Richard Chen

**K-Means Write Up**

**Part 1 Approach:**

In writing my K-Means algorithm, I realized that the difficult part was not the actual algorithm itself, but making it dynamic such that will find n-number of clusters. To circumvent the issue of possibly having to create 50 lists for part 2, I used classes, two-dimensional lists, and lots of for loops to cycle through my lists without having to worry about the index or size of my lists.

Essentially, in a K-Means algorithm, every coordinate must belong to the mean that it is closest to, which I interpreted in Python as a list. For K = 3, there are three means that the algorithm converges to, and thus, three lists of closest coordinates. To work with dynamic K values, I created a class called <Mean> which not only holds not only the <Float> data type for means, but also the <List> data type for a list of closest coordinates. Thus, every K-mean has a corresponding list of closest coordinates, and I don't have to worry about the number of lists I have to create for an arbitrary K. Now I can finally write the algorithm.

Since my algorithm uses lots of lists, here is a legend of what every list is:

- **coordinates_list** - A two-dimensional list that contains the coordinates of "clusters.dat". It contains a list of coordinates [x, y, z]
- **coordinates** - a list of the [x, y, z] coordinates. This list is of length 3
- **mean_list** - A list of <Mean>s. Within each <Mean> is a mean, and a list of its closest coordinates. Initially in the algorithm, each <Mean> is a randomly selected data point in coordinates_list, and has an empty closest coordinates list, since no coordinates have been assigned yet.
- **only_means** - A list of my means. Not to be confused with mean_list, only_means is of data type <Float>, and I made this list in order to compare it to the calculated new means in only_new_means.
- **distance_list** - A list that contains distances of all the means to an arbitrary coordinate.
- **mean.getCloseCoords()** - In the <Mean> I am referencing, mean.getCloseCoords is a list of the closest coordinates of that mean. In the class, it is called **close_coords**.
- **only_new_means** - A list that contains my calculated new means. It is useful in comparing it to the old means, which is "only_means".

First, in my algorithm, from my **mean_list**, I create **only_means**. Then, I cycle through **coordinates_list**, and for each **coordinates** in **coordinates_list**, I create a list of distances (**distance_list**), from that coordinate, to all the means in **mean_list**. I then find the shortest distance, which would have matching indexes with the corresponding <Mean> in mean_list. I then append that coordinate to **close_coords** in the respective <Mean>. Once I cycle through all the coordinates, I calculate the new means from close_coords in each <Mean>, which then gets appended to **only_new_means**. I then compare **only_new_means** and **only_means**, and if they aren't the same, I would reset close_coords in each <Mean>, update the mean in each <Mean>, set the old means list to the new means list, and go through the loop until the two lists of means are the same. I would then return **mean_list.** Using classes and for loops, I never once had to create extra lists to hold coordinates, or count indexes of the lists I am cycling through.

To print each cluster in a different color, since the coordinates in each cluster are altogether in one list, when outputting the clusters to a .txt file, each cluster had its corresponding cluster number printed beside it. This helps distinguish to me and to MATlab, which coordinate belongs to which cluster.

In MATlab, "scatter3" was used to create a 3D scatter plot of the clusters, with columns 1, 2, and 3 as x, y, and z. Column 4 is the cluster number that every coordinate has. Using the "filled" command in "scatter3", different cluster numbers correspond with different colors.

**Part 2 Approach**

The approach in Part 2 is identical to that in Part 1. The only thing that needed to be modified is that instead of 3 dimensions, there are 18 dimensions, and thus, **coordinates** has a length of 19, where the extra index is for the Gene ID. My distance function had to be also modified, and I had to also worry about some indexing, as the first index of **coordinates** was not a <Float>, but a <String>. In addition, in outputting the clusters, I was not outputting the coordinates for each cluster, but rather the gene ID. In addition, using some strange series of for loops, I printed each cluster in a column, then used the Gene Functional Classification Tool's Multi-List to analyze all of the clusters at once.

**Part 1 Data**

Enter a file name for K-Means Clustering: clusters.dat

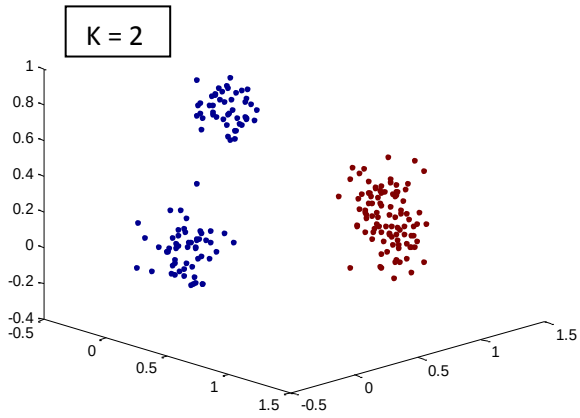
K = 2
Converged Mean
[0.8083405989795921, 0.9271132945918369, 0.15771767438265308]
[0.09294083463372553, 0.1805584581990196, 0.3417478407352941]

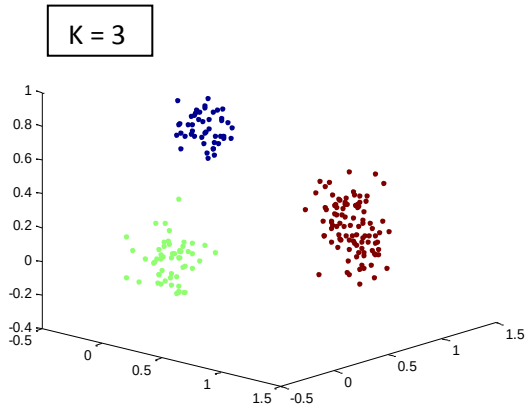
K = 3
Converged Mean
[0.9308105776595744, 0.8748854208510637, 0.06339761660638298]
[0.6954761088235293, 0.9752448645098037, 0.24464008056862743]
[0.09294083463372553, 0.1805584581990196, 0.3417478407352941]

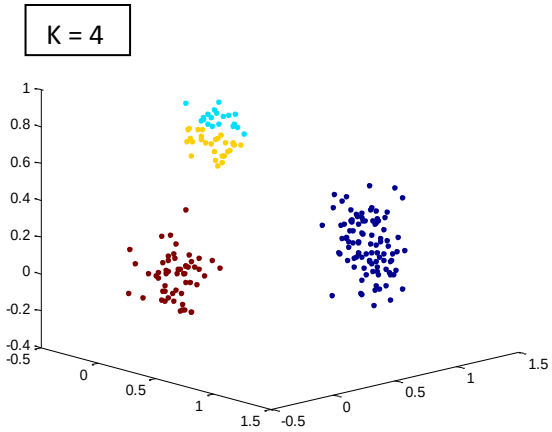
K = 4
Converged Mean
[0.07948555326170215, 0.36456535787234046, 0.7259523623404258]
[0.940004694888889, 0.8782013217777775, 0.061833950211111106]
[0.6965503288679243, 0.9686423281132075, 0.23912838358490568]
[0.10443898416981819, 0.023316198478181813, 0.013427613181818182]

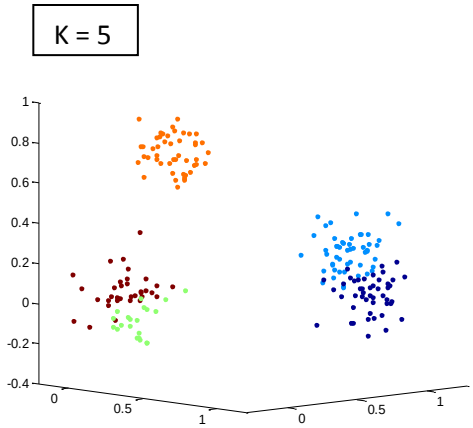
K = 5
Converged Mean
[0.1105231648888889, -0.020183978138888888, -0.10921639283333334]
[0.6965503288679243, 0.9686423281132075, 0.23912838358490568]
[0.07948555326170215, 0.36456535787234046, 0.7259523623404258]
[0.10147911246864863, 0.044478446562162155, 0.07309226475675676]
[0.940004694888889, 0.8782013217777775, 0.061833950211111106]

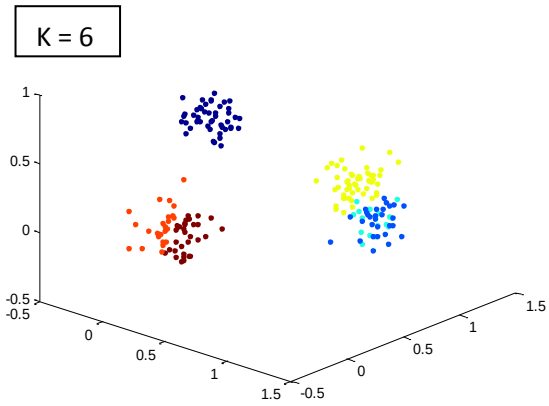
K = 6
Converged Mean
[0.07948555326170215, 0.36456535787234046, 0.7259523623404258]
[0.6693507162962964, 0.8913281144444445, 0.19558072677777774]
[0.10443898416981819, 0.023316198478181813, 0.013427613181818182]
[0.9040661735714285, 0.9902223257142859, 0.1335295617857143]
[0.952421516875, 0.8444514828125002, 0.036593101609374984]
[0.7204197756, 1.0362273508, 0.28541037399999997]




From these graphs, we see that K is most optimal at K = 3, as we can distinctively see three different clusters, which is best represented in having three converged means.

**Part 2 Data**

Using Gene Functional Classification Tool's Multi-List, each cluster can be analyzed for matching genes.