FEBRUARY 14, 2020

COSI126A_HW1

CHUYUE WU

INSTRUCTOR: DR. HONGHU LIU

SCHOOL: BRANDEIS UNIVERSITY

Course Name: Introduction to Data Mining

Section I: Clustering Problems

Problem 1 (5 points)

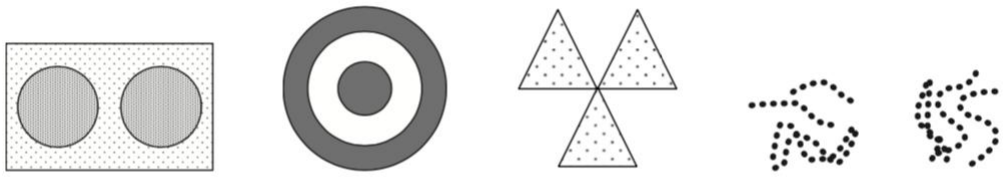
Many partitional clustering algorithms that automatically determine the number of clusters claim that this is an advantage. List two situations in which this is not the case.

(1) Sometimes we have the goal of clustering data into specific numbers of cluster. Then there is no need to let algorithms automatically determine the number of clusters. For example, when we have questionnaire data of customers, we want to divide them into three groups (high-income group, middle-income group, and low-income group) based on their salaries. If we use clustering algorithms that automatically determine the number of clusters to be 4, then we have trouble to reduce the clusters because that might cause inaccuracy.

(2) Sometimes when the characters between data point are not different obviously, then the number of clusters might be extremely large, which is useless in practical situation but only let the speed slow down.

Problem 2 (5 points)

Identify the clusters below using the center-, contiguity-, and density- based definitions. Also indicate the number of clusters for each case and give a brief indication of your reasoning. Note that darkness or the number of dots indicates density. If it helps, assume center-based means K-means, contiguity-based means single link, and density-based means DBSCAN.



(a) (b) (c) (d)

(a)
Center-based definition: There are 2 clusters. But the shape of cluster would not be those two circles, because both circle and noise point around it.
Contiguity-based definition: I guess there will only be 1 cluster, since the noise in between and around circles will connect the two distinct clusters into one.
Density-based definition: There are 2 clusters. There are two higher density clusters in the central and a lower density cluster at remaining surrounding. Since the density based definition takes into account the noise and eliminates those points from the cluster. So only two higher density clusters would be distinguished.

(b)
Center-based definition: There will be 1 cluster. Because the center of both dense regions is the same, the cluster will include the circle in the middle and the ring around it.
Contiguity-based definition: There will be 2 clusters. Because the two higher density regions can be separated by that lower density region which is totally white.
Density-based definition: There are 2 clusters. There are two higher density clusters in the central and a lower density cluster at remaining surrounding. Since the density based definition takes into account the noise and eliminates those points from the cluster. So only two higher density clusters would be distinguished.

(c)
Center-based definition: There are 3 clusters. Because there are three triangles and each of them could be defined as an individual cluster, so it could use k-means with 3 clusters.
Contiguity-based definition: There will be only 1 cluster because all points are connected in the center.
Density-based definition: There are 3 clusters. Each triangle is a cluster.

(d)
Center-based definition: There are 2 clusters, the left one and the right one.
Contiguity-based definition: There are 5 clusters. Since there are several single link groups of dots and several multiple links groups of dots, so it could use single link with 5 clusters.
Density-based definition: There are 2 clusters because the regions are separated by a space of low density in the middle, and higher density in each side.

Problem 3 (10 points)

Hierarchical clustering algorithms require O(m^2log(m)) time, and consequently, are impractical to use directly on larger data sets. One possible technique for reducing the time required is to sample the data set. For example, if K clusters are desired and $\sqrt{m}$ points are sampled from the m points, then a hierarchical clustering algorithm will produce a hierarchical clustering in roughly *O(m)* time. K clusters can be extracted from this hierarchical clustering by taking the clusters on the K^{th} level of the dendrogram. The remaining points can then be assigned to a cluster in linear time, by using various strategies. To give a specific example, the centroids of the *K* clusters can be computed, and then each of the m − $\sqrt{m}$ remaining points can be assigned to the cluster associated with the closest centroid.

For each of the following types of data or clusters, discuss briefly if (1) sampling will cause problems for this approach and (2) what those problems are. Assume that the sampling technique randomly chooses points from the total set of m points and that any unmentioned characteristics of the data or clusters are as optimal as possible. In other words, focus only on problems caused by the particular characteristic mentioned. Finally, assume that K is very much less than m.

(a) Data with very different sized clusters.
It would cause problems. By sampling, we may not choose certain representatives of the smaller clusters and therefore completely ignore them in the analysis. This will be biased towards the larger clusters.

(b) High-dimensional data.
It would cause problems. Because high-dimensional data is sparse and we need to use enough data to decide the cluster. Sampling could cause the worse cluster at the beginning and get bad results.

(c) Data with outliers, i.e., atypical points.
It would **not** cause problems. Sampling will not be a problem here, since by sampling our data, outliers are usually scarce, so there is a high chance that they are not selected for clustering and therefore they will not affect our results. For example, if we sample before performing k-means, by getting rid of outliers, our algorithm may converge faster or better identify correct clusters.

(d) Data with highly irregular regions.

It would cause problems. Because for highly irregular regions, some points which decide the structure of regions play more important roles than other points. So, sampling could cause problems.

(e) Data with globular clusters.
It would **not** cause problems. Because when sampling evenly from data with globular clusters, the data we got is still distributed in globular shape without losing their attributes.

(f) Data with widely different densities.
It would cause problems. Because there is some group with lower densities might be ignored at first when using $\sqrt{m}$ points do hierarchical clustering due to limited samples.

(g) Data with a small percentage of noise points.
It would not cause problems. Sampling will not be a problem since noise points will most likely not be sampled and therefore eliminated from clustering.

(h) Non-Euclidean data.
It would cause problems. Because the distribution of the non-euclidean data is different from Euclidean data, sampling random might cause uneven points selection.

(i) Euclidean data.
It would **not** cause problems.

(j) Data with many and mixed attribute types.
It would cause problems. Because data with many and mixed attribute types is just like high-dimensional data. The data is sparse and we need to use enough data to decide the cluster. Sampling could cause the worse cluster at the beginning and get bad results.

Problem 4 (8 points)

(1) Compute the entropy and purity for each cluster in the confusion matrix below.
Total1 = 8+22+0+0+767+4+45+22 = 868
Purity1 = maxP_{ij} = 767/868 = 0.884

Entropy1 = -(8/868)*log2(8/868)-(22/868)*log2(22/868)-(767/868)*log2(767/868)-(4/868)*log2(4/868)-(45/868)*log2(45/868)-(22/868)*log2(22/868) = 0.746

Total2 = 654+34+89+123+12+76+13+2 = 1003
Purity2 = maxP_{ij} = 654/1003 = 0.652
Entropy2 = -(654/1003)*log2(654/1003)-(34/1003)*log2(34/1003)-(89/1003)*log2(89/1003)-(123/1003)*log2(123/1003)-(12/1003)*log2(12/1003)-(76/1003)*log2(76/1003)-(13/1003)*log2(13/1003)-(2/1003)*log2(2/1003) = 1.707

Total3 = 6+301+2+3+98+23+31+1001 = 1465
Purity3 = maxP_{ij} = 1001/1465 = 0.683
Entropy3 = -(6/1465)*log2(6/1465)-(301/1465)*log2(301/1465)-(2/1465)*log2(2/1465)-(3/1465)*log2(3/1465)-(98/1465)*log2(98/1465)-(23/1465)*log2(23/1465)-(31/1465)*log2(31/1465)-(1001/1465)*log2(1001/1465) = 1.381

Total4 = 4+21+34+2+3+543+112+0 = 719
Purity4 = maxP_{ij} = 543/719 = 0.755
Entropy4 = -(4/719)*log2(4/719)-(21/719)*log2(21/719)-(34/719)*log2(34/719)-(2/719)*log2(2/719)-(3/719)*log2(3/719)-(543/719)*log2(543/719)-(112/719)*log2(112/719) = 1.179

(2) Compute the total entropy and total purity.

Total = Total1 + Total2 + Total3 + Total4 = 868+1003+1465+719 = 4055
Total_purity = (868/4053)*0.884 + (1003/4053)*0.652 + (1465/4053)*0.683 + (719/4053)*0.755 = 0.731
Total_entropy = (868/4053)* 0.746+ (1003/4053)*1.707 + (1465/4053)* 1.381+ (719/4053)* 1.179 = 1.291

(3) Compute the following F-measure: F (#3;Water)

R(#3;Water) = 301/1465 = 0.205
P(#3;Water) = 301/(22+34+301+21) = 0.796
F(#3;Water)
= 2 * R(#3;Water) * P(#3;Water) / (R(#3;Water) + P(#3;Water))
= 0.326

Cluster	Normal	Water	Grass	Fire	Electric	Ground	Flying	Ghost
#1	8	22	0	0	767	4	45	22
#2	654	34	89	123	12	76	13	2
#3	6	301	2	3	98	23	31	1001
#4	4	21	34	2	3	543	112	0

Problem 5 (7 points)

(a) Given the set of cluster labels and similarity matrix shown below, compute the correlation between the similarity matrix and the ideal similarity matrix, i.e., the matrix whose ij^{th} entry is 1 if two objects belong to the same cluster, and 0 otherwise.

	P1	P2	P3	P4	P5
P1	1	0.92	0.33	0.61	0.82
P2	0.92	1	0.43	0.01	0.22
P3	0.33	0.43	1	0.75	0.11
P4	0.61	0.01	0.75	1	0.17
P5	0.82	0.22	0.11	0.17	1

(Similarity Matrix)

	P1	P2	P3	P4	P5
P1	1	1	0	0	0
P2	1	1	0	0	0
P3	0	0	1	1	0
P4	0	0	1	1	0
P5	0	0	0	0	1

(Ideal Matrix)

correlation = corr(c(Similarity Matrix), c(Ideal Matrix)) = 0.800
(using R to get result)

(b) Compute the silhouette coefficient for each point, each of the three clusters, and the overall clustering.

Firstly, we need to construct the distance matrix from similarity matrix by simply subtracting 1 from every element in similarity matrix.

	P1	P2	P3	P4	P5
P1	0	0.08	0.67	0.39	0.18
P2	0.08	0	0.57	0.99	0.78
P3	0.67	0.57	0	0.25	0.89
P4	0.39	0.99	0.25	0	0.83
P5	0.18	0.78	0.89	0.83	0

(Distance Matrix)

For P1, s1 = (0.18 - 0.08)/0.18 = 0.56
For P2, s2 = (0.78 - 0.08)/0.78 = 0.90
For P3, s3 = ((0.67+0.57)/2-0.25)/((0.67+0.57)/2) = 0.60
For P4, s4 = ((0.39+0.99)/2-0.25)/((0.39+0.99)/2) = 0.64
For P5, s5 = 0

So For C1, sc1 = (0.56+0.90)/2 = 0.73
For C2, sc2 = (0.60+0.64)/2 = 0.62
For C3, sc3 = 0

And the total silhouette coefficient is an average for every cluster:
S = (0.73 + 0.62 + 0)/3 = 0.45

Section II: Programming

Please submit a Python package implementing the features specified below. A skeleton file will be provided.

Part I: Cluster Validity (15 points)

1. Implement the following external measurements:
    (a) Accuracy
    (b) Normalized Mutual Information
    (c) Normalized Rand Index
2. Implement the following internal measurements:
    (a) Silhouette Index
    (b) Clustering Validation Index based on Nearest Neighbors (CVNN)

In this part, I write my own function and put them in HW1_Programming_1.py.
For external measurements, I use label of cluster result and true class as variable to measure the cluster results.
For internal measurements, I only use label of cluster result and dataset to measure the results. Internal measurements are only way to measure unsupervised learning and its efficiency are not as good as external measurements when we have the true class of data and want to check the cluster results.

References:

Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J., & Wu, S. (2013). “Understanding and enhancement of internal clustering validation measures.” IEEE transactions on cyber- netics, 43(3), 982-994.

Wu, J., Xiong, H., & Chen, J. (2009, June). “Adapting the right measures for k- means clustering.” Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 877-886). ACM.

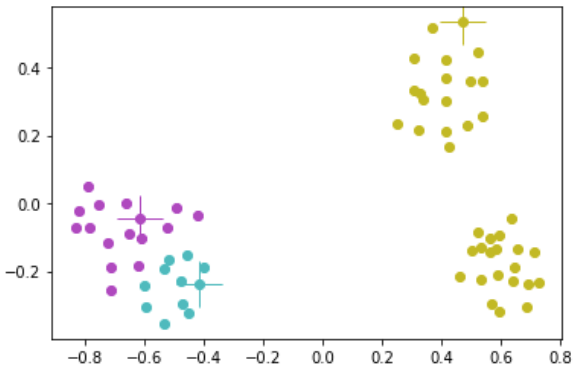
Part II: K-Means Clustering (20 points)

1. Implement Lloyd’s K-means algorithm. Ensure that it works for the basic dataset three globs.csv. Provide the following three initialization methods:
    (a) Random
    (b) K-means++

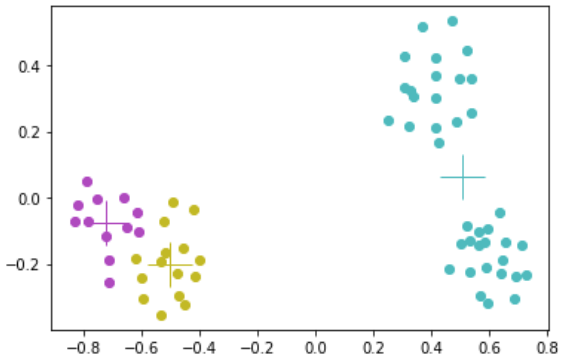
I write my own K-means algorithm in HW1_Programming_2_1_Lloyd.py and use three_globs.csv to test the data.

For random initialization, the results are below:

```
initial centroids:  [array([-0.61347 , -0.044888]),
array([0.47382, 0.53367]), array([-0.41397, -0.2394 ])]
```




```
7 times centroids:  [array([-0.72236167, -0.07605971]),
array([-0.50108133, -0.1998334 ]), array([0.50991053,
0.06285707])]
```

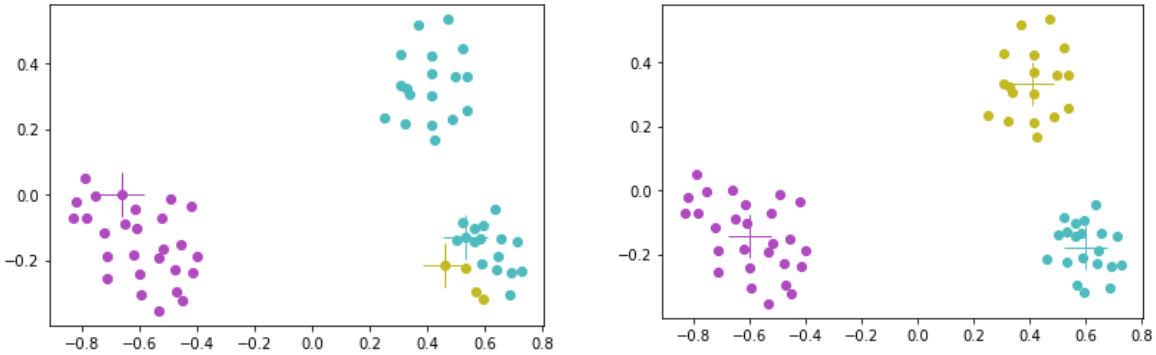


It uses 7 times to get convergence. The results are not very ideal. Actually, based on the scatter plot, the ideal clustering should be group of left bottom, group of right bottom and group of right upper. But because two of random initial points are at the left bottom, the final results are not difficult to understand.

Since the results are not very good, I run the function again. Initialization is a total random event. In this time, because the locations of three initial points are better, ideal three groups are successfully separated. It uses 5 times to get convergence. The results are shown below:

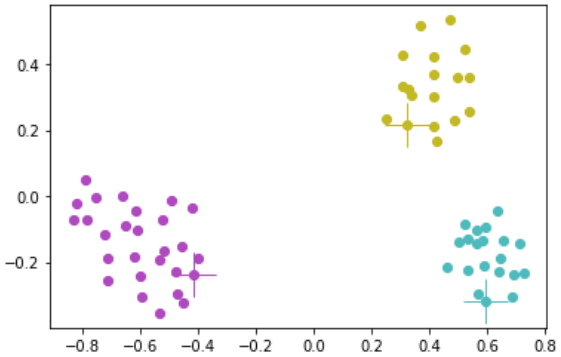
```
initial centroids:  [array([-6.6334e-01, -2.2204e-16]),
array([ 0.45885, -0.21446]), array([ 0.53367, -0.12968])]
```

```
5 times centroids:  [array([-0.59942815, -0.14482287]),
array([0.40953333, 0.33361   ]), array([ 0.60025    ,
-0.18082058])]
```

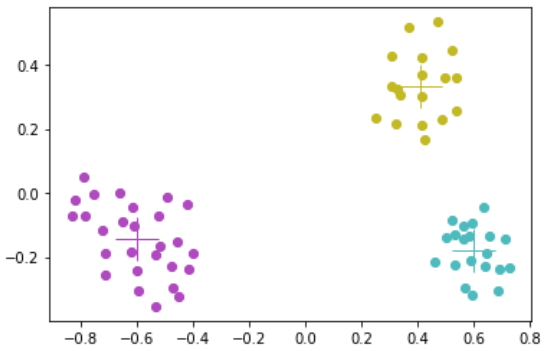


Then I use my own k-means++ function to initialize the centroid points, the results are here:

```
initial centroids:  [array([-0.41397, -0.2394 ]),
array([0.32419, 0.21446]), array([ 0.59352, -0.3192 ])]
```




```
3 times centroids:  [array([-0.59942815, -0.14482287]),
array([0.40953333, 0.33361   ]), array([ 0.60025   ,
-0.18082058])]
```



It uses three times to get convergence and the initialized points as well as result does look better. However, when I run it again and again, the bad clustering in the first situation of random cluster does appear. So we can say k-means++ could help increases the probability of getting the better results, but it could not guarantee it.

3. Use the internal measurements in Part 1 to find the proper cluster number for the image segmentation.csv dataset.
In HW1_Programming_2_3.py, I use my Silhouette_Index() function in HW1_Programming_1.py to test k from 2 to 10, the results are shown below:

```
for k =  2
Silhouette Index:  0.23717514677301008
for k =  3
Silhouette Index:  0.36118321250675195
for k =  4
Silhouette Index:  0.30753714743618393
for k =  5
Silhouette Index:  0.2924532035114545
for k =  6
Traceback (most recent call last):

  File "<ipython-input-339-a668509d6d51>", line 19, in
<module>
    Silhouette_Index(labels, data)

  File "/Users/cyfile/Documents/Brandeis/Data Mining/HW1/
HW1_Programming_1.py", line 101, in Silhouette_Index
    a_list.append(a/counta)

ZeroDivisionError: division by zero
```

However, although we can see when k is equal to 3 there is highest silhouette index, but why is there an error when k = 6? So I check the cluster result in Variable explorer block:

cluster - Dictionary (6 elements)

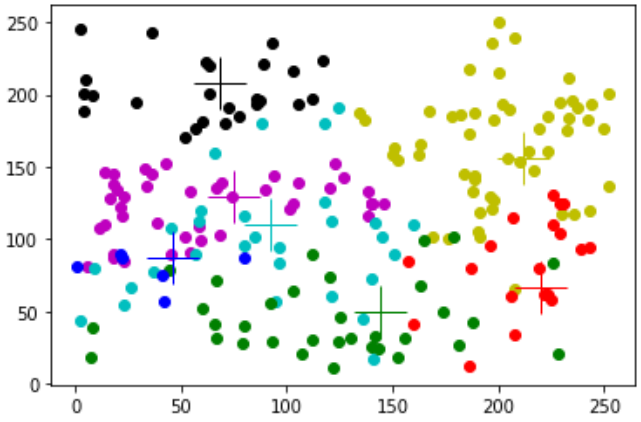
e ▲	Type	Size	Value
0	list	86	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
1	list	57	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
2	list	18	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
3	list	17	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...
4	list	1	[Numpy array]
5	list	31	[Numpy array, Numpy array, Numpy array, Numpy array, Numpy array, Nump ...

Save and Close Close

We found that when k = 6, there is a cluster only have one initial point due to our limited samples. So when I calculate silhouette index, denominator count a is zero, then it appears an error and stops the loop. Anyway, in this case, we use k = 3 as the best cluster number.

4. Given the true cluster number, run your Lloyd's K-means algorithm on the image segmentation.csv dataset, and evaluate the results in terms of the external measurements completed in Part I.

According to the true cluster, the number of cluster is 7. So I run my Lloyd's K-means algorithm in HW1_Programming_2_4.py. The 2D visualization is shown below, but it's just for reference since our data is high-dimensioned data.



The external measurements results are shown below：

```
In [393]: Accuracy(cluster, true_class)
     ...: NMI(cluster, true_class)
     ...: NRI(cluster, true_class)
Accuracy:  0.7450980392156863
Normalized Mutual Information:  0.5400411226680838
Normalized Rand Index 0.4113144273529197
```

5. Run the algorithm ten times, and record the average, standard deviation, and execution time.

Execution time for running ten times is 17s. Average accuracy is 0.69, average NMI is 0.53, and average NRI is 0.44.

References:

Arthur, D., & Vassilvitskii, S. (2007, January). “k-means++: The advantages of careful seeding.” Proceedings of the eighteenth annual ACM-SIAM Symposium on Discrete Algorithms (pp. 1027-1035). Society for Industrial and Applied Mathematics.

Likas, A., Vlassis, N., & Verbeek, J. J. (2003). “The global k-means clustering algo- rithm.” Pattern Recognition, 36(2), 451-461.

Part III: DBSCAN(15 points)

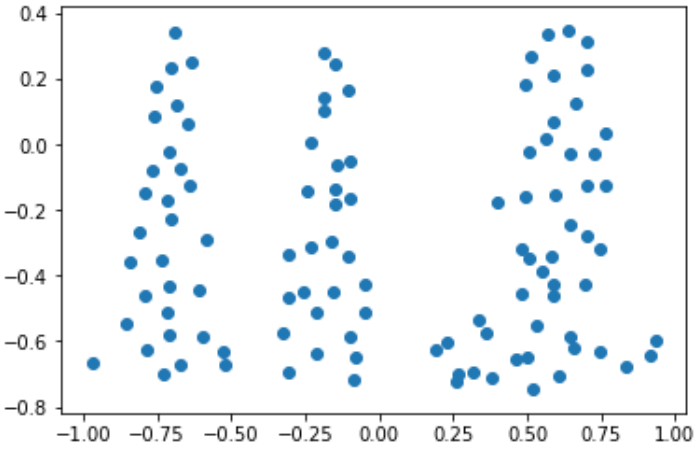
1. Implement your own DBSCAN. 5-point bonus for an implementation within 40 lines.

2. Run your algorithm on the anthill.csv dataset. You can use the recommended parameters in the textbook, or you can choose your own. Evaluate the performance in terms of the external measurements in Part 1.

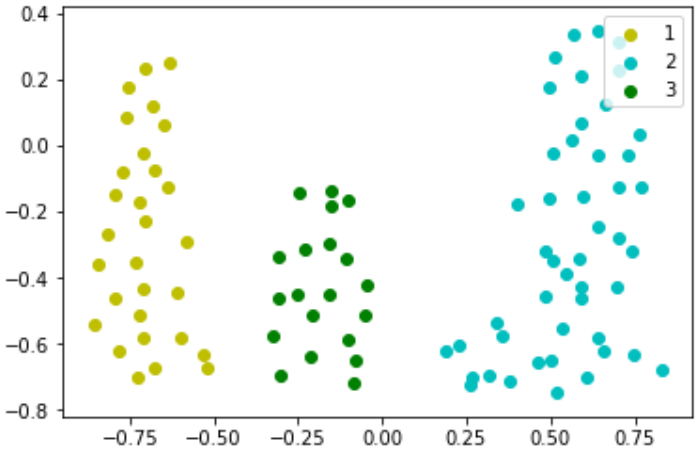
Process:

I use my own function my_DBSCAN (HW1_Programming_3.py) to cluster the anthill.csv dataset.

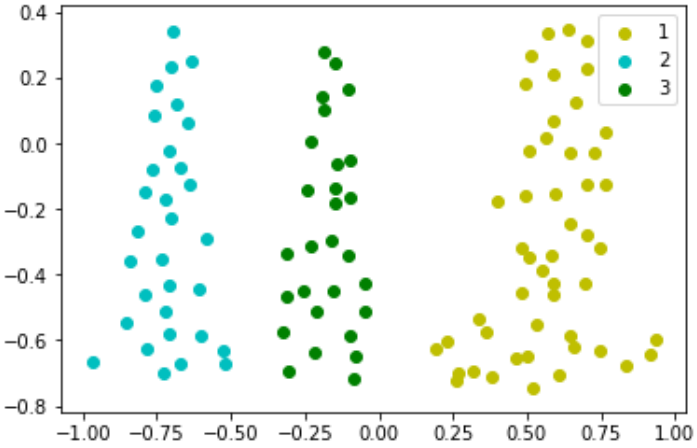
At first, I had a quick look at dataset, it might be clustered into three parts：



Since there are 108 points and the distance between each point is not big, so I set eps to be 0.2 and minPts to be 10. The visualization is shown below:



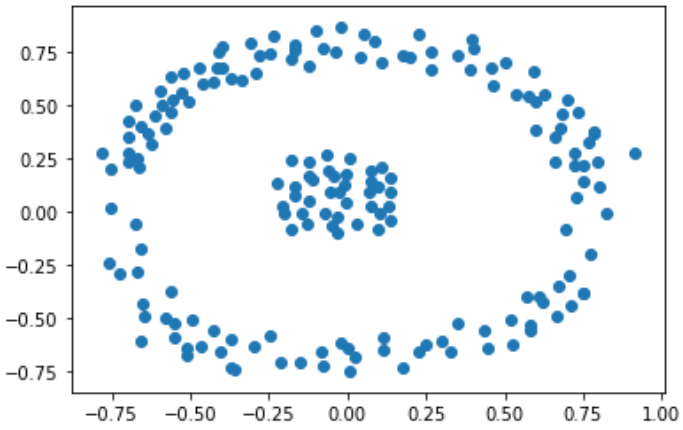
Actually, it's not bad, but it seems that there are too many points be deleted when minPts is set to be 10. MinPts needs to be smaller. Thus, I changed it to be 5, the results are shown below. It's great!



Part IV: Spectral Clustering and Kernel K-means (15 points)

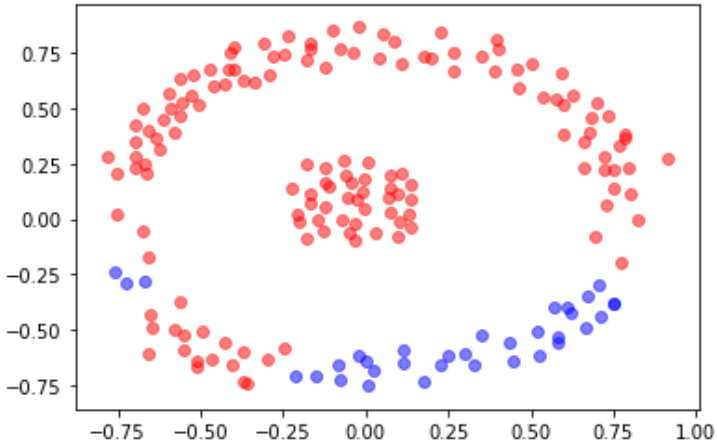
1. Implement your own spectral clustering algorithm.
2. Implement your own Kernel K-means algorithm.
3. Run both algorithms on the eye.csv dataset. Demonstrate empirically whether they are the same or not.

For this question, I put my code of spectral clustering algorithm in HW1_Programming_4_1.py, and Kernel K-means in HW1_Programming_4_2.py. Before we run the algorithms, we can have a look at the original dataset:

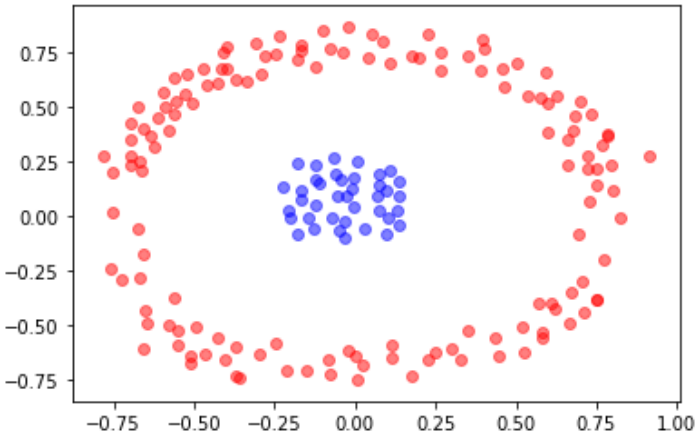


It is obvious that there should be two clusters, one is outside circle, and the other is the inside circle. I set the k to 2 for each algorithm.

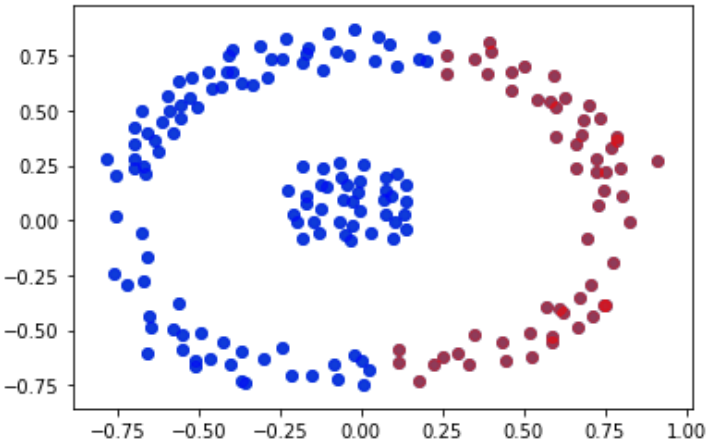
For spectral cluster, at the beginning, I set cutoff to 0.99, the results are shown here:



It didn't split well. That's because the cut off is a little big. Then, I change cut off to 0.98, the results are quite idea.



Then I ran kernel k-means with k equals to 2. The results are shown here. At the beginning, I thought kernel k-means would be as good as spectral clustering. However, from this implementation, I found it is not very practical to cluster this kind of data.



References:

• Dhillon, I. S., Guan, Y., & Kulis, B. (2004, August). "Kernel k-means: spectral clustering and normalized cuts." Proceedings of the tenth ACM SIGKDD international conference on knowledge discovery and data mining (pp. 551-556). ACM.