TP1 Dadoun Hind

November 7, 2018

1 Graphs in Machine learning : Homework 1

```
In [1]: from graph_construction.generate_data import blobs, two_moons
```

```
In [2]: from graph_construction.func import plot_similarity_graph,how_to_choose_epsilon
```

```
In [3]: from spectral_clustering.funcs import two_blobs_clustering,parameter_sensitivity,find_th
```

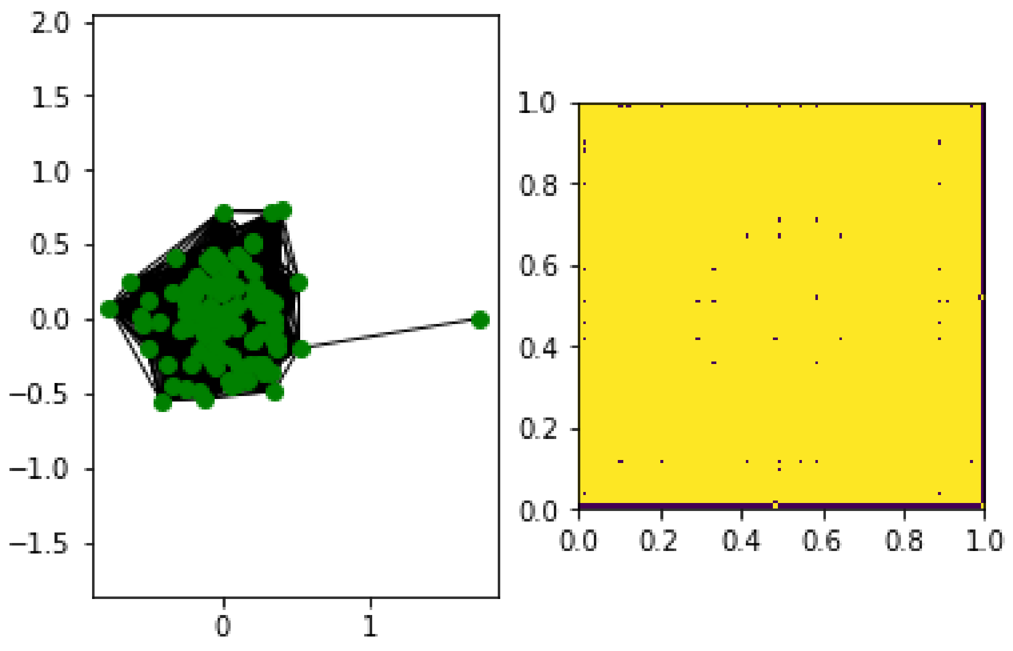
```
In [4]: from image_segmentation.image_segmentation import image_segmentation
        from IPython.display import Latex
```

```
In [5]: X,Y=blobs(100,2,10,surplus=0)
        X1,Y1= two_moons(100,2,1)
```

1.1. What is the purpose of the option parameter in worst_case_blob?
The gen parameter is added to the value of the last point in the data generated by skd.make_blobs. It's an outlier usefull if we want to make sure the function how_to_choose_epsilon allows us to get a fully connected graph.
1.2. What happens when you change the generating parameter of worst_ case_blob.m in how_to_choose_espilon.m and run the function? What if the parameter is very large?
When we change the parameter, the function builds a similarity graph but this time choosing a specific ϵ such that the minimum distance between two points is equal to the gap between the point added by worst_case_blob and the other points. This is equivalent to choosing ϵ as the length of the longest edge in the minimum spanning tree.

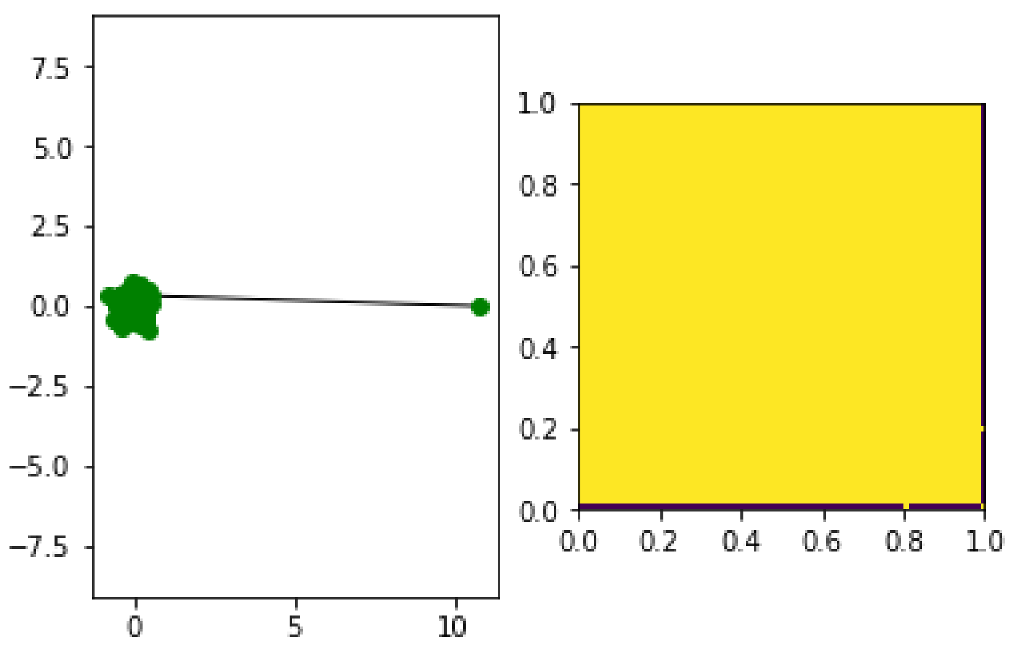
```
In [6]: how_to_choose_epsilon(1)
```

```
0.2233330262548987
```



```
In [7]: how_to_choose_epsilon(10)
```

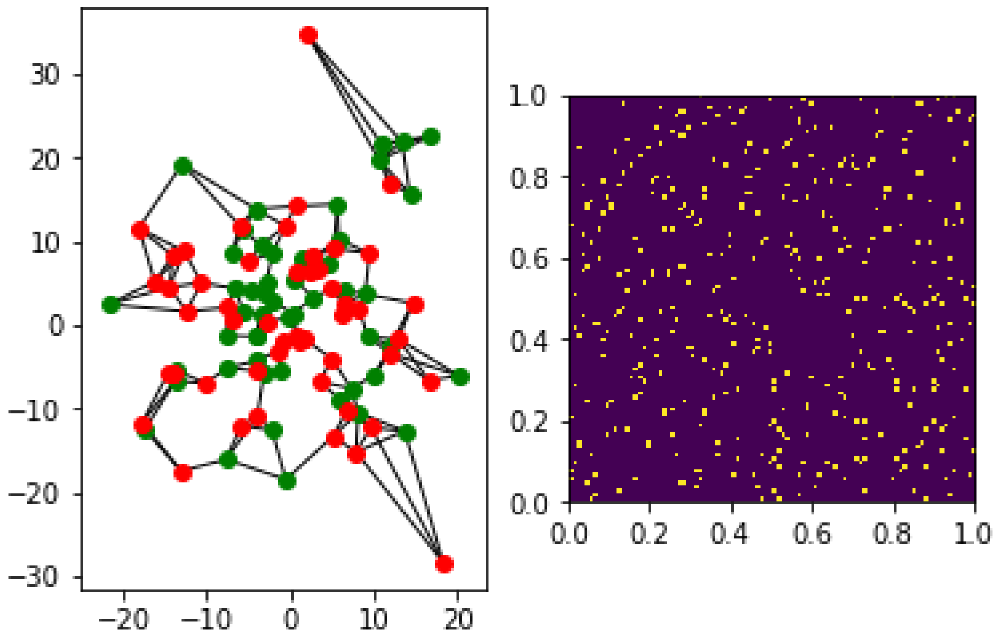
```
2.317064555902338e-46
```



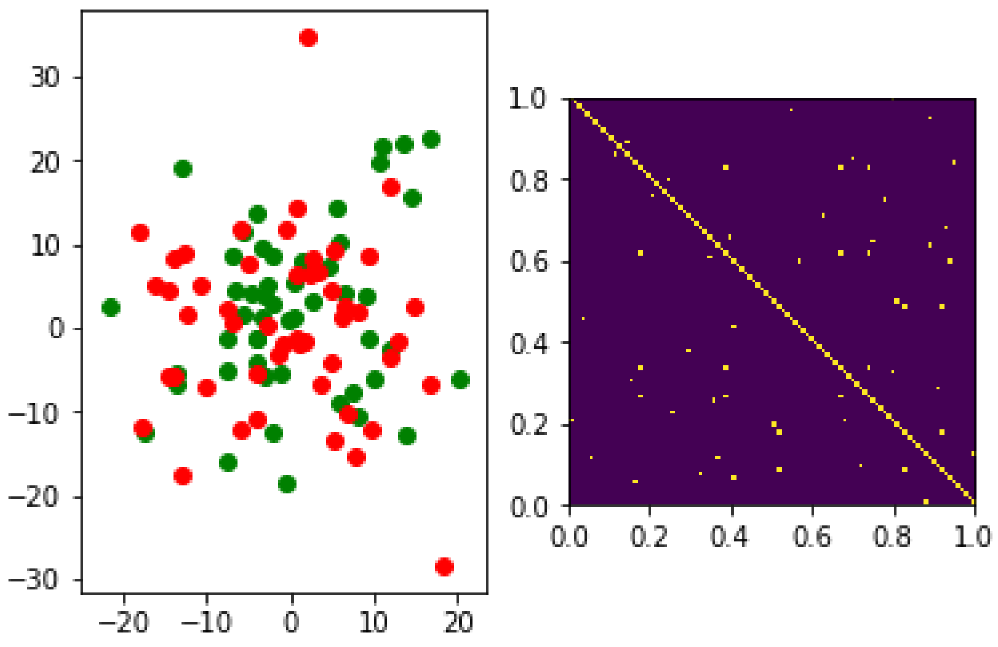
1.3. Using plot_similarity_graph and one of the datasets, compare k- nn to graphs. When is it easier to build a connected graph using k-nn? When using graphs?

If we have data on different scales, meaning the distances between data points are different in different regions of the space it is better to choose K-nn because it can still connect points on different scales.

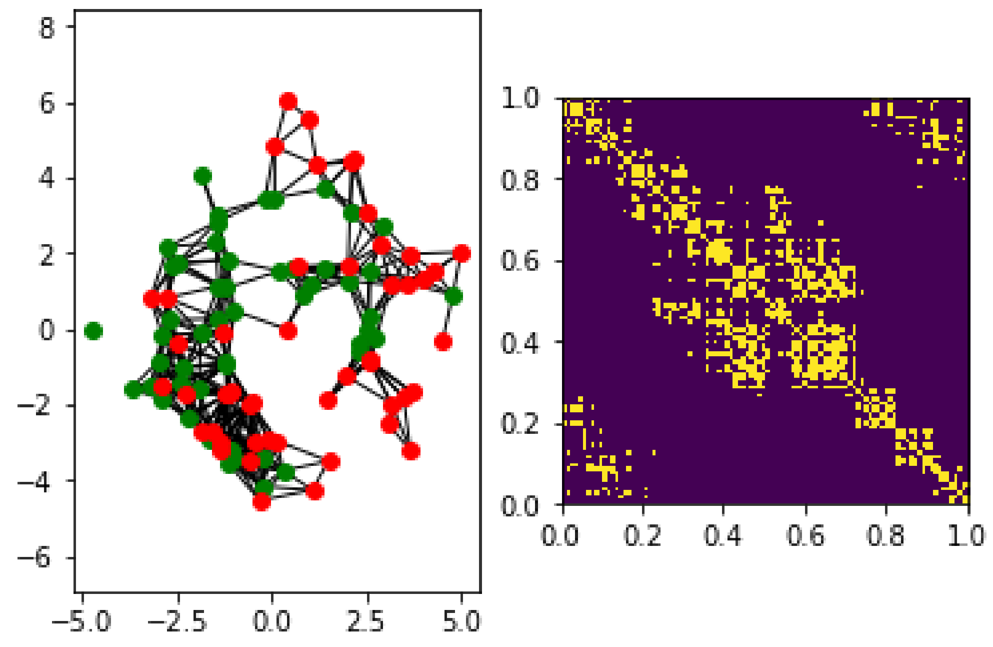
```
In [7]: plot_similarity_graph(X,Y,k=3,eps=0)
```



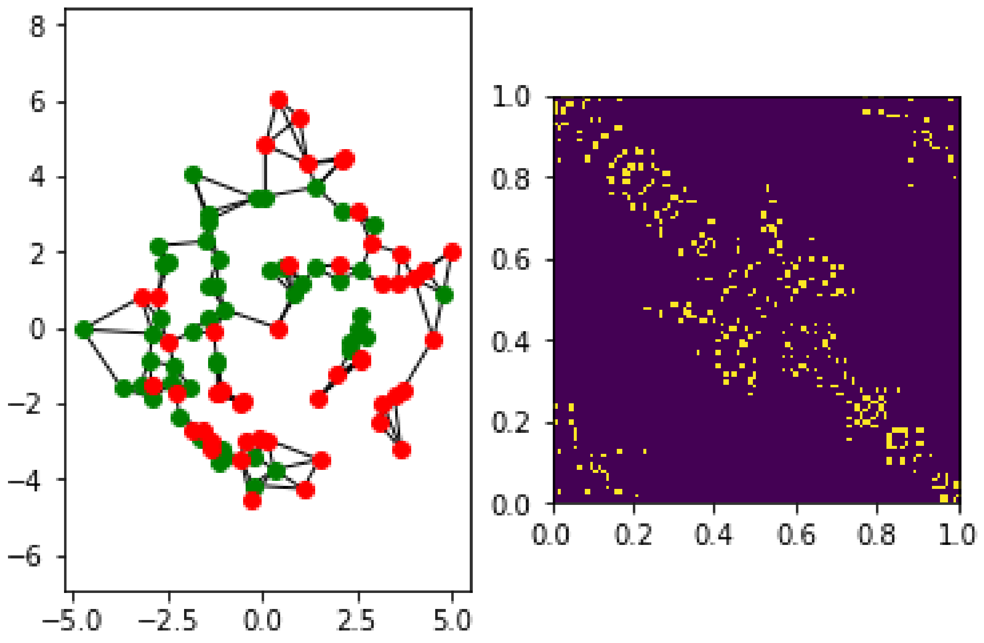
```
In [8]: plot_similarity_graph(X,Y,eps=0.1,k=0)
```



```
In [9]: plot_similarity_graph(X1,Y1,eps=0.1,k=0)
```



```
In [10]: plot_similarity_graph(X1,Y1,k=3,eps=0)
```



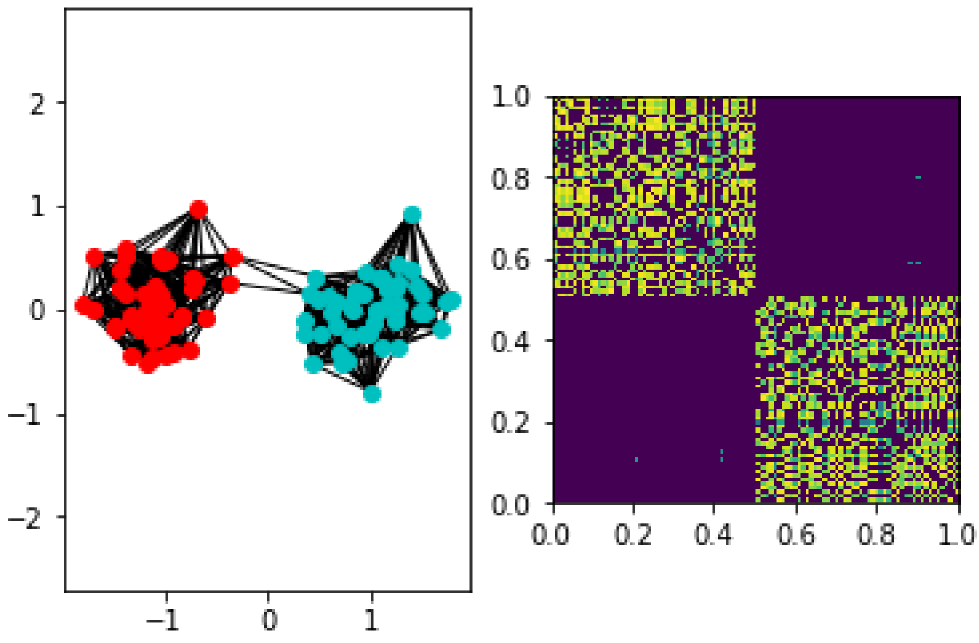
2.1. Build a graph starting from the dataloaded intwo_blobs_clustering. m, and remember to keep the graph connected. Motivate your choice on which eigenvectors to use and how you computed the clustering assignments from the eigenvectors. Now compute a similar clustering using the builtin k-means and compare the results.

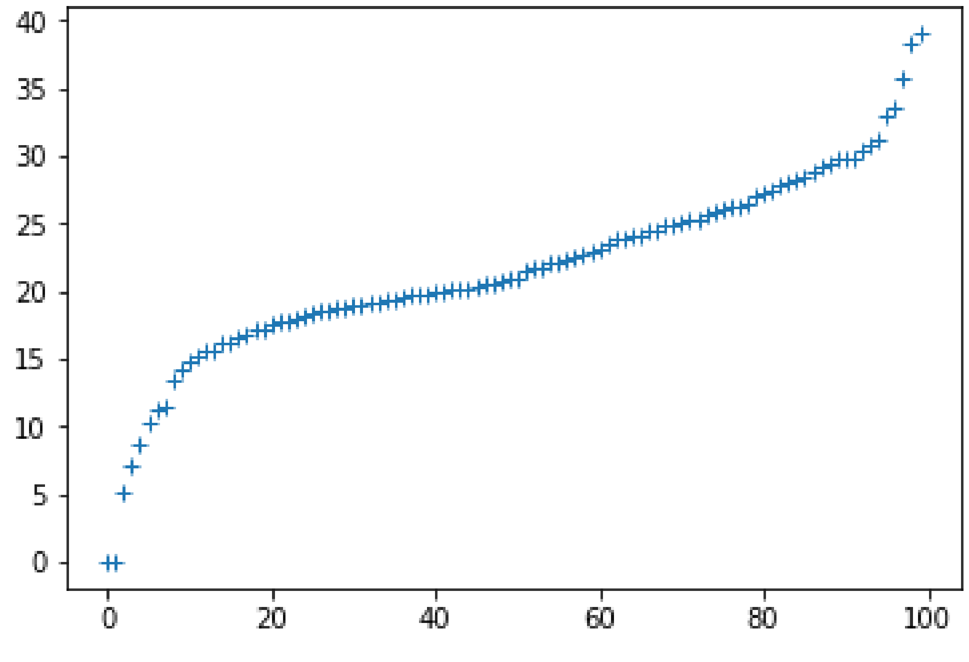
The Rayleigh-Ritz theorem states that the answer to the minimazation problem are the eigenvectors corresponding to the k-smallest eigenvalues (not including 0 which corresponds to eigenvector 1_N). So here we now that the data has only two labels, we can use the eigenvectors corresponding to the 2 smallest eigen values.

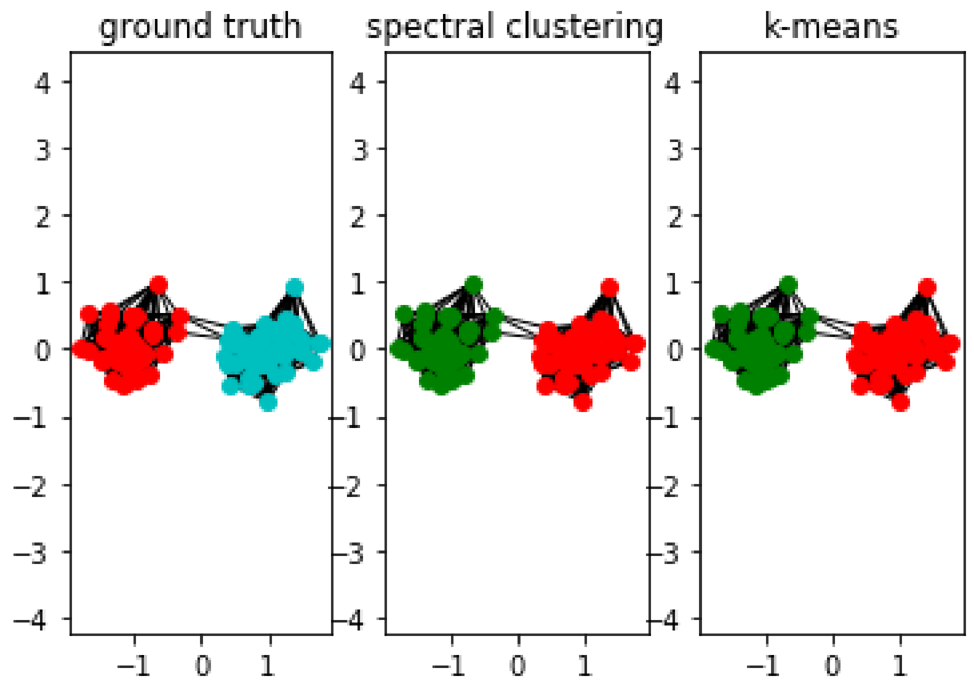
2.2. Build a graph starting from the data loaded in two_blobs_clustering. m, but this time makes it so that the two components are separate. How do you choose which eigenvectors to use in this case? Motivate your answer.

In the situation of k disconnected clusters, the eigenvalue 0 has multiplicity k. So we can choose the first two eigen vectors associated with the two null eigenvalues, this way we know they belong to the connected part of the graph.

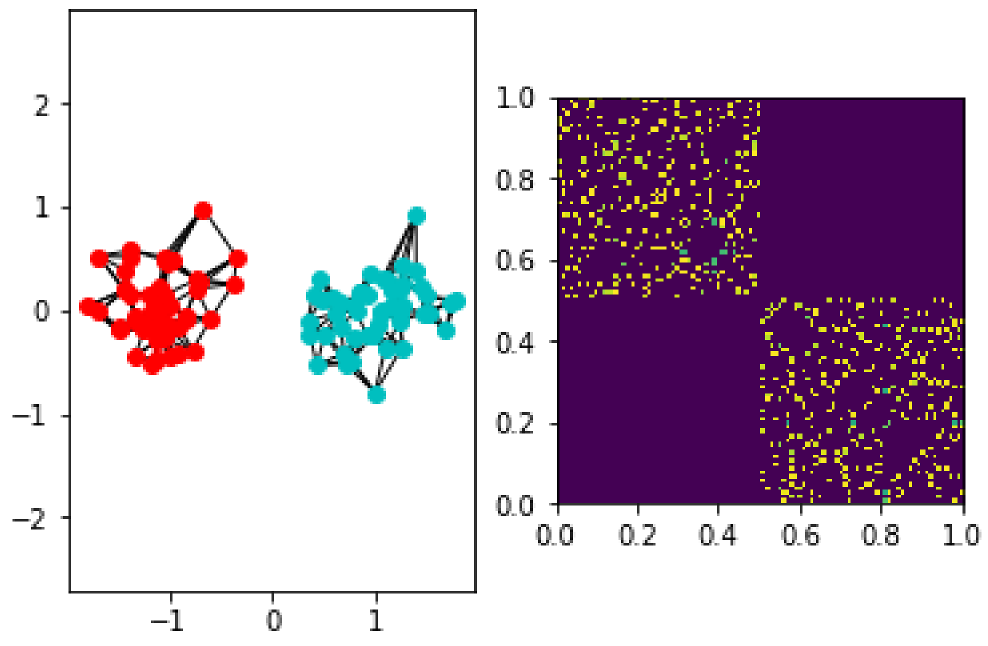
```
In [17]: ##connected:
         two_blobs_clustering(eps=0,var=1,k=20)
```

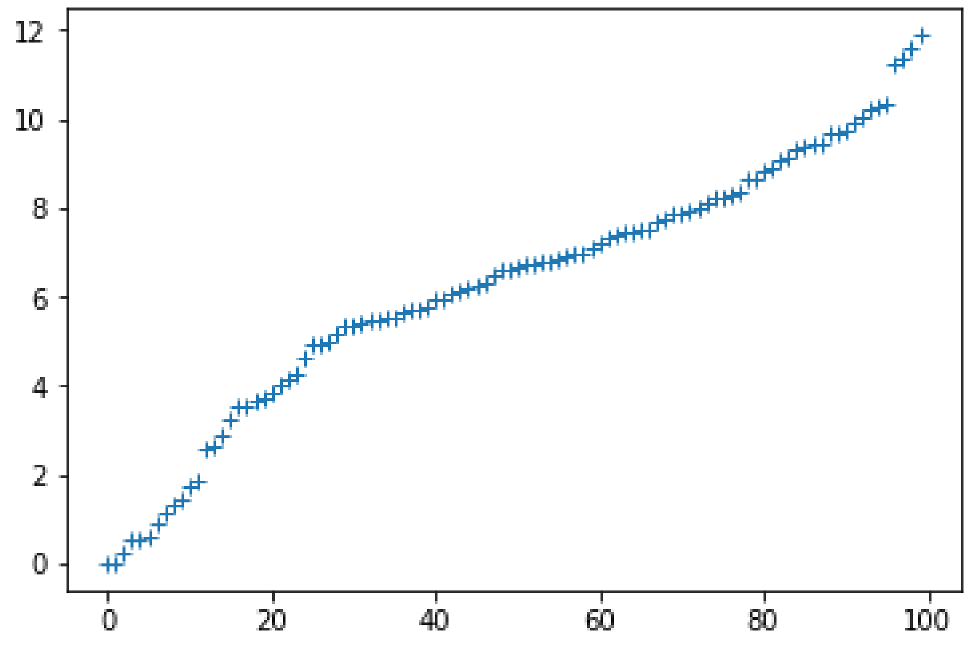

2
1
0
−1
−2
−1
0
1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

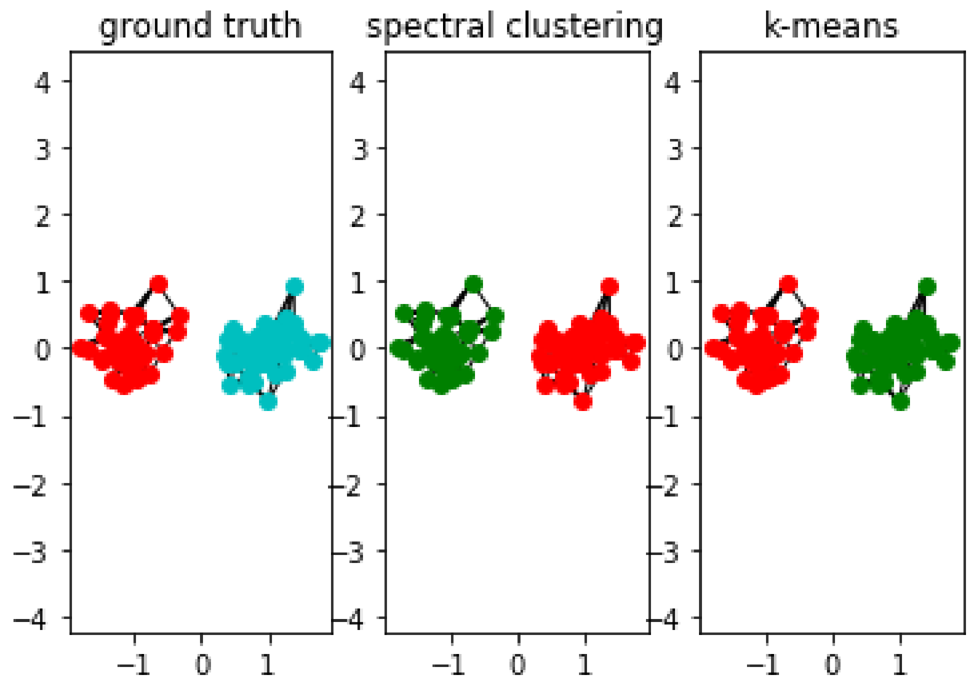

40
35
30
25
20
15
10
5
0
0
20
40
60
80
100



```
In [15]: ##not connected:
         two_blobs_clustering(eps=0,var=1,k=5)
```

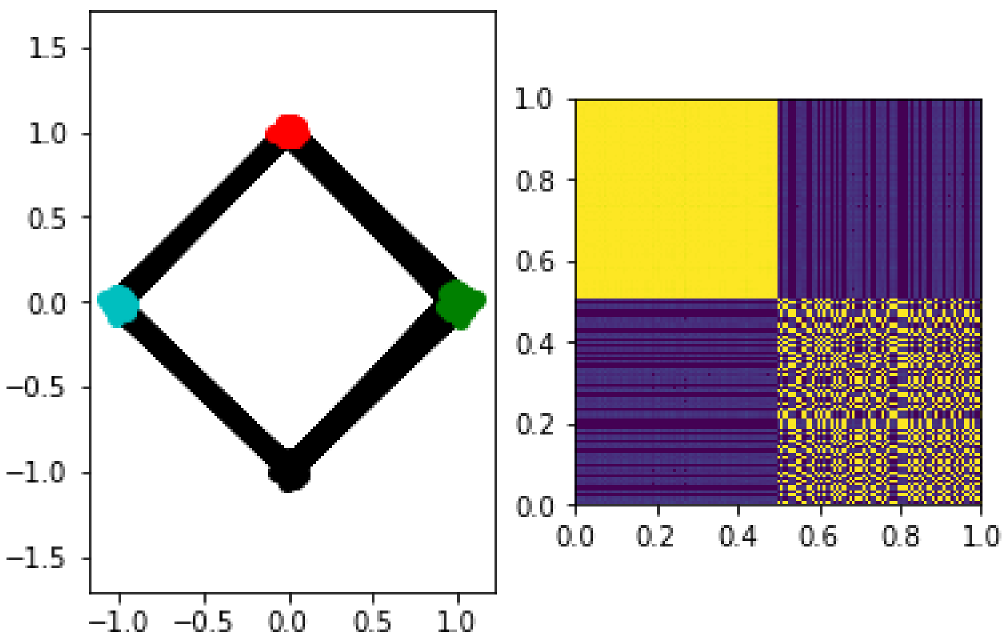




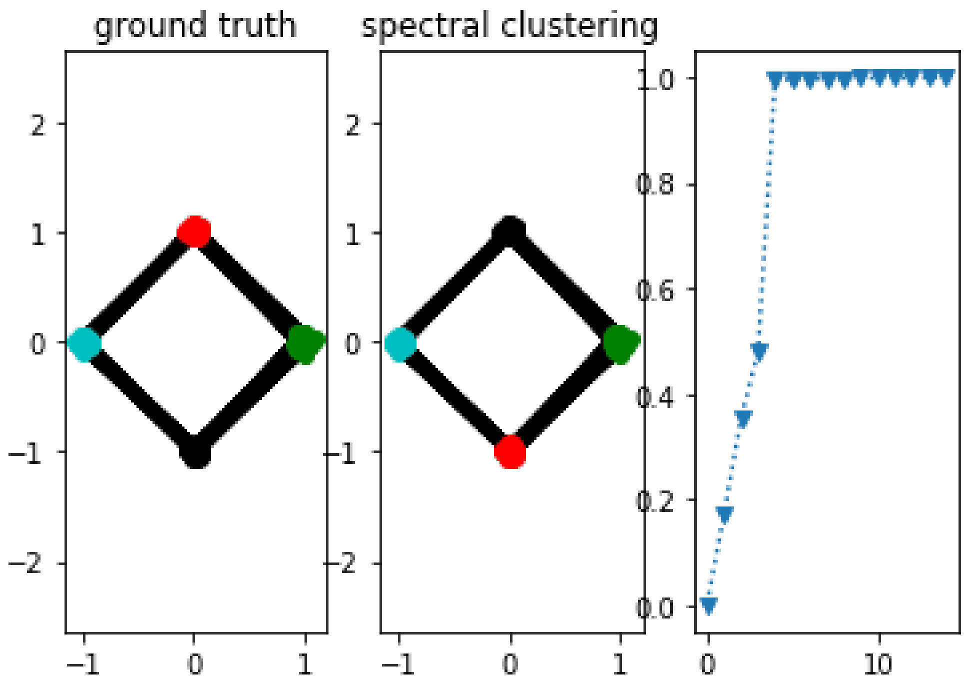


2.3.Look at find_the_bend.m. Generate a dataset with 4 blobs and 2 = 0.03. Build a graph out of it and plot the first 15 eigenvalues of the Lapla- cian. Complete choose_eig_function.m to automatically choose the number of eigenvectors to include. The decision rule must adapt to the actual eigenvalues of the problem.

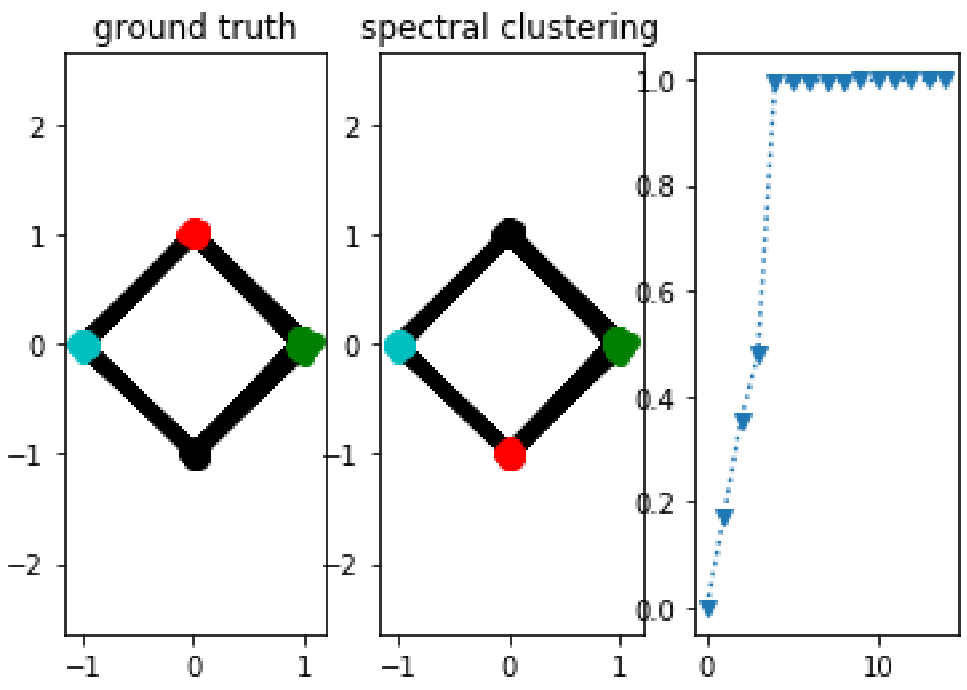
```
In [7]: find_the_bend(0.03,eps=0.1,k=0)
```



```
adaptative
```



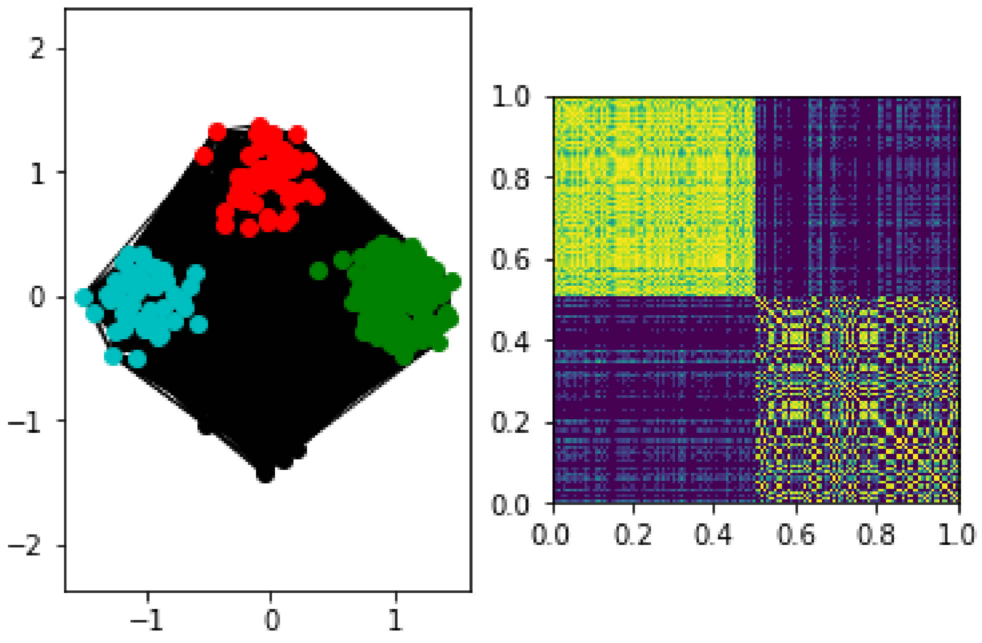
```
non adaptative
```



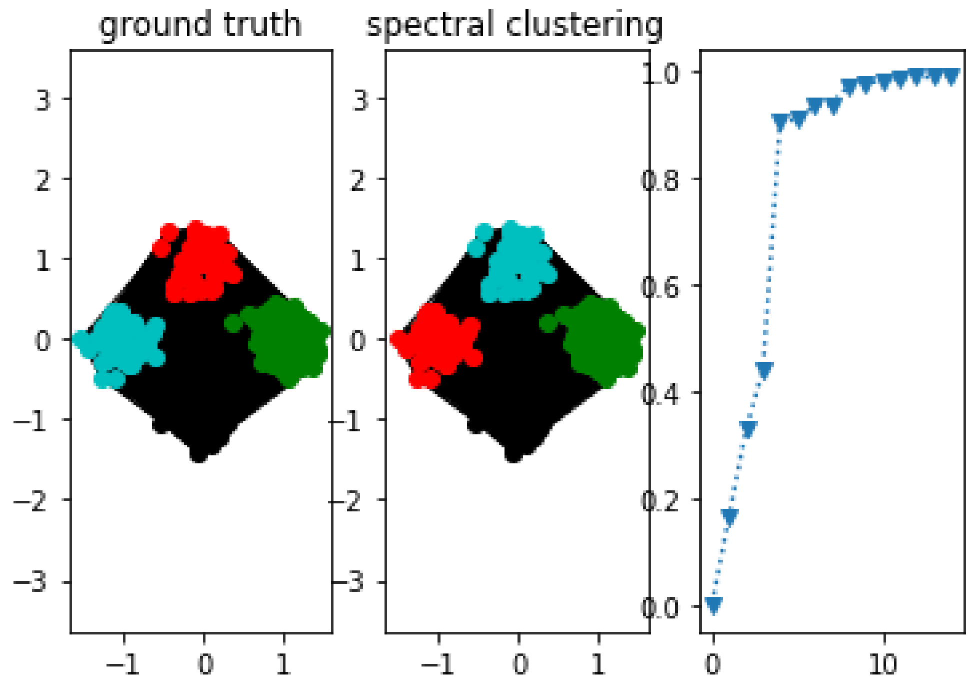
2.4. Now increase the variance of the Blobs to 2 = 0.20 as you keep plot- ting the eigenvalues. Use choose_eig_function.m. Do you see any difference?

We can see that when you increase the variance the eigen values tend to become closer and it would be harder to use the elbow method for cluster detection.

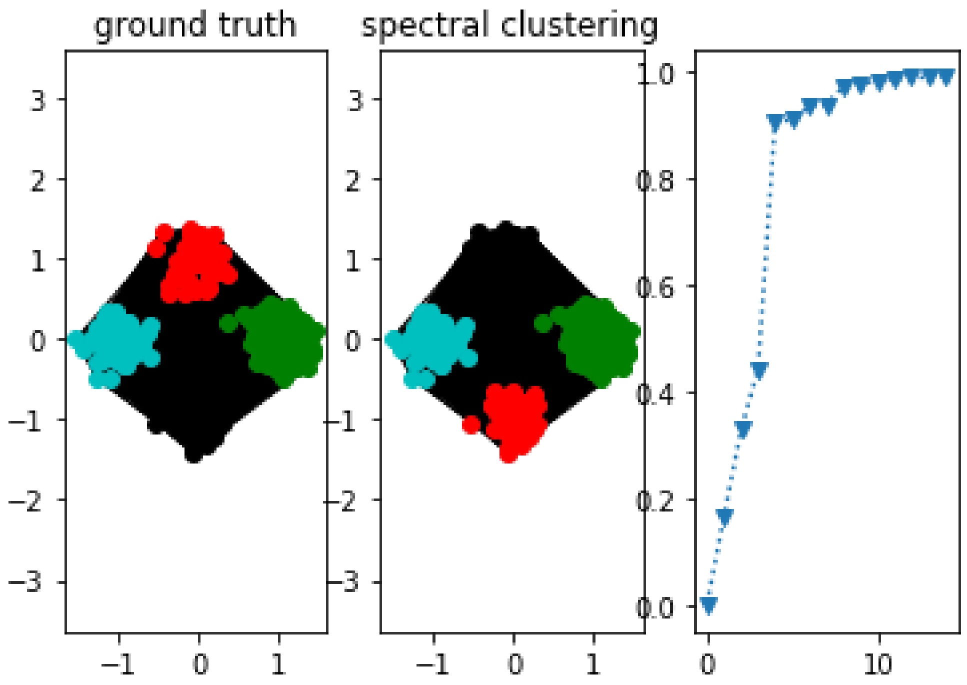
```
In [11]: find_the_bend(0.2,eps=0.1,k=0)
```



`adaptative`



```
non adaptative
```



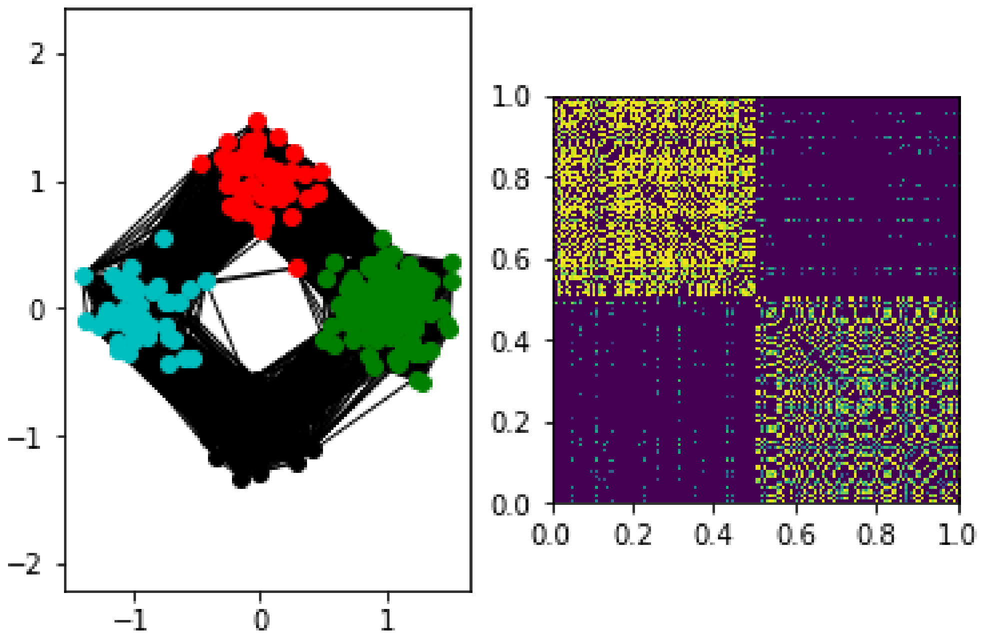
2.5. When you built the cluster assignment did you use thresholding, kmeans or both? Do you have any opinion on when to use each?

I first chose the tresholding parameter because it performed better when I used the usual values (k=6 to 12 and eps=0.01). Then I decided to take a very large K so that all clusters are connected and I got the same results as ϵ graph. The k-nearest neighbor graph can fall into several disconnected components if there are high density regions which are reasonably far away from each other which is the case here.

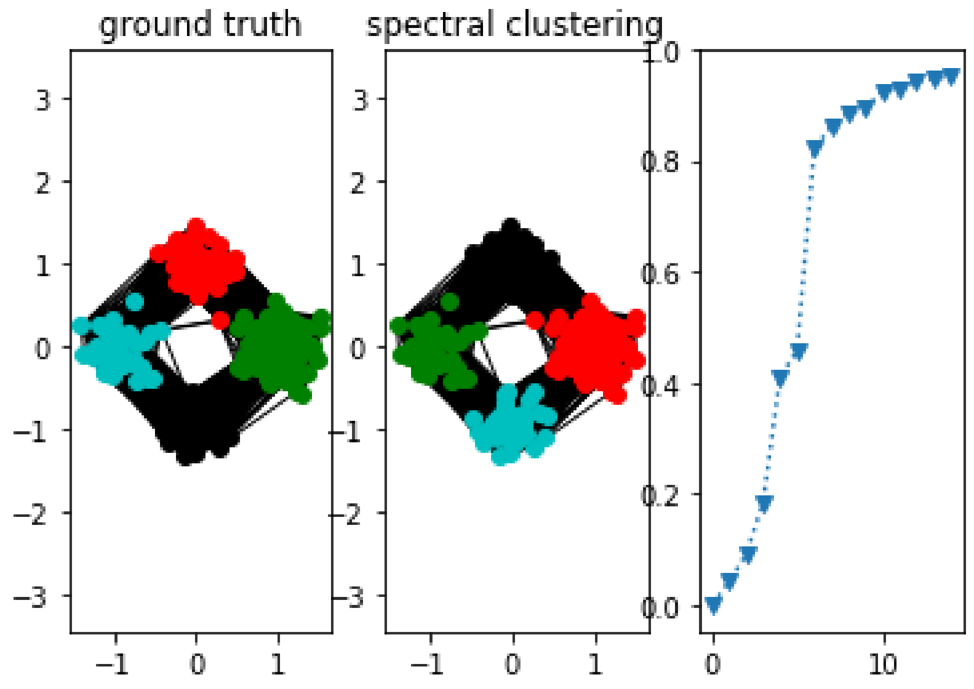
2.6. What is another use that looking at the distribution of the eigenvalues can have during clustering, beside choosing which eigenvectors to include?

If we don't have information on how much clusters we have, looking at the eigenvalue graph we can guess how much clusters we have.

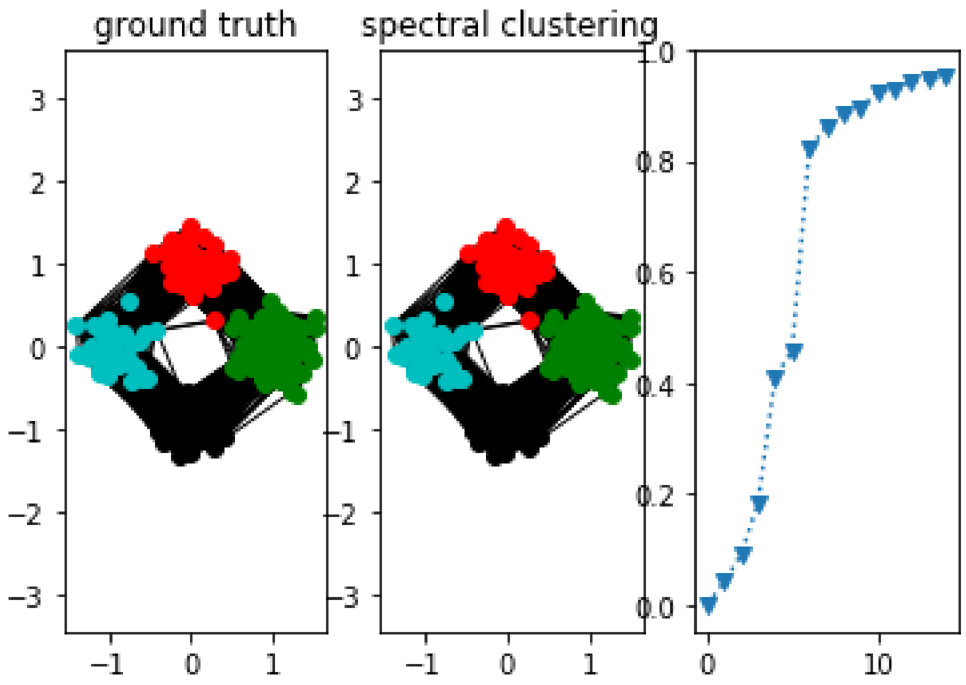
```
In [18]: find_the_bend(0.2,eps=0,k=60)
```


2
1
0
−1
−2
−1
0
1
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0

`adaptative`


ground truth
spectral clustering
3
2
1
0
−1
−2
−3
−1
0
1
1.0
0.8
0.6
0.4
0.2
0.0
0
10

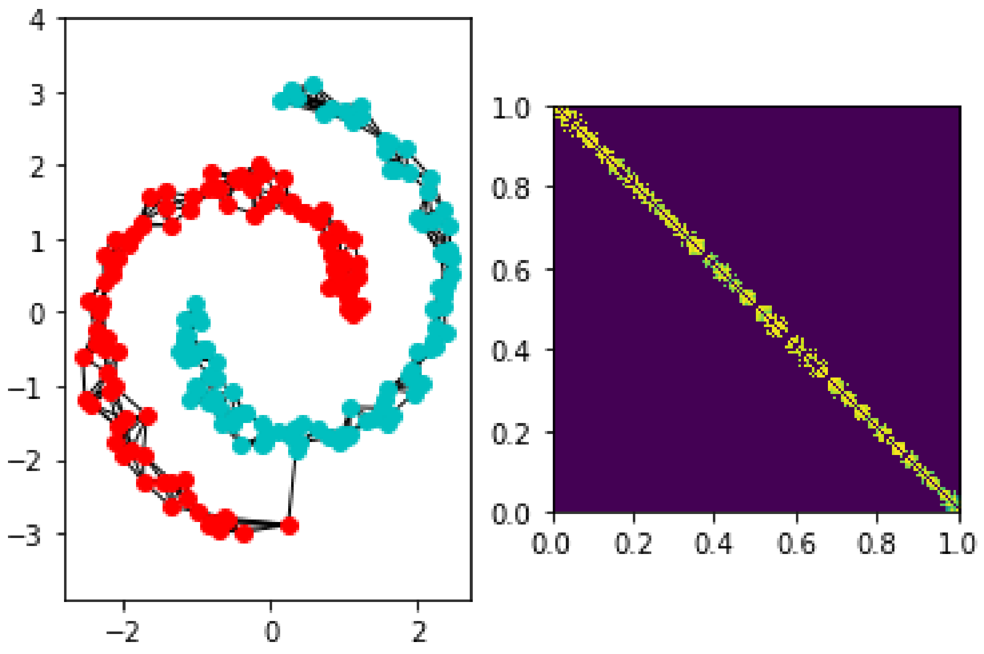
```
non adaptative
```



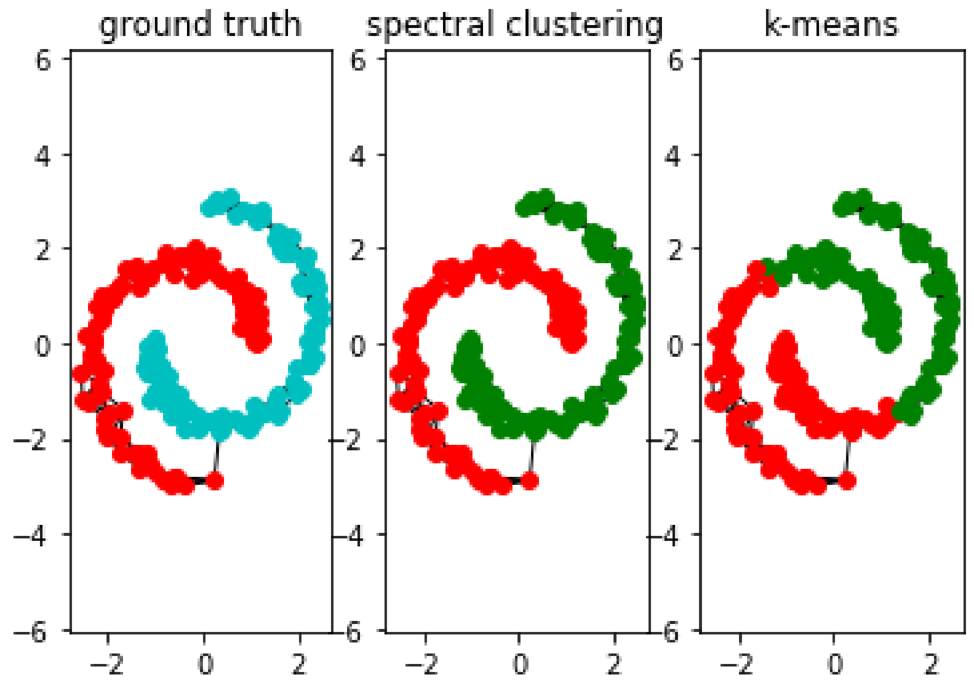
2.7. Plot your results using spectral clustering and k-means in two_moons_ clustering.m and compare the results. Do you notice any difference? Taking into consideration the graph structure, can you explain them?

If our data looked like a mixture of gaussians we wouldn't care about spectral clustering but for data where we don't have compactness and only connectivity K-means will totally fail and spectral clustering will work.

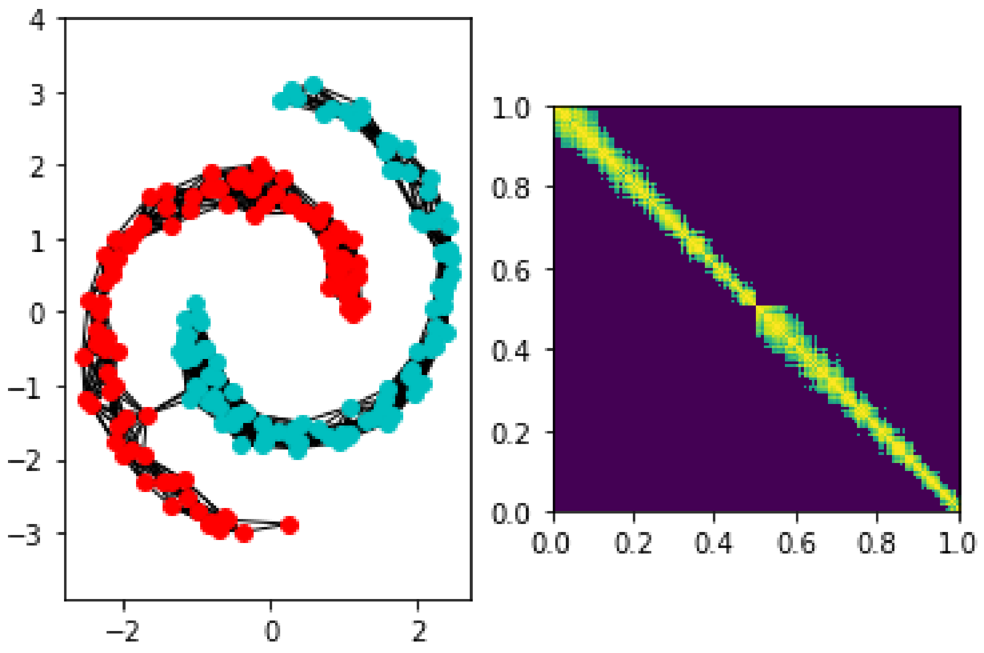
```
In [6]: two_moons_clustering(k=6,eps=0)
```



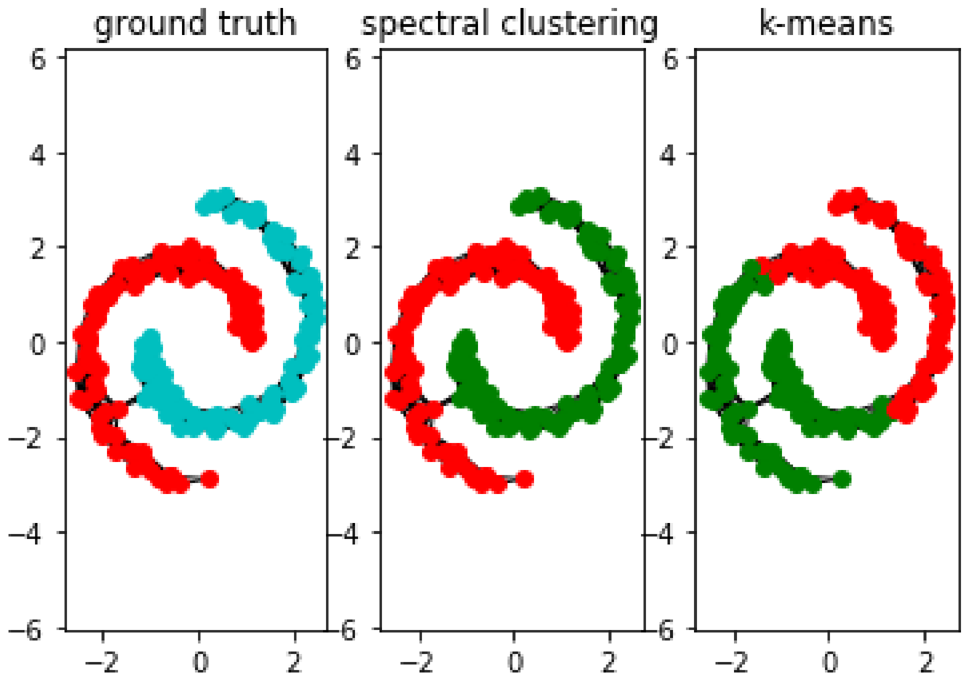
non adaptative



```
In [7]: two_moons_clustering(k=0,eps=0.5)
```



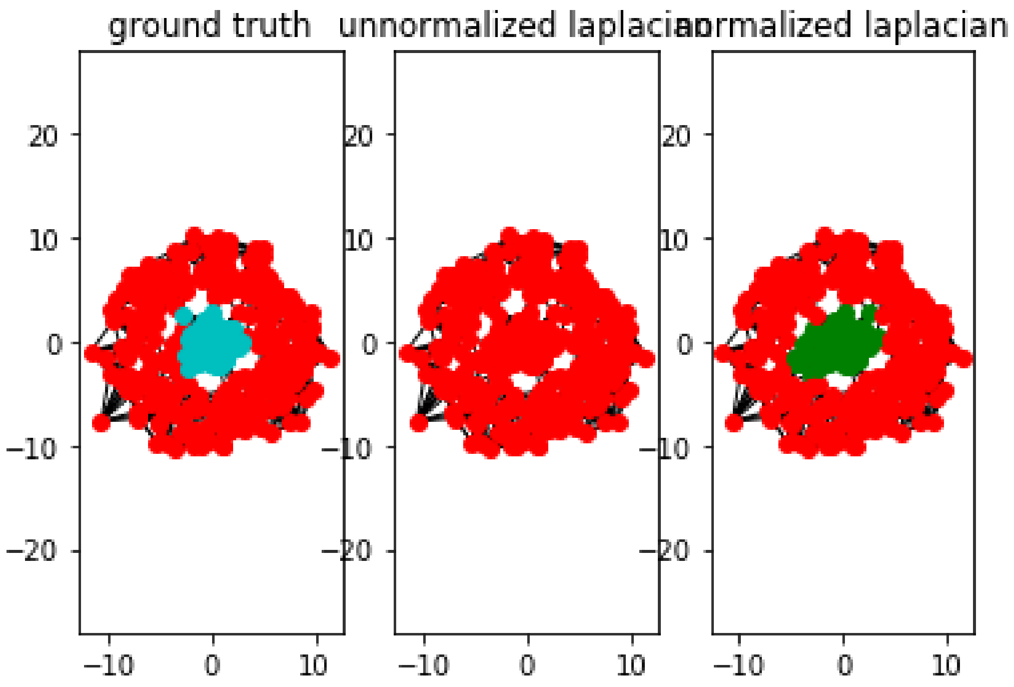
```
non adaptative
```



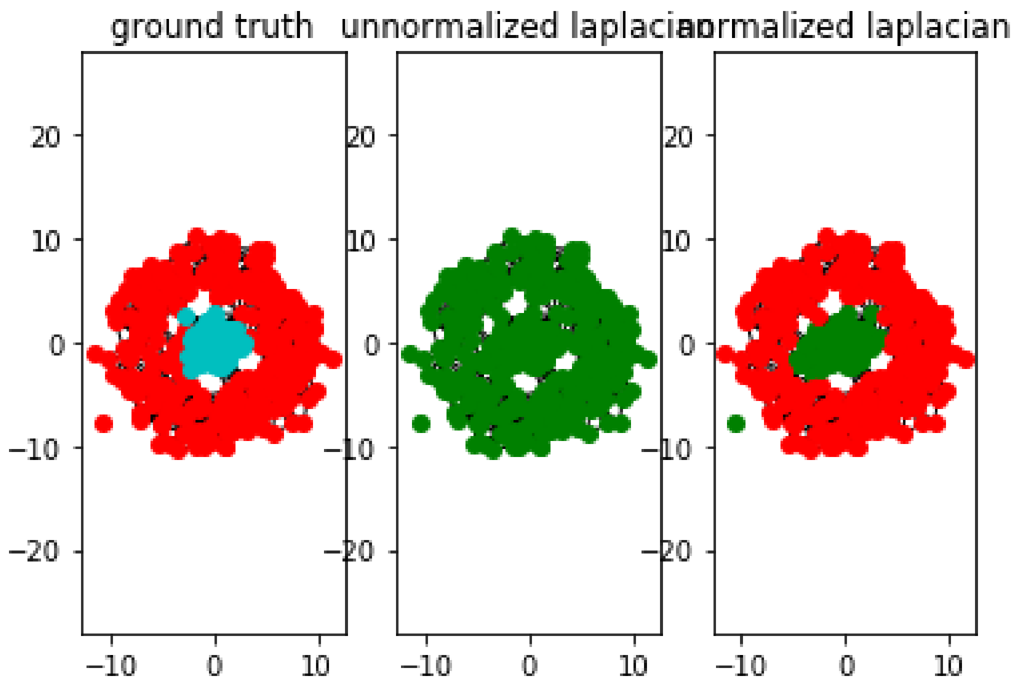
2.8. point_and_circle_clustering.m will compare spectral clustering using the normal laplacian L and the random-walk regularized Laplacian Lrw. Do you notice any difference? Taking into consideration the graph structure, can you explain them?

We've seen that N-cut leads to normalized spectral clustering, while relaxing Ratio-Cut leads to unnormalized spectral clustering.In Ratio-Cut, the size of a subset A of a graph is measured by its number of vertices |A|, while in N-cut the size is measured by the weights of its edges vol(A). The cut cost of a potential segmentation of this graph takes into account both distances between points and the density of points. This makes the N-cut algorithm more robust to false connections.

```
In [19]: point_and_circle_clustering(eps=0,k=12,var=1)
```



```
In [18]: point_and_circle_clustering(eps=0.001,k=0,var=1)
```



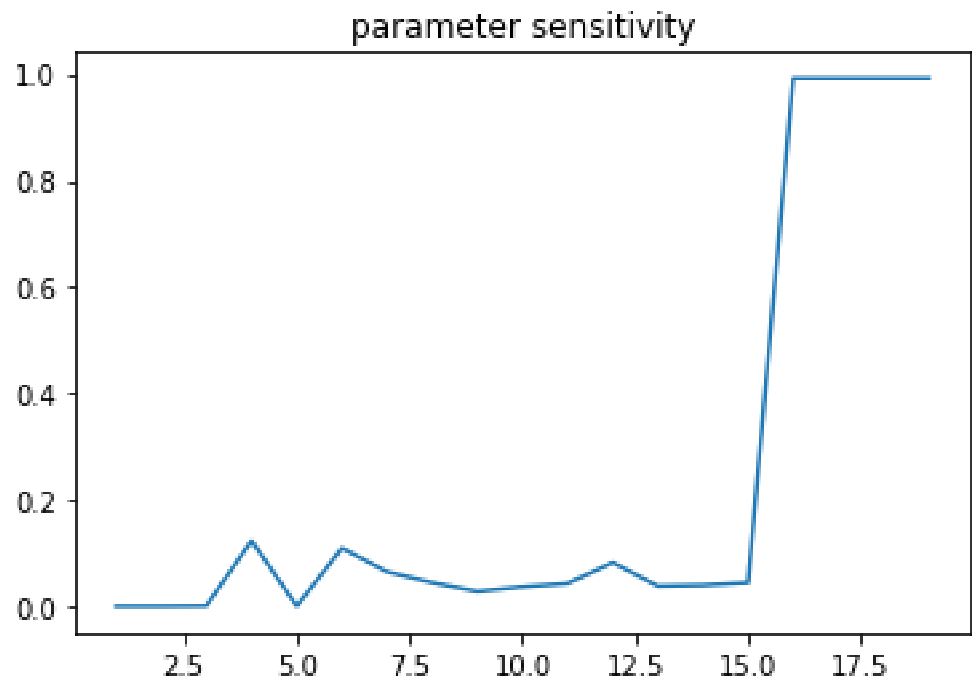
2.9. Complete parameter_sensitivity.m, and generate a plot of the ARI index while varying one of the parameters in the graph construction or k. Comment on the stability of spectral clustering.

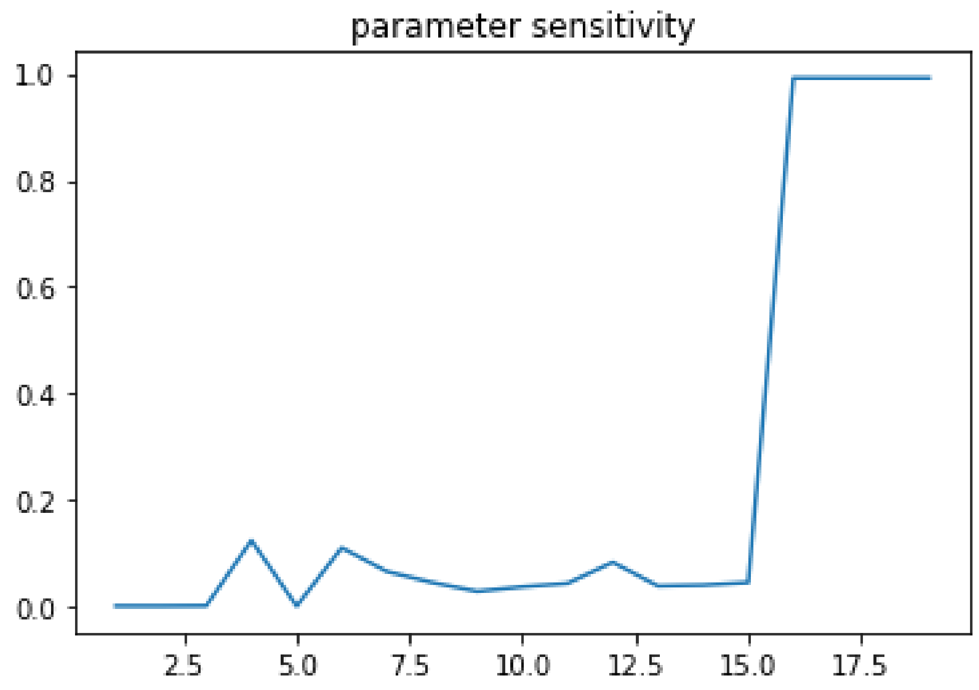
It seems that spectral clustering is quite sensitive to changes in the similarity graph and to the choice of its parameters.

2.10. If we did not have access to true1 labels how could we evaluate the clustering result (or what should we not use as evaluation)?

Inertial indices are the best known and most used to evaluate the quality of a classification. The intra-class inertia makes it possible to measure the degree of homogeneity between the objects belonging to the same class.Inter-class inertia measures the degree of heterogeneity between classes.BUT for data like two moons clustering it's a bad idea, as we've seen the K-mean algorithm performs very badly.

```
In [6]: parameter_sensitivity()
```



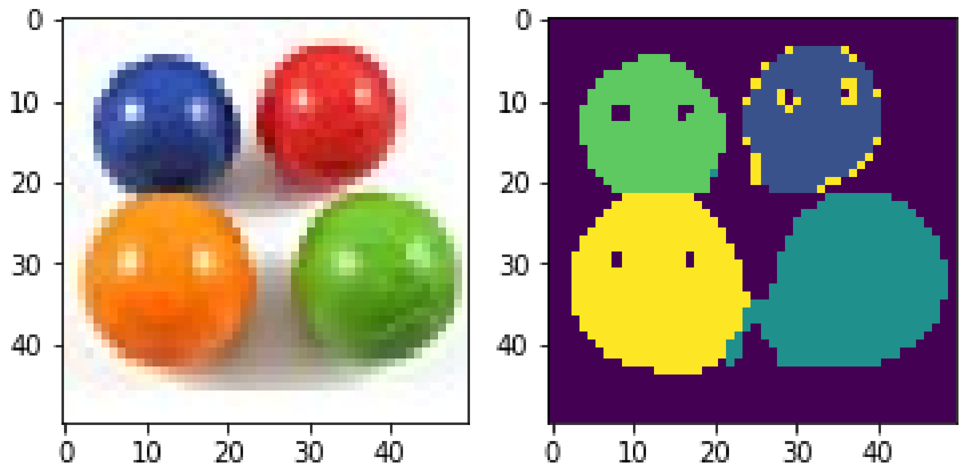


3.2. Can you think two simple techniques to reduce the computational and occupational cost of Spectral Clustering?

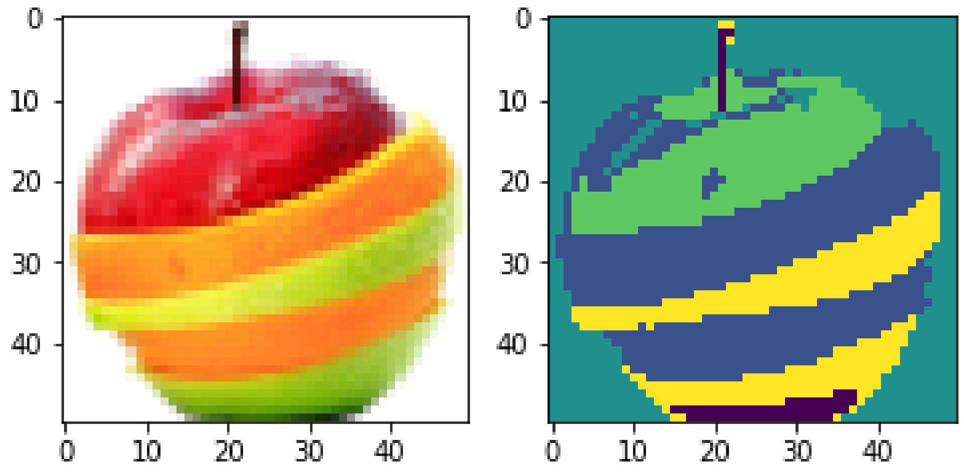
We can start by using k-nn and graphs (or sparse graphs in general) because they are computationnaly fast giving the fact that each node connects to only a few nodes. We can also perfom a downsampling on the image, for example using max pooling method to have a smaller dimension.

3.3. Did you use eig or eigs to extract the final eigenvectors? Shortly, what is the difference between the two? How do they scale to large graphs (order of complexity)? I used eig to extract the eigenvectors in python because we know that the matrix is symmetric so the algorithm is faster. Eigh can be used for a general matrix (slower algorithm)

```
In [5]: %matplotlib inline
        image_segmentation("four_elements.bmp")
```



```
In [6]: image_segmentation("fruit_salad.bmp")
```



2 Name: Dadoun Hind