K-Mean Python Implementation

K-mean clustering is a machine learning algorithm that given the number of clusters, finds the best center points for all cluster. K-mean works on the logistics that each point should be closer to its cluster's center than to other clusters.

Procedure

The algorithm begins by picking K random points from the dataset as the clusters center. Next each point will be assigned to one of the K clusters based on shortest distance to the cluster center picked randomly. As a result, each cluster will have a set of points assigned to it. For each set of points the center of gravity will be calculated and the center will shift to that point. The procedure will repeat itself over and over, each time shifting the center and reassigning points to the new closest centers. When the points will stop shifting, the centers are returned as the minimum. The issue with this is that K-mean may find a local minimum instead of a global minimum. To measure the quality of our cluster we calculate the Variation of the results. To do that we add all the distances of each point to its cluster center. To find the best results, the algorithm will have to run a number of times and find the result with the lowest variation.

Implementation

All the steps stated above are implemented using Python. The only packages used for computational reasons are: numpy, random, and scipy.spatial.

The data:

The data is extracted from a txt file where the first line is 2 integers a)the K value b) the dimensions of the points. Following the first line every line represents a point in m-dimensions where each dimension is separated by space. For example: this is a set of 3-dimensional points with k=6:

```
6 3
8.3 19.5 6.2
9.3 0.5 15.9
23.8 2.6 12.7
1.7 7.1 0.3
…
…
```

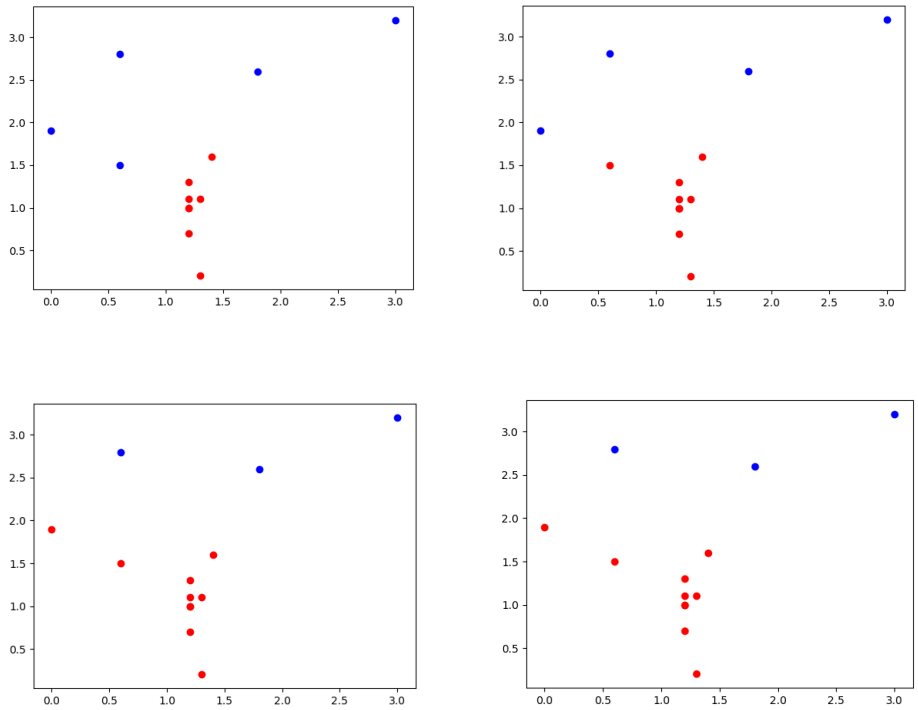
The import_data(path) method takes the path this txt file is located and outputs the K and a matrix where each row represents a point and each column represents a different axis:

```
[[8.3],[19.5],[6.2]]
[[9.3],[0.5],[15.9]]
[[23.8],[2.6],[12.7]]
[[1.7,[7.1],[0.3]]
```

The Algorithm:

The algorithm works with 3 methods, Driver(), Kmean(),and find_center(). Once points are imported they are sent into ***driver***: This method only calls the K-mean algorithm n amount of times keeping the one with the lowest variation. ***Kmean:*** this function picks k random points to be the random cluster centers, and assigns each point to a center cluster. With each of the sets of points of each cluster, the method calls find_center() to change the location of the cluster centers toward the center of gravity of the assigned points. It will then check if the cluster centers changed returning them if they didn't and reassigning cluster sets and calling find_center() over and over again if they did. ***Find_center:*** This method uses numpy to divide the set of points into m arrays where each array is an axis of the m dimensional points. Now it will use numpy.mean to find the mean of each position and combine them to get the center of gravity.

Visualization:



As shown in the images, the first iteration picked clusters that did not minimize the variation, the center shifted twice until reaching a minimum, on the fourth iteration we see that the clusters remained the same knowing that we are done.

3D plotting

I experimented with different methods of plotting. For 2D it was rather easy using matplotlib.pyplot (plotcluster method). For 3 dimensional points, however, it was much harder to visualize without being able to interact and rotate the graph. I found a library called plotly that allows this interaction. By creating an account for this open source library, I generated an API key that sends python data into a desired plot that allows an interactive visualization (ThreeDplotting method).

See plot at: https://plot.ly/~Tkeren/0/#/ (better works on Microsoft edge or at least not chrome for some reason)