0880803 – 馮清海

Thanh – Hai, Phung

**HOMEWORK 6**

**MACHINE LEARNING**

**K Mean Clustering, Spectral Clustering**

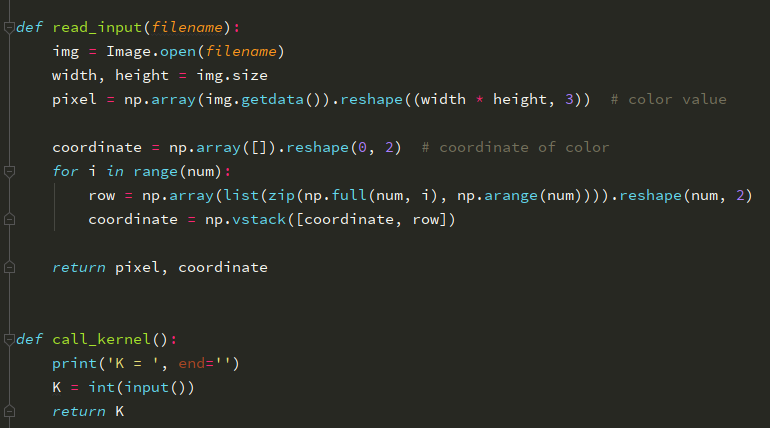
**Input image**



1. **Kernel K mean**

**Reading input**

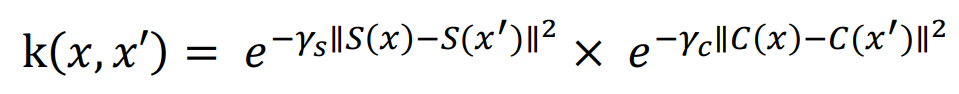
<code>



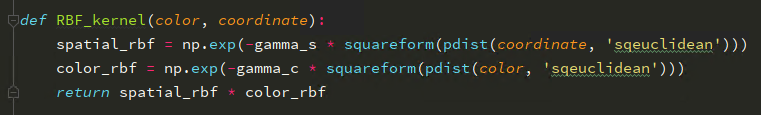
Firstly, we read the input picture using pillow library, separating the coordinate and pixel of the RGB picture, calling the kernel (e.g. 2, 3, 4)

**Method**

In this algorithm, we implement two RBF kernel to be one to calculate the Gram matrix by the following formula

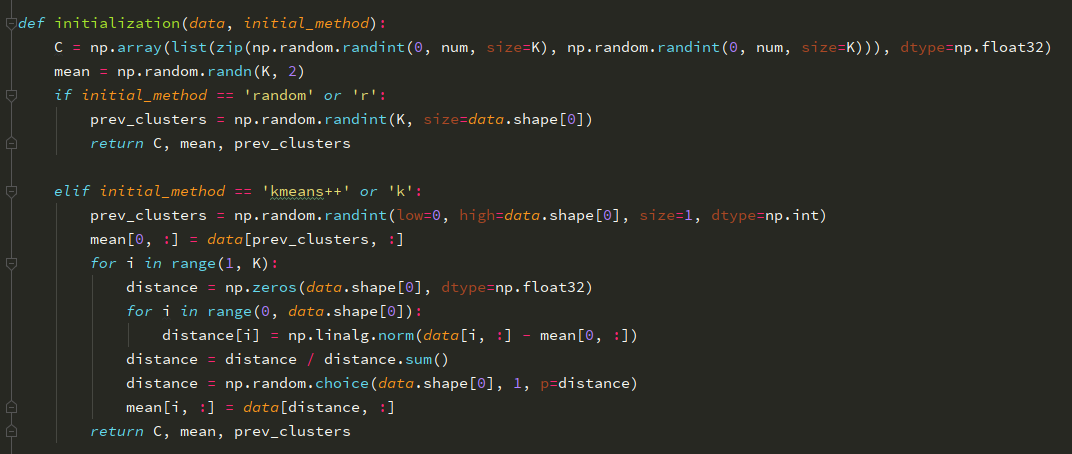


<code>



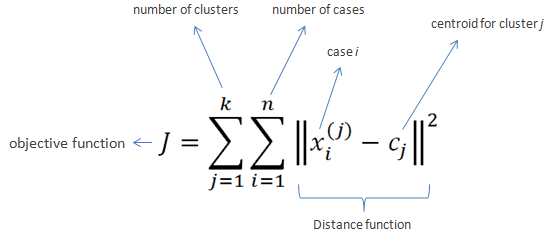
Then following we define the initialization K mean cluster, in this project, we did 2 types of methods which are **RANDOM** and **KMEAN++**.

<code>



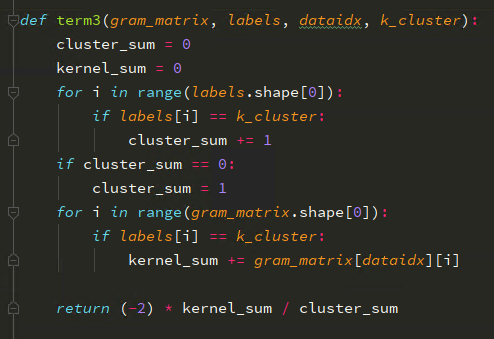
In the fundamental of Kernel K – Mean Clustering, it said that the K mean ++ has converges faster than other methods by defining the initial cluster instead of **None** as other methods. Let’s see if it is True or Not in the following result.

Following we define the two terms of calculating the cluster centers by two step:



**Step 1**: We cluster the data in *K* group where *K* is predefined

<code>



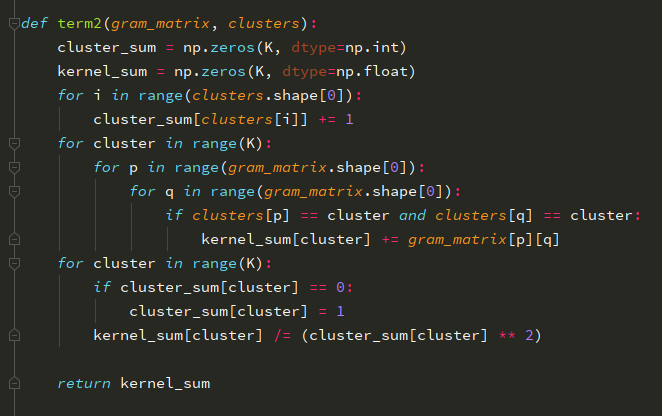
Then we count the number of elements of each cluster by ***labels[i] == k\_cluster***

**Step 2**: Applying the initialization method by choosing random data point in same group. It can be “random” or “kmeans++” as above defined.

**Step 3**: Assign the data point with their nearest cluster center according to Euclidean distance function.

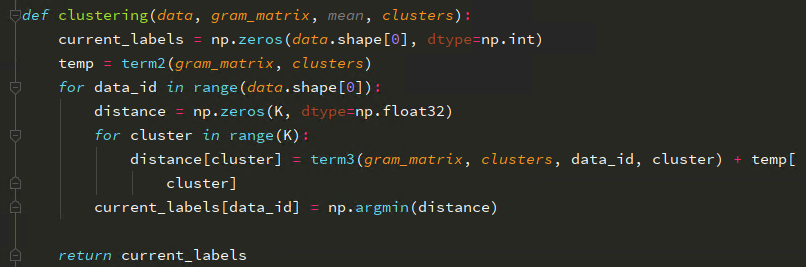
In this step, we calculate the second term of applied kernel distance

<code>



**Step 4**: We calculate the centroid (or mean) of the same data point which was assigned in the previous step)

<code>



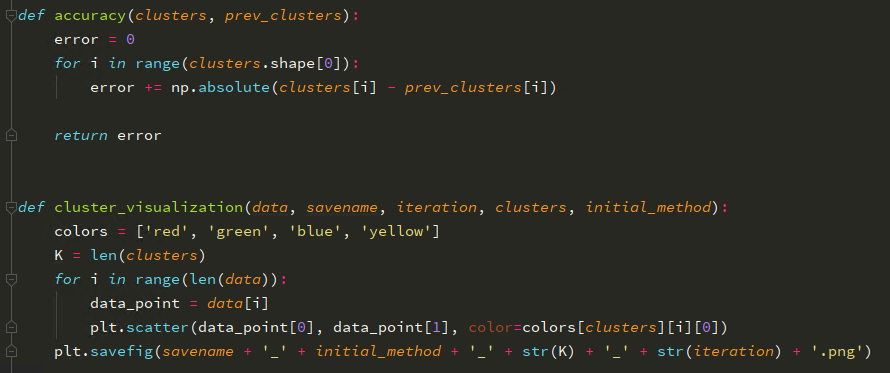
Assign the data point to the nearest mean.

**Step 5**: We repeat the step 2 and 3 until it converges.

<code> 

Following that we calculate the accuracy of the step assign labels and cluster of the datapoint for observing how and how long it take to converge.

<code>



**Table 1: Result of K – mean Clustering with different method, different kernels.**

\*\*Error: Image 1, Image 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mode** | **Random** | | | | | | **K mean ++** | | | | | |
| **Kernel**  **Iteration** | **2** | | **3** | | **4** | | **2** | | **3** | | **4** | |
| 1 | 4968 | 4902 | 9437 | 9339 | 13951 | 12781 | 4928 | 4922 | 8536 | 8778 | 11687 | 12292 |
| 2 | 446 | 974 | 930 | 1598 | 1741 | 1794 | 912 | 628 | 1689 | 1683 | 1964 | 3061 |
| 3 | 120 | 218 | 95 | 734 | 879 | 902 | 176 | 218 | 531 | 842 | 340 | 1225 |
| 4 | 63 | 86 | 19 | 331 | 545 | 618 | 28 | 89 | 222 | 582 | 94 | 687 |
| 5 | 20 | 54 | 7 | 194 | 353 | 402 | 4 | 33 | 174 | 368 | 26 | 443 |
| 6 | 7 | 46 | 4 | 140 | 284 | 297 | 0 | 9 | 100 | 208 | 9 | 314 |
| 7 | 2 | 29 | 2 | 109 | 378 | 211 | 0 | 6 | 60 | 101 | 9 | 210 |
| 8 | 1 | 20 | 1 | 82 | 906 | 135 |  | 3 | 41 | 69 | 1 | 205 |
| 9 | 0 | 14 | 0 | 36 | 1272 | 81 |  | 3 | 35 | 43 | 0 | 205 |
| 10 | 0 | 6 |  | 21 | 539 | 46 |  |  | 17 | 35 | 0 |  |
| 11 |  | 2 |  | 13 | 218 | 28 |  |  | 10 | 20 |  |  |
| 12 |  | 0 |  | 15 | 114 | 21 |  |  | 11 | 18 |  |  |
| 13 |  |  |  | 12 | 59 | 11 |  |  | 8 | 8 |  |  |
| 14 |  |  |  | 5 | 34 | 8 |  |  | 6 | 8 |  |  |
| 15 |  |  |  | 3 | 17 | 4 |  |  | 0 |  |  |  |
| 16 |  |  |  | 2 | 16 | 3 |  |  |  |  |  |  |
| 17 |  |  |  | 2 | 7 | 0 |  |  |  |  |  |  |
| 18 |  |  |  |  | 4 | 0 |  |  |  |  |  |  |
| 19 |  |  |  |  | 2 |  |  |  |  |  |  |  |
| 20 |  |  |  |  | 0 |  |  |  |  |  |  |  |

**COMMENT:**

* Based on the table above, we may see the changing of distance between the clusters of different data point, the **Random** is said that it converges slower than **Kmean++** method, but in this implementation I didn’t see it has much sense. In some case, the Kmean++ have slower converges than Random. May be it because of the input structure don’t have a good spherical-like shape. We can easily see it in the table I, where the Random method have better performance and time consuming in Image 1 compare to Image 2, which have more complicate spherical shape.
* One more comment is that it looks likes the Kernel K mean couldn’t figure out the clusters correctly, we may have mentioned it in the gif file showing time series of an individual method, from kernel 2 to kernel 4 with different method, different type of images. In other words, data points in smaller clusters may be left away from the centroid itself in order to focus more to the larger clusters.
* Last but not the least, Kmean assumes spherical shapes of clusters which have radius equal to the distance between the centroid and the furthest data point and it didn’t work well when clusters are in different shape such as elliptical clusters.

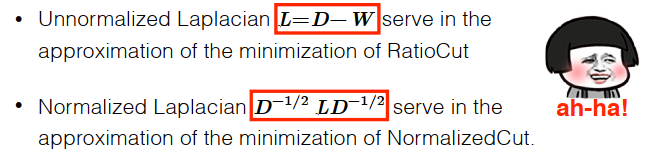
1. **Spectral Clustering**

In this implementation, the Spectral Clustering is implement same as the K mean method. There are few essential points here make them different:

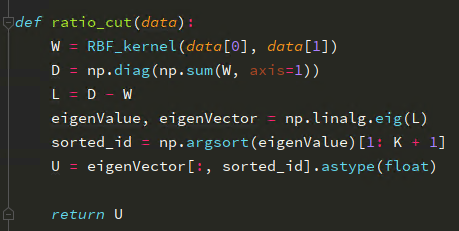
* The K mean clustering divide the objects into k clusters such that some metric relative to the centroids of the clusters is minimized.
* The Spectral clustering make the data points as nodes of a connected graph and clusters are found by partitioning this graph, based on its spectral decomposition into the subgraphs.

To define the Spectral Clustering method, we need to use the Ratio Cut (graph cut) and Normalized Cut to sort the eigenvalue and eigenvector of L (define by the weight and degree)

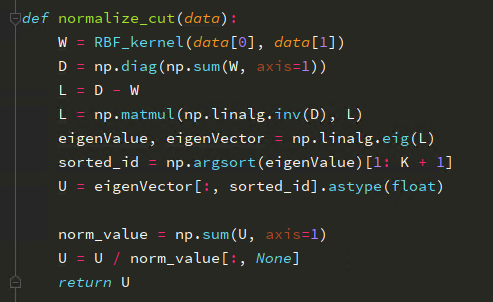
We can see the different between the Ratio and Normalize here by defining the weight eigenvalue of every data point later.

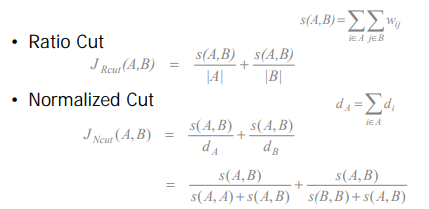


<code>



<code>

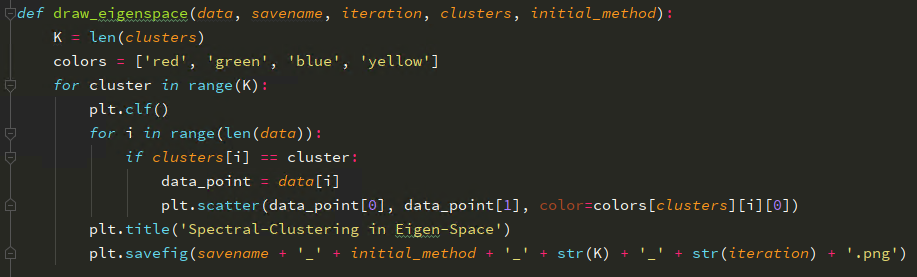




We need to find the Min similar between A and B for both Normalize and Ratio Cut, then we need to calculate the L by the Diagonal Matrix D and L.

And use the matrix U sorted by the eigenvector and eigenid itself to calculate the clustering of the data point input.

Visualize the eigen space map.



Calculate the clustering by repeat the step 2 and 3 by Normalize Cut and Ratio Cut

<code>



**Table 2: Result of Spectra Clustering with different method, different kernels.**

\*\*Error: Image 1, Image 2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Mode** | | **Random** | | | | | | **K mean ++** | | | | | |
| **Kernel**  **Iteration** | | **2** | | **3** | | **4** | | **2** | | **3** | | **4** | |
| 1 | Normalize Cut | 5019 | 4983 | 10096 | 9810 | 14811 | 9998 | 4960 | 5041 | 8852 | 9302 | 12943 | 12315 |
| Ratio Cut | 4987 | 5026 | 6697 | 9819 | 15093 | 10003 | 4977 | 5045 | 8123 | 8678 | 11883 | 13533 |
| 2 | Normalize Cut | 5026 | 4126 | 9958 | 4260 | 14832 | 11056 | 258 | 107 | 738 | 1719 | 4169 | 1851 |
| Ratio Cut | 6396 | 5098 | 6396 | 10196 | 19188 | 10186 | 264 | 161 | 867 | 823 | 3075 | 1154 |
| 3 | Normalize Cut | 551 | 217 | 1207 | 3866 | 2527 | 2650 | 172 | 57 | 520 | 1242 | 1116 | 784 |
| Ratio Cut | 388 | 65 | 2822 | 3373 | 3078 | 3362 | 109 | 122 | 1150 | 631 | 1601 | 958 |
| 4 | Normalize Cut | 198 | 71 | 1771 | 3273 | 2228 | 3368 | 108 | 34 | 452 | 618 | 187 | 520 |
| Ratio Cut | 158 | 54 | 597 | 1014 | 2250 | 4995 | 49 | 114 | 1179 | 556 | 986 | 763 |
| 5 | Normalize Cut | 46 | 42 | 397 | 1015 | 1603 | 2989 | 44 | 30 | 254 | 256 | 127 | 449 |
| Ratio Cut | 71 | 44 | 821 | 358 | 2311 | 2654 | 27 | 90 | 964 | 531 | 694 | 754 |
| 6 | Normalize Cut | 10 | 48 | 12 | 294 | 584 | 971 | 17 | 20 | 137 | 148 | 111 | 461 |
| Ratio Cut | 35 | 34 | 678 | 180 | 1372 | 1578 | 14 | 60 | 764 | 520 | 645 | 709 |
| 7 | Normalize Cut | 2 | 48 | 4 | 127 | 615 | 1191 | 9 | 18 | 63 | 63 | 101 | 414 |
| Ratio Cut | 17 | 33 | 619 | 153 | 773 | 622 | 9 | 32 | 652 | 506 | 616 | 653 |
| 8 | Normalize Cut | 0 | None | 0 | 65 | 519 | 1029 | 6 | 19 | 34 | 33 | 83 | 349 |
| Ratio Cut | 10 | 20 | 447 | 127 | 359 | 334 | 3 | 25 | 518 | 401 | 554 | 554 |
| 9 | Normalize Cut | 0 | None | 0 | 41 | 388 | 846 | 3 | 8 | 15 | 21 | 46 | 247 |
| Ratio Cut | 4 | 17 | 328 | 113 | 367 | 226 | 1 | 24 | 452 | 337 | 524 | 486 |
| 10 | Normalize Cut | None | None | None | 26 | 319 | 805 | 3 | 4 | 6 | 18 | 36 | 184 |
| Ratio Cut | 1 | 11 | 195 | 66 | 335 | 172 | 0 | 13 | 349 | 304 | 500 | 440 |
| 11 | Normalize Cut | None | None | None | 10 | 252 | 864 | None | 1 | 6 | 5 | 17 | 107 |
| Ratio Cut | 0 | 9 | 147 | 42 | 343 | 169 | 0 | 13 | 248 | 234 | 478 | 411 |
| 12 | Normalize Cut | None | None | None | 10 | 189 | 960 |  | 1 | None | 5 | 17 | 77 |
| Ratio Cut | 0 | 7 | 93 | 34 | 348 | 163 |  | None | 155 | 210 | 436 | 340 |
| 13 | Normalize Cut |  | None | None | None | 138 | 988 |  |  | None | None | None | 66 |
| Ratio Cut |  | 4 | 53 | 23 | 343 | 156 |  |  | 98 | 153 | 387 | 294 |
| 14 | Normalize Cut |  | None | None | None | 88 | 631 |  |  | None | None | None | 22 |
| Ratio Cut |  | 1 | 40 | 9 | 348 | 110 |  |  | 64 | 92 | 344 | 249 |
| 15 | Normalize Cut |  | None | None | None | 65 | 368 |  |  | None | None | None | 9 |
| Ratio Cut |  | 0 | 39 | 4 | 368 | 87 |  |  | 41 | 53 | 312 | 205 |
| 16 | Normalize Cut |  |  | None | None | 31 | 178 |  |  | None | None | None | 9 |
| Ratio Cut |  |  | 27 | 3 | 386 | 67 |  |  | 29 | 41 | 319 | 208 |
| 17 | Normalize Cut |  |  | None | None | 27 | 50 |  |  | None | None | None | None |
| Ratio Cut |  |  | 16 | 4 | 370 | 41 |  |  | 24 | 30 | 325 | 175 |
| 18 | Normalize Cut |  |  | None | None | 14 | 26 |  |  | None | None | None | None |
| Ratio Cut |  |  | 18 | 2 | 348 | 43 |  |  | 22 | 14 | 317 | 131 |
| 19 | Normalize Cut |  |  | None | None | 7 | 5 |  |  | None | None | None | None |
| Ratio Cut |  |  | 6 | 2 | 313 | 51 |  |  | 11 | 6 | 309 | 109 |
| 20 | Normalize Cut |  |  | None |  | 2 | 0 |  |  | None | None |  | None |
| Ratio Cut |  |  | 3 |  | 290 | 37 |  |  | 13 | 3 |  | 88 |
| 21 | Normalize Cut |  |  | None |  | 0 | 0 |  |  | None | None |  | None |
| Ratio Cut |  |  | 1 |  | 263 | 36 |  |  | 4 | 6 |  | 68 |

**COMMENT:**

* There are two methods for calculate the L and Lsym as kernel matrix. The success of spectral clustering compare to K mean is mainly based on the fact that it does not make strong assumptions on the form of the clusters. The method of convex sets to be precise.
* It is not trivial to have good clusters of data point, but it can be separate the two data point of kernel instead of mixing it as Kmean. We can easily to see that the Spectral separate so well the neighbors of different clusters.

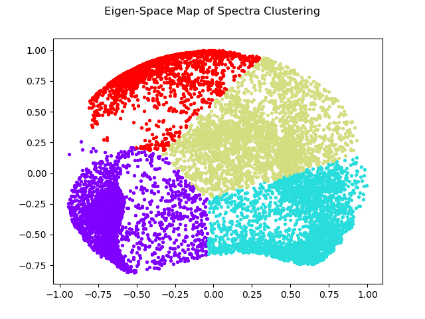
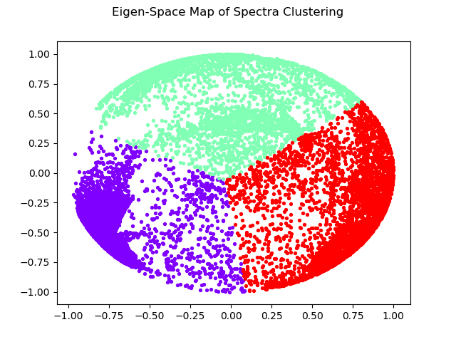
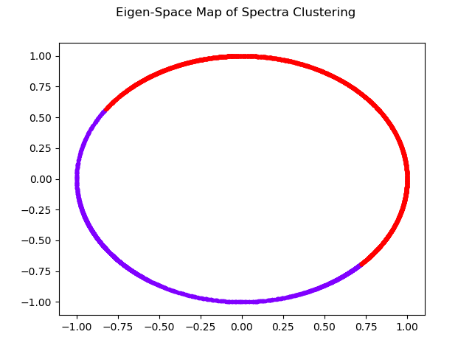
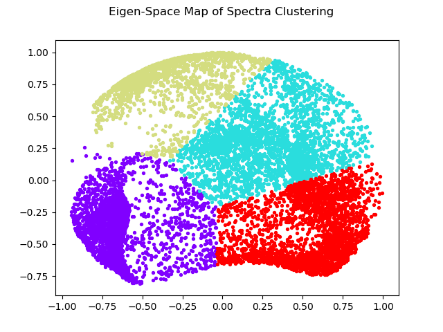
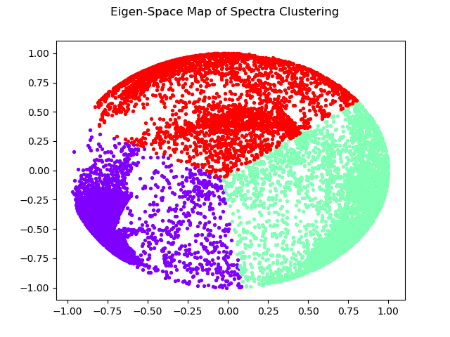
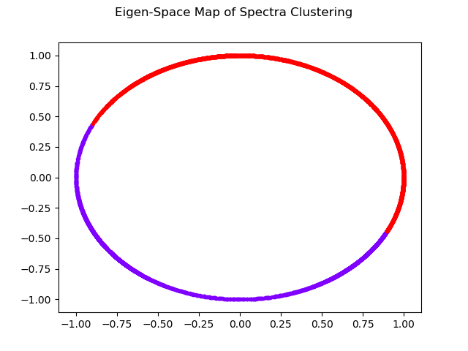
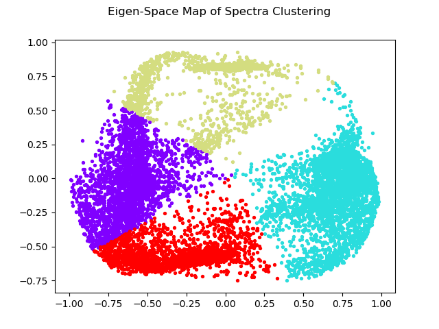
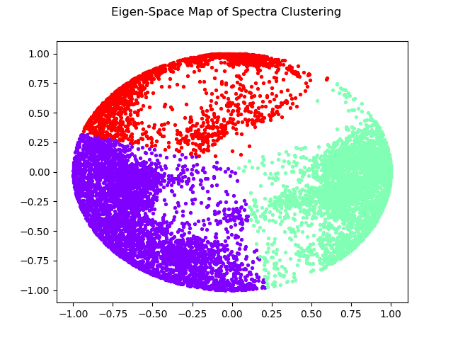
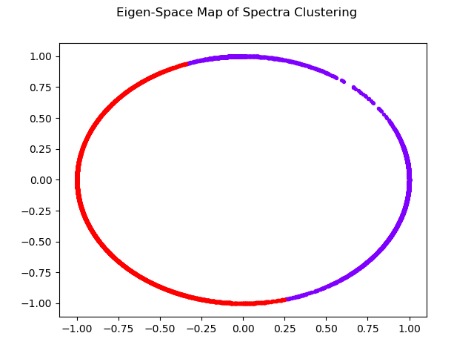
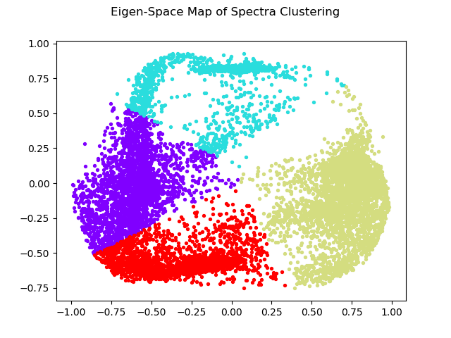
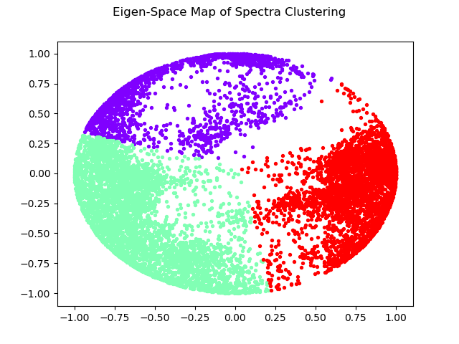
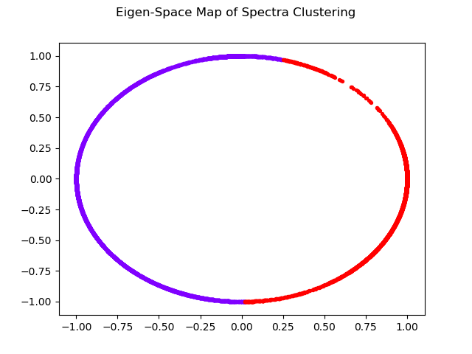


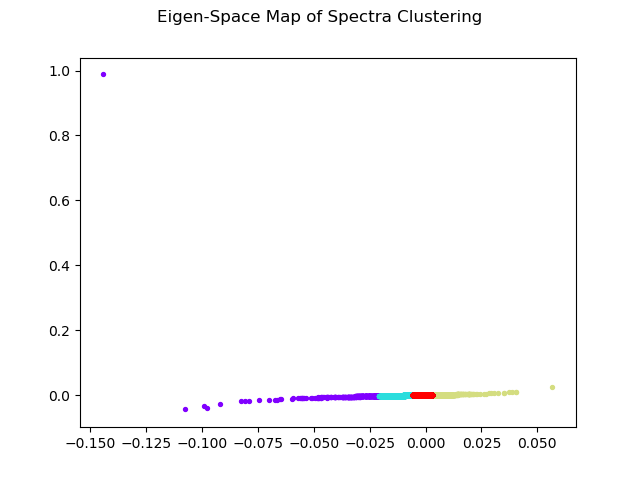
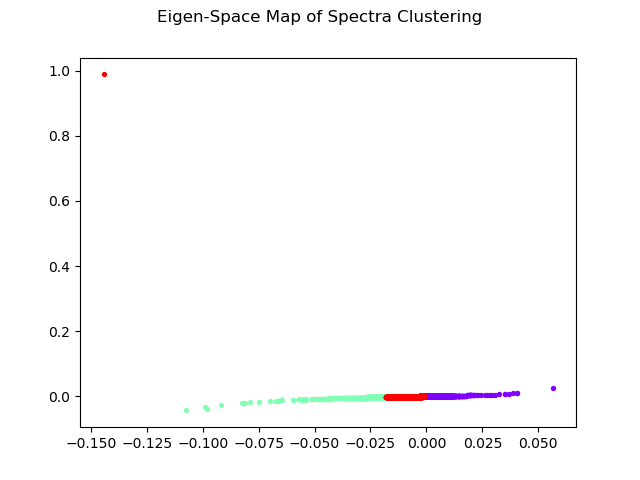
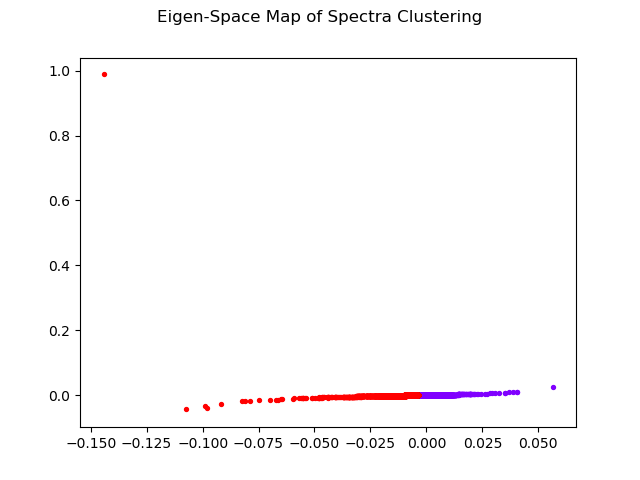
Figure 1: Eigen-space of Cluster point by Normalize Cut. From left to right, from top to bottom.

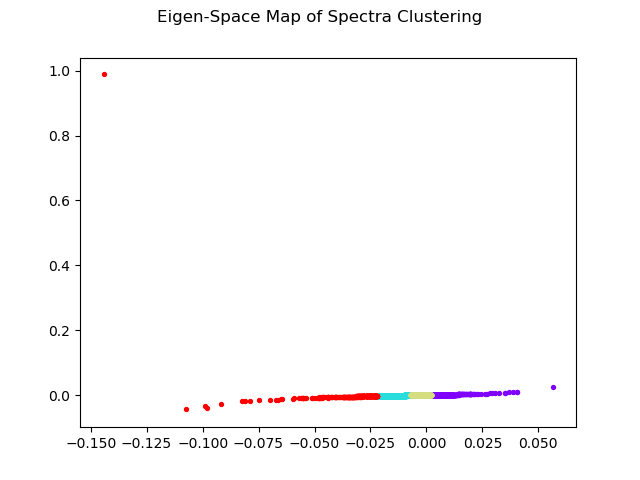
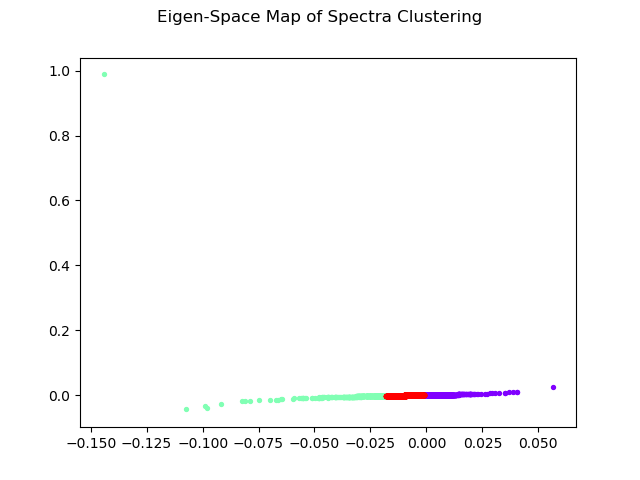
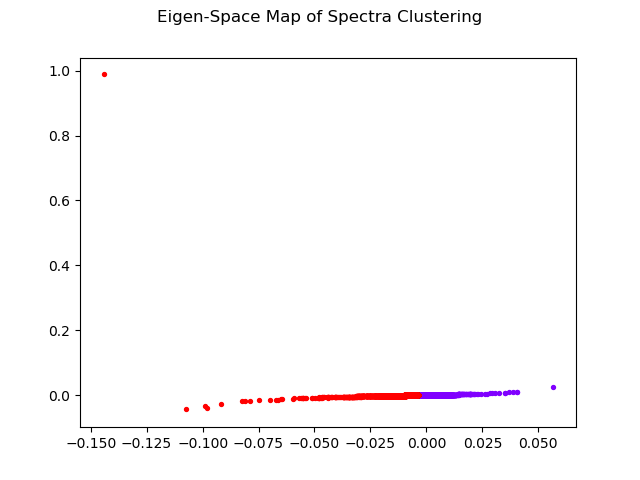
Image 1 Normalize Cut with 2 – 3 – 4 cluster point in Kmean++ method,

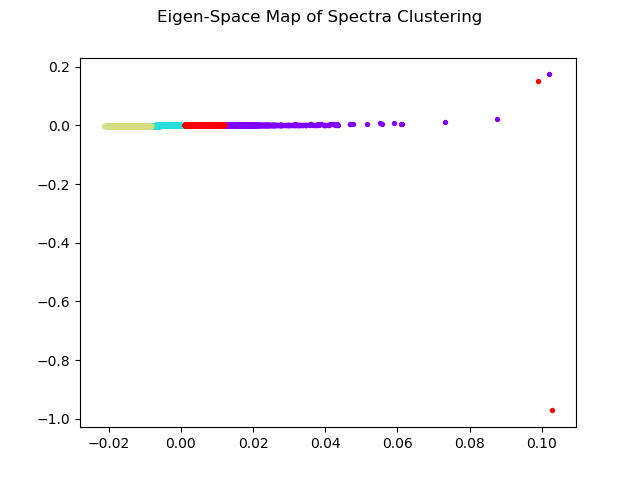
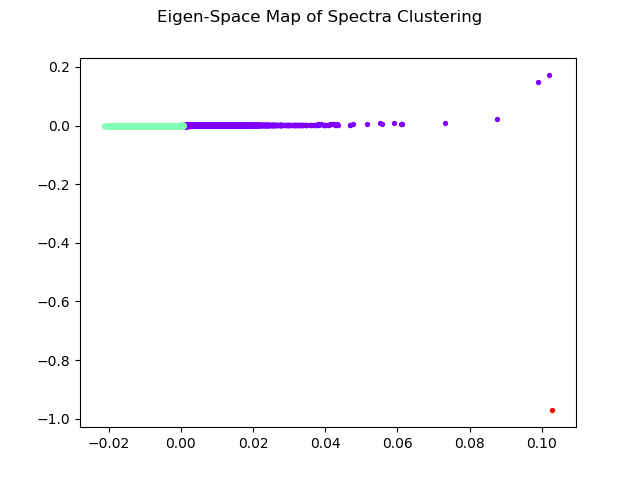
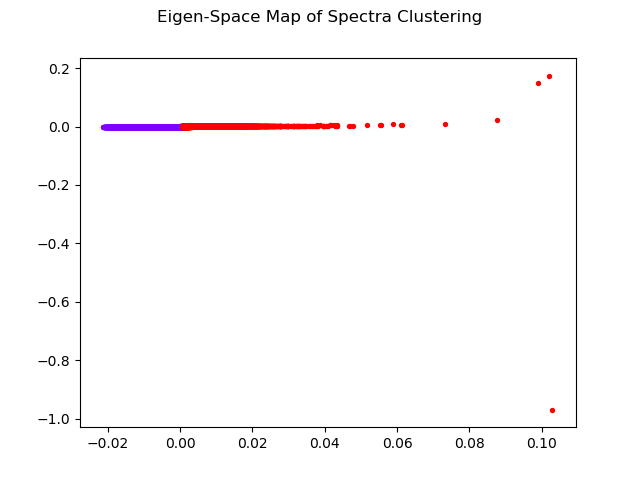
Image 1 Normalize Cut with 2 – 3 – 4 cluster point in Random method,

Image 2 Normalize Cut with 2 – 3 – 4 cluster point in Kmean++ method,

Image 2 Normalize Cut with 2 – 3 – 4 cluster point in Random method,







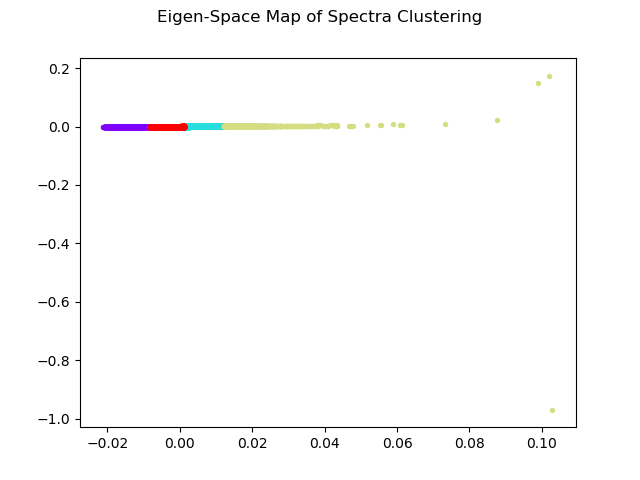
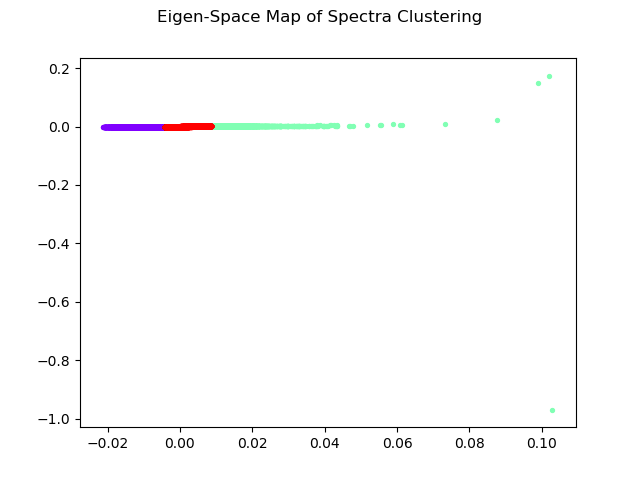
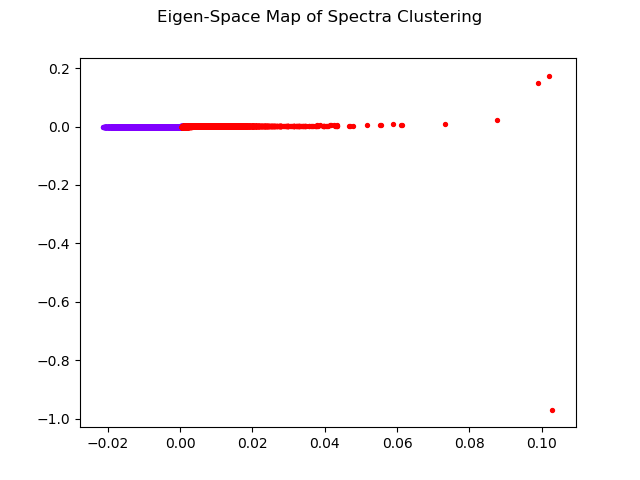


Figure 1: Eigen-space of Cluster point by Normalize Cut. From left to right, from top to bottom.

Image 1 Normalize Cut with 2 – 3 – 4 cluster point in Kmean++ method,

Image 1 Normalize Cut with 2 – 3 – 4 cluster point in Random method,

Image 2 Normalize Cut with 2 – 3 – 4 cluster point in Kmean++ method,

Image 2 Normalize Cut with 2 – 3 – 4 cluster point in Random method,

If you have any concern about my code or other question don’t hesitate to contact me!

My email is: *haiphung106@gmail.com*