kMeans Clustering Report

By: Soliyana Negash

March 10th, 2024

Introduction

In this report, I'm presenting my implementation of the kMeans algorithm in Python. This implementation follows the basic kMeans algorithm, which includes assigning data points to the closest cluster centroid iteratively, and updating those centroids based on the mean of the data points. This continues until convergence—where the centroids can no longer significantly change—or until the maximum number of iterations I set is met.

We demonstrate this algorithm through multiple datasets such as, twoCircles.txt, twoEllipses.txt, fourCircles.txt, t4.8k.txt, and iris.txt. The goal here is to discover distinct clusters within these datasets.

First, I will provide the brief objective of this report. Then, I'll present the results--after explaining the preparation--and discuss the insights gained from applying this algorithm to all the datasets. Finally, it'll conclude with a summary of my findings.

Objective

One of the main objectives of this project is to try and understand the steps involved in the kMeans clustering. By showing the way the algorithm works, we'll gain insight into how kMeans divides data into distinct clusters and refines their centroids. Another main objective is to be able to learn the limitations of kMeans clustering on non-globular shape clusters. KMeans isn't the best when having datasets with clusters of irregular shapes. Through this experiment using datasets with diverse ranges, we highlight the challenges the kMeans faces when trying to accurately divide data. The last main objective is investigating how random initialization of kMeans affects the quality of the clustering. The initialization of the centroids play a major role in the convergence and effectiveness of the algorithm. When seeing the effects of random initialization on clustering outcomes, we'll be able to gain insights on the algorithm.

Data Cleaning and Preprocessing

fourCircles.txt Preprocessing

With this dataset, it was originally taken in as a string. I hadn't noticed that until I tried plotting the points and it wasn't taking the values in. Once I had figured that out, when plotting the points I just converted the values into floats. I should've converted them beforehand, but decided it was only really necessary when plotting. When attempting to cluster this dataset, I was running into an issue with the fact that all the data values were in 1 column, which didn't allow me to cluster properly. I hadn't checked the shape of the dataset beforehand, which would've shown me the 1 column. So, I had to go back to when I originally imported the dataset, and split it into two columns, and converted it to float there too since I needed floating number to cluster.

iris.txt, twoCircles.txt, twoEllipses.txt, t4.8k.txt Preprocessing:

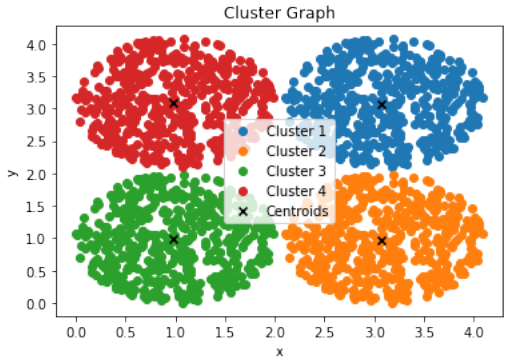
Also learning from my mistake with fourCircles.txt, I broke up each column within the dataset to be their own and converted it from string to float values. This allowed to me to process this dataset faster than

fourCircles.txt, I also no longer needed to convert it to a float when plotting since it was converted at import.

Exploratory Data Analysis (EDA)

fourCircles.txt Analysis:

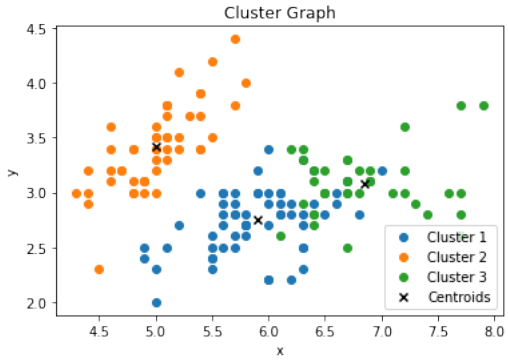
The graph below (*Figure 1*) shows 4 very well rounded, almost oval like clusters. There is a good amount of separation between each cluster, not much overlapping between them or many outliers. Each centroid seems very centered to each cluster, none of them very skewed. The centroids seem to be very surrounded by their data points, almost dense, but they do have some holes near the centroids. Everything seems to be very clean and organized, I would like to think it's because there wasn't much variation within the dataset. This dataset was all 1s and 0s, which would contribute to the neatness of the clustering.



[Figure 1: fourCircles.txt Clustered Scatter Plot]

iris.txt Analysis:

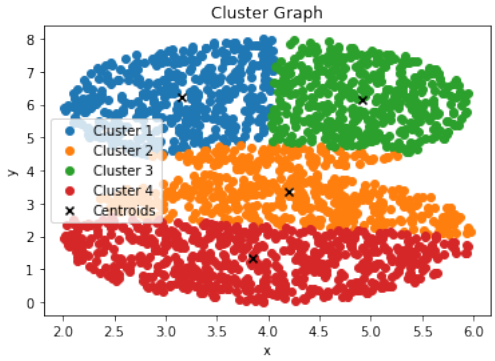
The graph below (*Figure 2*) shows 3 very sparse, all over the place clusters. There is some separation between the clusters, Cluster 1 and Cluster 3 crossover a bit, whereas Cluster 2 is in its own space. There looks to be some outliers within each cluster, just a few data points that seem more further out than the rest. It looks as though each centroid is correctly centered, since the data points are sparse it's kind of hard to tell. Since there are less values within this dataset, it makes sense that the data points are more spread out and it's not very dense. It could also be because there is a little bit more range when it comes to this dataset, from about 0-6.



[Figure 2: iris.txt Clustered Scatter Plot]]

twoCircles.txt Analysis:

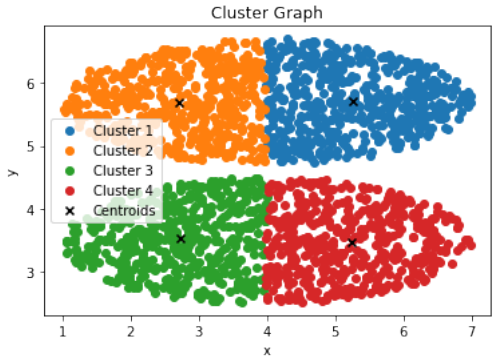
The graph below (*Figure 3*) shows 4 compacted clusters. I think the shape of these clusters are a little funny, and maybe the more inconsistent shapes out of all the datasets. There is separation between each cluster, but they're bordering each other so closely that it looks as though they're going to overlap. Each centroid look very centered, with their data points surrounding them closely. There are a lot of values in this dataset which makes sense why the datapoints are so dense, there is a little more range in values from 0-5, which is more than fourCircles.txt. Looking at the shape overall, it looks like two massive ovals. The top oval is mainly Cluster 1 and Cluster 3, with some of Cluster 2, whereas the bottom oval is just Cluster 2 and Cluster 4. It's interesting that Cluster 2 is the only one that's in both.



[Figure 3: twoCircles.txt Cluster Scatter Plot]

twoEllipses.txt Analysis:

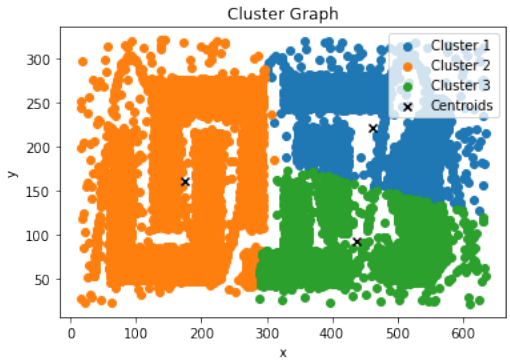
The graph below (*Figure 4*) shows 4 compacted, dense clusters. Pretty similar to *Figure 3* above, there seems to be two massive ovals, but this time each oval is evenly split for each cluster. There is a good amount of separation going on between Cluster 1 & 2 and Clusters 3 & 4. Between each of those clusters groups though, there is a lot overlapping going on. Every centroid looks correctly centered, with the data points very strongly surrounding them. This dataset has a lot of values within it, which explains the density of it. The clusters look very wide which can probably be explained by the range of about 1-6.



[Figure 4: twoEllipses.txt Cluster Scatter Plot]

t4.8k.txt Analysis:

The graph below (*Figure 5*) shows 3 really complicated clusters. It almost looks like writing in the middle of all the clusters. There is almost a rectangular shape to each of them. There is some separation between the clusters. Clusters 1 and 3 seem to have a good amount of overlap, also with Cluster 2 and 3. Every centroid looks correctly centered, but not all are surrounded by their data points. This dataset has a great number of values with a massive range of about 50-600. I think the sheer size of the dataset explain why it's so dense to where you can't even see some datapoints.



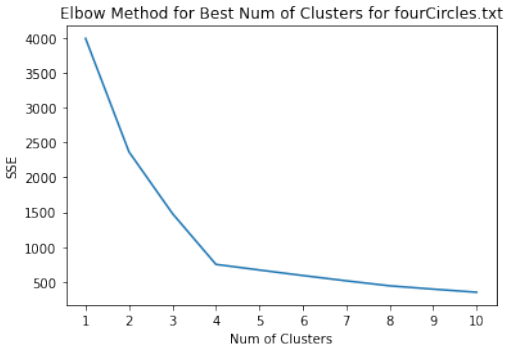
[Figure 5: t4.8k.txt Cluster Scatter Plot]

Results

fourCircles.txt Results:

I did 4 clusters for this dataset because when I plotted it using the elbow method, there was a major bend at x = 4 (*Figure 6*). There were two different means for each cluster because the dataset was split into two columns, Cluster 1 being 3.07 and 3.09, Cluster 2 being 3.07 and 0.99, Cluster 3 being 0.97 and 0.99, and Cluster 4 being 0.97 and 3.09. The size for this dataset was 500 data points for each cluster. Though I cannot see all of the final cluster assignments because there are so many data points, it was overall [222…000]. And the total number of iterations was 3, with a delta of 0.00031 (*Figure 7*). The cluster sizes all being the same shows a really balanced distribution of the data across all the clusters. Also, only 3

iterations for the kMeans algorithm to converge indicates that this was a stabilized process, also proven by the low delta.




```
labels, centroids = kmeans(dataFourCircle.values,4)
makeGraphs(dataFourCircle, labels, centroids)
```

```
Final mean and size for each cluster:
Cluster 1: Mean = [3.07441235 3.08870518], Size (amount of data points) = 500
Cluster 2: Mean = [3.07441235 0.98870518], Size (amount of data points) = 500
Cluster 3: Mean = [0.97441235 0.98870518], Size (amount of data points) = 500
Cluster 4: Mean = [0.97441235 3.08870518], Size (amount of data points) = 500
Final Cluster Assignments: [2 2 2 ... 0 0 0]
Number of Iterations: 3
Final Delta: 0.00031910666682305683
```

[Figure 6: fourCircles.txt Elbow Graph] *[Figure 7: fourCircles.txt Clustering Results]*

iris.txt Results:

I did 3 clusters, as told, for this dataset. There were four different means for each cluster because the dataset was split into three columns, Cluster 1 being 5.9, 2.74, 2.39, 1.43, Cluster 2 being 2, 3.42, 1.46, 0.24, and Cluster 3 being 6.85, 3.07, 5.74, 2.07. The size for this dataset was 62 data points for Cluster 1, 50 data points for Cluster 2, and 38 data points for Cluster 3. The final cluster assignments were: [0 2 0 1 0 1 2 0 2 2 1 1 0 0 1 1 1 0 1 2 0 2 1 0 0 1 0 1 2 2 1 1 0 0 0 0 0 1 0 0 0 0 2 2 0 1 2 0 1 1 0 0 1 2 1 2 2 2 0 0 2 2 1 1 0 0 0 1 1 1 2 2 0 1 1 1 1 0 1 0 2 0 0 2 2 0 0 2 1 0 1 1 2 0 0 2 0 0 0 0 0 0 2 2 0 1 1 2 1 1 0 1 0 1 0 1 2 0 2 0 0 1 0 0 1 1 0 1 0 2 1 2 2 1 1 1 1 0 2 0 2 2 0 0 0 1 0 2 2 1]. And the total number of iterations was 2, with a delta of 0 (*Figure 8*). Different to fourCircles.txt, the distribution of data points wasn't even. Cluster 3 was given the least amount, which might explain it's overlap with Cluster 1. I've noticed a pattern now where the number of iterations seems to be 1 less than the number of clusters. This process was shown the be a stabilized one due to the delta being exactly 0.

```
labels, centroids = kmeans(dataIris.values,3)
makeGraphs(dataIris,labels,centroids)
```

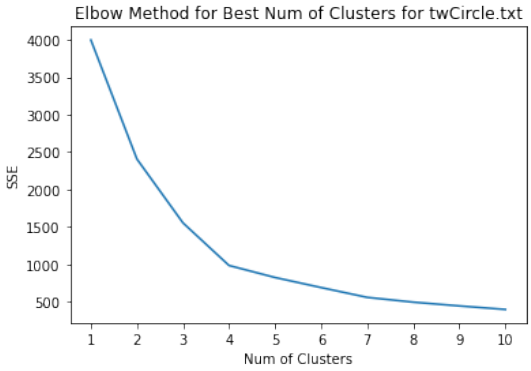
```
Final mean and size for each cluster:
Cluster 1: Mean = [5.9016129  2.7483871  4.39354839 1.43387097], Size (amount o
f data points) = 62
Cluster 2: Mean = [5.006 3.418 1.464 0.244], Size (amount of data points) = 50
Cluster 3: Mean = [6.85       3.07368421 5.74210526 2.07105263], Size (amount o
f data points) = 38
Final Cluster Assignments: [0 2 0 1 0 1 2 0 2 2 1 1 0 0 1 1 1 0 1 2 0 2 1 0 0 1
0 1 2 2 1 1 0 0 0 0 0
 1 0 0 0 0 2 2 0 1 2 0 1 1 0 0 1 2 1 2 2 2 0 0 2 2 1 1 0 0 0 1 1 1 2 2 0 1
 1 1 1 0 1 0 2 0 0 2 2 0 0 2 1 0 1 1 2 0 0 2 0 0 0 0 0 0 2 2 0 1 1 2 1 1 0
 1 0 1 0 1 2 0 2 0 0 1 0 0 1 1 0 1 0 2 1 2 2 1 1 1 1 0 2 0 2 2 0 0 0 1 0 2
 2 1]
Number of Iterations: 2
Final Delta: 0.0
```

[Figure 8: iris.txt Clustering Results]

twoCircles.txt Results:

I did 4 clusters for this dataset because when I plotted it using the elbow method, there was a noticeable bend at x = 4 (*Figure 9*). There were two different means for each cluster because the dataset was split

into two columns, Cluster 1 being 3.15 and 6.22, Cluster 2 being 4.12 and 3.36, Cluster 3 being 4.91 and 6.15, and Cluster 4 being 3.86 and 1.33. The size for this dataset was different for each dataset. For Cluster 1 it was 428 data points, Cluster 2 was 506 data points, Cluster 3 was 446 data points, and Cluster 4 was 620 data points. Though I cannot see all of the final cluster assignments because there are so many data points, it was overall [111…222]. And the total number of iterations was 3, with a delta of 0.00085 (*Figure 10*). The cluster sizes all being different shows a little imbalance with the distribution of the data points between each cluster, with Cluster 4 having the most data points. Again, only 3 iterations for the kMeans algorithm to converge indicates that this was a stabilized process, also proven by the low delta.



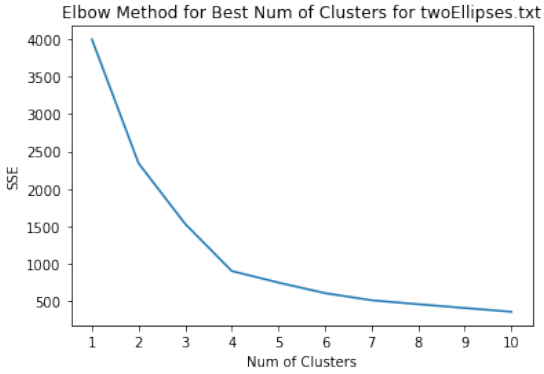

```
labels, centroids = kmeans(dataTwoCircle.values,4)
makeGraphs(dataTwoCircle, labels, centroids)
```

```
Final mean and size for each cluster:
Cluster 1: Mean = [3.15125372 6.21989316], Size (amount of data points) = 428
Cluster 2: Mean = [4.1817     3.36103079], Size (amount of data points) = 506
Cluster 3: Mean = [4.91460444 6.15219646], Size (amount of data points) = 446
Cluster 4: Mean = [3.85743206 1.3289077 ], Size (amount of data points) = 620
Final Cluster Assignments: [1 1 1 ... 2 2 2]
Number of Iterations: 3
Final Delta: 0.0008502678118243011
```

[Figure 9: twoCircles.txt Elbow Graph] *[Figure 10: twoCircles.txt Clustering Results]*

twoEllipses.txt Results:

I did 4 clusters for this dataset because when I plotted it using the elbow method, there was a notable bend at x = 4 (*Figure 11*). There were two different means for each cluster because the dataset was split into two columns, Cluster 1 being 5.26 and 5.7, Cluster 2 being 2.71 and 5.68, Cluster 3 being 2.71 and 3.53, and Cluster 4 being 5.23 and 3.46. The size for this dataset was different for each dataset. For Cluster 1 it was 511 data points, Cluster 2 was 289 data points, Cluster 3 was 516 data points, and Cluster 4 was 484 data points. Though I cannot see all of the final cluster assignments because there are so many data points, it was overall [323…110]. And the total number of iterations was 3, with a delta of 0.00033 (*Figure 12*). The cluster sizes all being different shows a little imbalance with the distribution of the data points between each cluster, but looking at the graph, it's not like you could tell. Again, only 3 iterations for the kMeans algorithm to converge indicates that this was a stabilized process, also proven by the low delta.



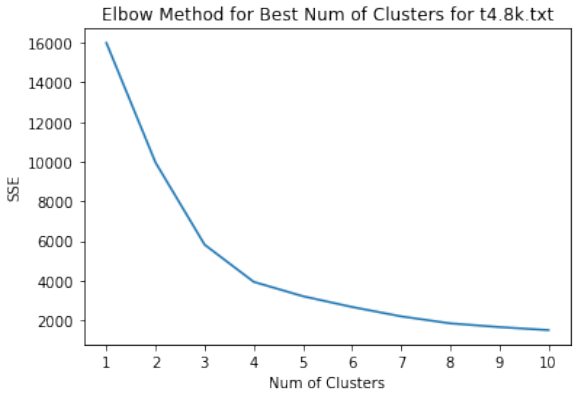

```
labels, centroids = kmeans(dataTwoEllipse.values,4)
makeGraphs(dataTwoEllipse, labels, centroids)
```

```
Final mean and size for each cluster:
Cluster 1: Mean = [5.25836181 5.69938319], Size (amount of data points) = 511
Cluster 2: Mean = [2.71109944 5.67742348], Size (amount of data points) = 489
Cluster 3: Mean = [2.71131116 3.53001387], Size (amount of data points) = 516
Cluster 4: Mean = [5.22919222 3.46357939], Size (amount of data points) = 484
Final Cluster Assignments: [3 2 3 ... 1 1 0]
Number of Iterations: 3
Final Delta: 0.00033051171780545474
```

[Figure 11: twoEllipses.txt Elbow Graph] *[Figure 12: twoEllipses.txt Clustering Results]*

t.4.8k.txt Results:

I did 3 clusters for this dataset because when I plotted it using the elbow method. This one I had a hard time deciding where the most noticeable bend was. Each graph before had two bends, but at x=4 it was a way stronger bend. But for this one, I didn't know if 4 was the right answer, so I decided to switch it up and do 3. (*Figure 13*). There were two different means for each cluster because the dataset was split into two columns, Cluster 1 being 461.14 and 220.84, Cluster 2 being 174.64 and 160.48, and Cluster 3 being 436.82 and 91.89. The size for this dataset was different for each dataset. For Cluster 1 it was 2,024 data points, Cluster 2 was 3,822 data points, and Cluster 3 was 2,154 data points. Though I cannot see all of the final cluster assignments because there are so many data points, it was overall [101…111]. And the total number of iterations was 2, with a delta of 0.0 (*Figure 14*). The cluster sizes all being different show a little imbalance with the distribution of the data points between each cluster, this is reflected in the graph when Cluster 2 is noticeably bigger than Cluster 1 and 2 which are very similar in size. Only 2 iterations for the kMeans algorithm to converge indicates that this was a stabilized process, also proven by the low delta. This iteration again proves my theory that the iteration are always one less than the number of clusters requested.



[Figure 13: t4.8k.txt Elbow Graph]

```
labels, centroids = kmeans(dataTFour.values,3)
makeGraphs(dataTFour, labels, centroids)

Final mean and size for each cluster:
Cluster 1: Mean = [461.14237254 220.84095166], Size (amount of data points) = 2
024
Cluster 2: Mean = [174.6356706  160.47867343], Size (amount of data points) = 3
822
Cluster 3: Mean = [436.81780208  91.8876161 ], Size (amount of data points) = 2
154
Final Cluster Assignments: [1 0 1 ... 1 1 1]
Number of Iterations: 2
Final Delta: 0.0
```

[Figure 14: t4.8k.txt Clustering Results]

Conclusion

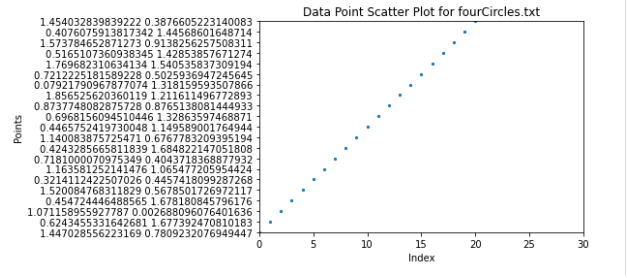
In this project, we applied our own kMeans clustering algorithm to 5 different datasets all filled with a range of floating-point numbers. This analysis allowed me to learn more about how clustering works and learn more about Python. Doing this project also helped me to understand these datasets better, and the different impacts each detail might have on the results. I am still unsure what is considered to be a dataset of irregular shape before clustering, but I did notice that with the datasets with a larger range of numbers, the clusters were definitely odd-looking. There were also some inconsistencies when using kMeans; whenever you reran the algorithm it have different results, which would change the way you analyze your data each time. This probably due to the random initialization of the centroids.

Reflections and Limitations

When trying to plot to find out how many clusters to request, I hadn't known I was supposed to use a specific method like the Elbow Method or Silhouette Method. So I originally just made a scatter plot, plotting all the points, which when I made it, made no sense to me because I didn't get how I was supposed to read it. I then realized there was a specific way I was supposed to plot it (*Figure 15*). When finally figuring out how to properly plot (using the Elbow Method) I was converting the datasets as floats there instead of converting it when importing it, I soon realized that and changed it.

```
x = dataFourCircle.index #gets the first element from dataset
y = dataFourCircle.iloc[:,0] #gets the second element from dataset

#plotting the data pts below
plt.scatter(x,y,s=5) #using matplotlib and x,y above to create a scatter plot, '
plt.title('Data Point Scatter Plot for fourCircles.txt') #titling graph
plt.xlabel('Index') #titling x-axis
plt.ylabel('Points') #titling y-axis
plt.xlim(0,30) #limits range of x-axis
plt.ylim(0,20) #limits range of y-axis
plt.show() #show graph
```



[Figure 15: Original Pre-Cluster Scatter Plot for fourCircles.txt]