**Machine Learning Homework 6**

**K-means & spectral clustering**

**數據所碩一葉詠富310554031**

**Github:** [**https://github.com/frankye1000/NYCU-MachineLearning/tree/master/HW6**](https://github.com/frankye1000/NYCU-MachineLearning/tree/master/HW6)

‣ Part1: You need to make videos or GIF images to show the clustering procedure of your kernel k-means and spectral clustering programs.

‣ Part2: In addition to cluster data into 2 clusters, try more clusters (e.g. 3 or 4) and show your results.

‣ Part3: For the initialization of k-means clustering used in kernel k-means, (e.g. k-means++) and spectral clustering (both normalized cut and ratio cut), try different ways and show corresponding results.

Columns introduction:

Image name: image name.

Initial mean: two type initial mean (1) random、(2)k-means++.

Type: three type method (1)Kernel kmeans、(2)Normalized、(3)Unnormalized.

K: K clusters.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Image name | Initial mean | Type | K(Clusters) | result | GIF link |
| Image1 | random | Kernel kmeans | 2 |  | HW6\GIF\ image1\_random\_kernel\_kmeans\_2Clusters.gif |
| Image1 | random | Kernel kmeans | 3 |  | HW6\GIF\ image1\_random\_kernel\_kmeans\_3Clusters.gif |
| Image1 | random | Kernel kmeans | 4 |  | HW6\GIF\ image1\_random\_kernel\_kmeans\_4Clusters.gif |
| Image1 | random | Normalized | 2 |  | HW6\GIF\ image1\_random\_Normalized\_2Clusters.gif |
| Image1 | random | Normalized | 3 |  | HW6\GIF\ image1\_random\_Normalized\_3Clusters.gif |
| Image1 | random | Normalized | 4 |  | HW6\GIF\ image1\_random\_Normalized\_4Clusters.gif |
| Image1 | random | Unnormalized | 2 |  | HW6\GIF\ image1\_random\_Unnormalized\_2Clusters.gif |
| Image1 | random | Unnormalized | 3 |  | HW6\GIF\ image1\_random\_Unnormalized\_3Clusters.gif |
| Image1 | random | Unnormalized | 4 |  | HW6\GIF\ image1\_random\_Unnormalized\_4Clusters.gif |
| Image1 | k-means++ | Kernel kmeans | 2 |  | HW6\GIF\image1\_kmeans\_pp\_Kernelkmeans\_2Clusters.gif |
| Image1 | k-means++ | Kernel kmeans | 3 |  | HW6\GIF\image1\_kmeans\_pp\_Kernelkmeans\_3Clusters.gif |
| Image1 | k-means++ | Kernel kmeans | 4 |  | HW6\GIF\image1\_kmeans\_pp\_Kernelkmeans\_4Clusters.gif |
| Image1 | k-means++ | Normalized | 2 |  | HW6\GIF\image1\_kmeans\_pp\_Normalized\_2Clusters.gif |
| Image1 | k-means++ | Normalized | 3 |  | HW6\GIF\image1\_kmeans\_pp\_Normalized\_3Clusters.gif |
| Image1 | k-means++ | Normalized | 4 |  | HW6\GIF\image1\_kmeans\_pp\_Normalized\_4Clusters.gif |
| Image1 | k-means++ | Unnormalized | 2 |  | HW6\GIF\image1\_kmeans\_pp\_Unnormalized\_2Clusters.gif |
| Image1 | k-means++ | Unnormalized | 3 |  | HW6\GIF\image1\_kmeans\_pp\_Unnormalized\_3Clusters.gif |
| Image1 | k-means++ | Unnormalized | 4 |  | HW6\GIF\image1\_kmeans\_pp\_Unnormalized\_4Clusters.gif |
|  |  |  |  |  |  |
| Image2 | random | Kernel kmeans | 2 |  | HW6\GIF\image2\_random\_Kernelkmeans\_2Clusters.gif |
| Image2 | random | Kernel kmeans | 3 |  | HW6\GIF\image2\_random\_Kernelkmeans\_3Clusters.gif |
| Image2 | random | Kernel kmeans | 4 |  | HW6\GIF\image2\_random\_Kernelkmeans\_4Clusters.gif |
| Image2 | random | Normalized | 2 |  | HW6\GIF\image2\_random\_Normalized\_2Clusters.gif |
| Image2 | random | Normalized | 3 |  | HW6\GIF\image2\_random\_Normalized\_3Clusters.gif |
| Image2 | random | Normalized | 4 |  | HW6\GIF\image2\_random\_Normalized\_4Clusters.gif |
| Image2 | random | Unnormalized | 2 |  | HW6\GIF\image2\_random\_Unnormalized\_2Clusters.gif |
| Image2 | random | Unnormalized | 3 |  | HW6\GIF\image2\_random\_Unnormalized\_3Clusters.gif |
| Image2 | random | Unnormalized | 4 |  | HW6\GIF\image2\_random\_Unnormalized\_4Clusters.gif |
| Image2 | k-means++ | Kernel kmeans | 2 |  | HW6\GIF\image2\_kmeans\_pp\_Kernelkmeans\_2Clusters.gif |
| Image2 | k-means++ | Kernel kmeans | 3 |  | HW6\GIF\image2\_kmeans\_pp\_Kernelkmeans\_3Clusters.gif |
| Image2 | k-means++ | Kernel kmeans | 4 |  | HW6\GIF\image2\_kmeans\_pp\_Kernelkmeans\_4Clusters.gif |
| Image2 | k-means++ | Normalized | 2 |  | HW6\GIF\image2\_kmeans\_pp\_Normalized\_2Clusters.gif |
| Image2 | k-means++ | Normalized | 3 |  | HW6\GIF\image2\_kmeans\_pp\_Normalized\_3Clusters.gif |
| Image2 | k-means++ | Normalized | 4 |  | HW6\GIF\image2\_kmeans\_pp\_Normalized\_4Clusters.gif |
| Image2 | k-means++ | Unnormalized | 2 |  | HW6\GIF\image2\_kmeans\_pp\_Unnormalized\_2Clusters.gif |
| Image2 | k-means++ | Unnormalized | 3 |  | HW6\GIF\image2\_kmeans\_pp\_Unnormalized\_3Clusters.gif |
| Image2 | k-means++ | Unnormalized | 4 |  | HW6\GIF\image2\_kmeans\_pp\_Unnormalized\_4Clusters.gif |

‣ Part4: For spectral clustering (both normalized cut and ratio cut), you can try to examine whether the data points within the same cluster do have the same coordinates in the eigenspace of graph Laplacian or not. You should plot the result and discuss it in the report.

Image: images1

Initial mean: k-means++

K(Clusters): 3

Discuss:

(1) Unnormalized clustering results are more rough, and it has a better result on land segmtation.

|  |  |
| --- | --- |
|  |  |

Image: images2

Initial mean: k-means++

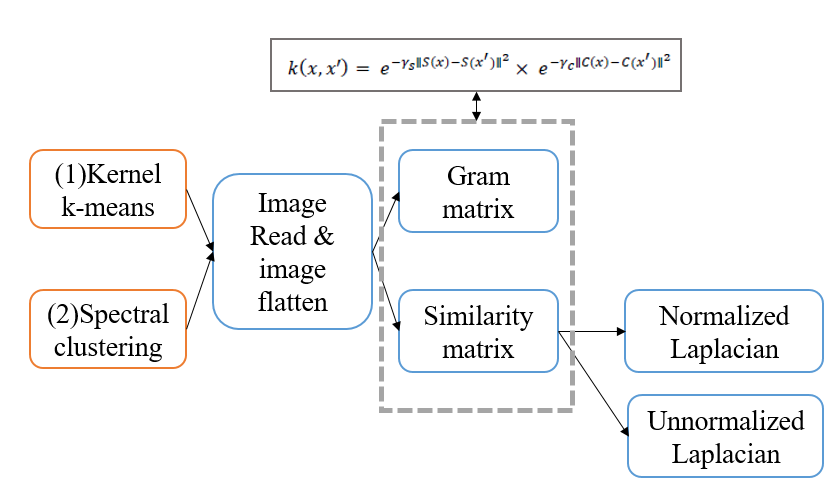
K(Clusters): 3

Discuss:

(1) Unnormalized clustering have same outliers. Outliers will affect the clustering results.

|  |  |
| --- | --- |
|  |  |

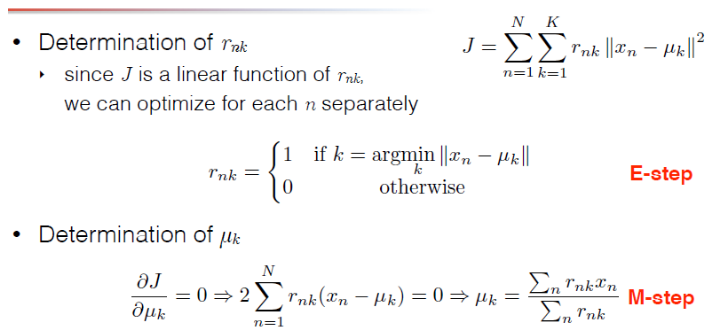
**Process:**



**Code:**

(1) Kernel k-means

* Use EM algorithm to do k-means.

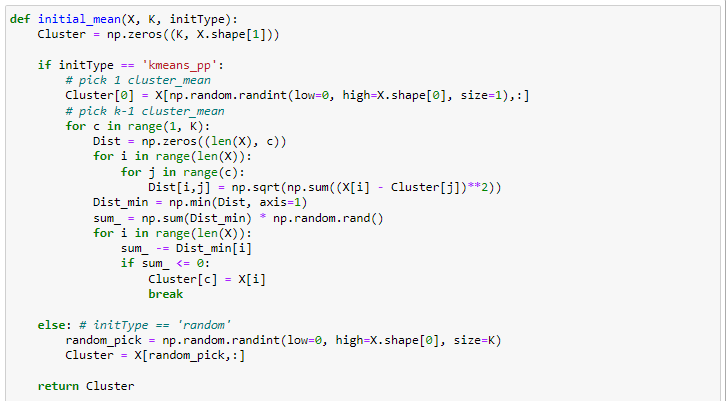


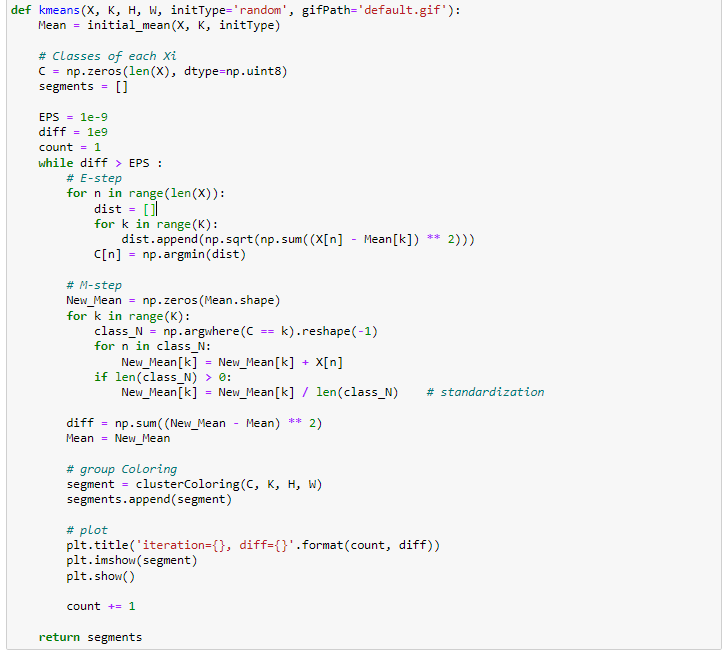
* Define two type initial mean(random, k-means++).

Random: random pick initial mean.

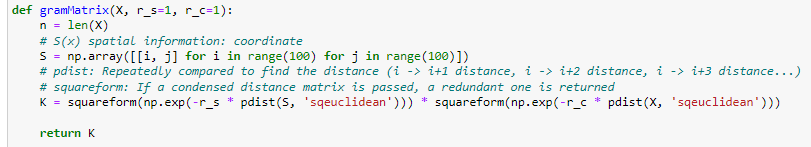
K-means++: When selecting a new cluster center, the **farther** the point is from the existing cluster

center, the greater the probability of being selected as the cluster center.





* Use gram matrix to calculate kernel k-means.

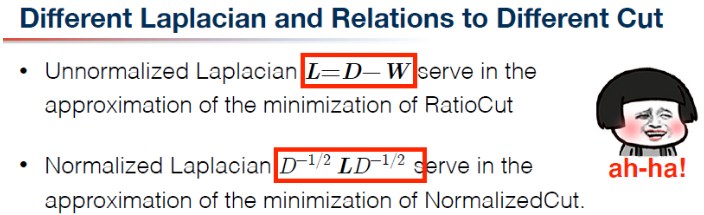


* Main function to set kernel k-means parameter.

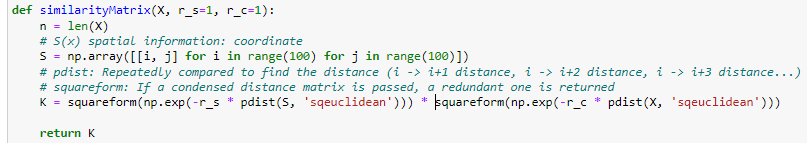


(2) Spectral clustering

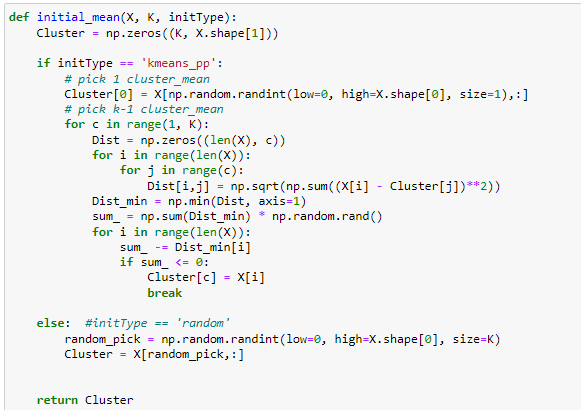
* Use different laplacian to do different cut.

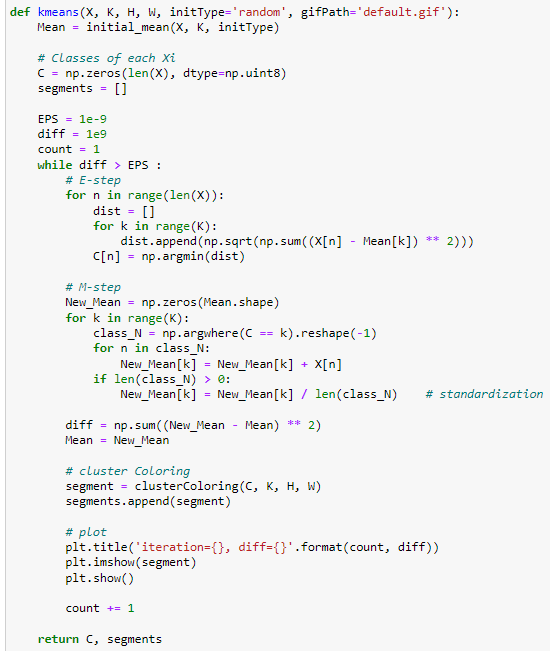


* Calculate two type cut by similarity matrix.

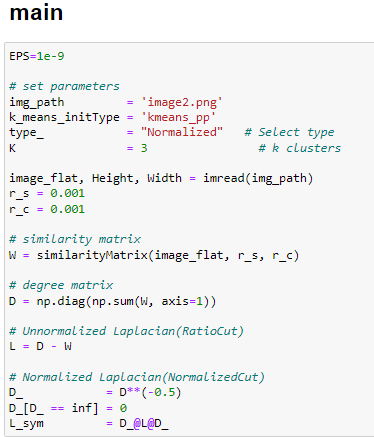


* Use the same k-means & initial mean.





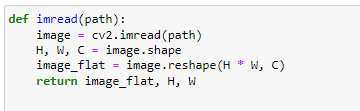
* Main function to set spectral clustering parameter, and calculate Normalized Laplacian and Unnormalized Laplacian.
* Also use np.linalg.eig() to get eigenvalue, eigenvector.



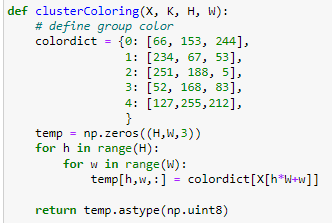


**Tool function:**

(1) read image & image flatten.



(2) define every cluster color.



(3) save array to gif.

