter *k* that will be used for *k*-means.
2. Reads the input file (prog2-input-data.txt) and stores the points into a list
3. Applies the k-means algorithm to find the cluster for each point.
4. Displays the points that each cluster contains after each iteration of the algorithm
5. Writes the final cluster assignments to the screen and the output file (prog2-output-data.txt).

YOU CANNOT USE ANY PYTHON PACKAGES FOR THIS PROGRAM (NUMPY, PANDAS, …) - NO IMPORT STATEMENTS.

Additional Requirements

1. The name of your source code file should be `kMeans.py`. All your code should be within a single file.
2. Your code should follow good coding practices, including good use of whitespace and use of both inline and block comments.
3. You need to use meaningful identifier names that conform to standard naming conventions.
4. At the top of each file, you need to put in a block comment with the following information: your name, date, course name, semester, and assignment name.
5. The output of your program should **exactly** match the sample program output given at the end. That is, for same input, it should generate the same output. Note that I may use other test cases for grading your program and your code needs to work correctly in all cases.

Data File Format

Let N be the number of points and Pi to be the value of point i. The input file should be of the following format:

```
P1
P2
…
PN
```

Example:

```
1.2
2.1
4.56
2.113
2.2
```

The name of the input file is always:

`prog2-input-data.txt`

What to Turn In

You will turn in a **screenshot of your output** and a **single kMeans.py** file using BlackBoard.

HINTS

- Make use of list comprehensions for reading lines from a file and then converting the strings into a list of floats.
- Use pwd() to check the directory where you should place your input file.
- Use a dict data structures for storing centroids and clusters. The centroids dict will be a mapping from cluster number to centroids. The clusters dict will be a mapping from cluster number to a list of points in the cluster.

Sample Program Output

```
DATA-51100, [semester] [year]
NAME: [put your name here]
PROGRAMMING ASSIGNMENT #2

Enter the number of clusters: 5

Iteration 1
0 [1.8]
1 [4.5, 6.5]
2 [1.1, 0.5]
3 [2.1, 3.2]
4 [9.8, 7.6, 11.32]

Iteration 2
0 [1.8, 2.1]
1 [4.5, 6.5]
2 [1.1, 0.5]
3 [3.2]
4 [9.8, 7.6, 11.32]

Iteration 3
0 [1.8, 2.1]
1 [4.5, 6.5]
2 [1.1, 0.5]
3 [3.2]
4 [9.8, 7.6, 11.32]

Point 1.8 in cluster 0
Point 4.5 in cluster 1
Point 1.1 in cluster 2
Point 2.1 in cluster 0
Point 9.8 in cluster 4
Point 7.6 in cluster 4
Point 11.32 in cluster 4
Point 3.2 in cluster 3
Point 0.5 in cluster 2
Point 6.5 in cluster 1
```

Output File Contents

```
Point 1.8 in cluster 0
Point 4.5 in cluster 1
Point 1.1 in cluster 2
Point 2.1 in cluster 0
Point 9.8 in cluster 4
Point 7.6 in cluster 4
Point 11.32 in cluster 4
Point 3.2 in cluster 3
Point 0.5 in cluster 2
Point 6.5 in cluster 1
```