Graphs in Machine Learning
Spectral clustering

Mohammed Amine Kheldouni

6 November 2018

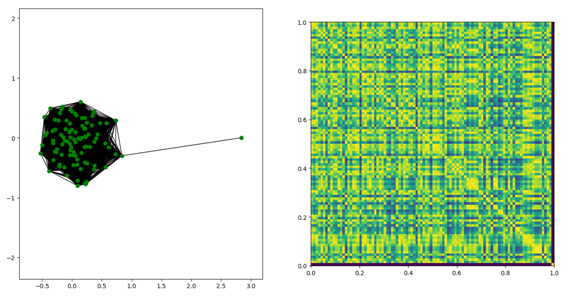
1 Graph construction

(1.1) The purpose of the option parameter gen_{param} in *worst_case_blob* is to generate a point with an *offset* which impacts the first component of the point and keeps it away (for high values of gen_{param}) or closer (for low or negative values of gen_{param}) to the other nodes of the graph.

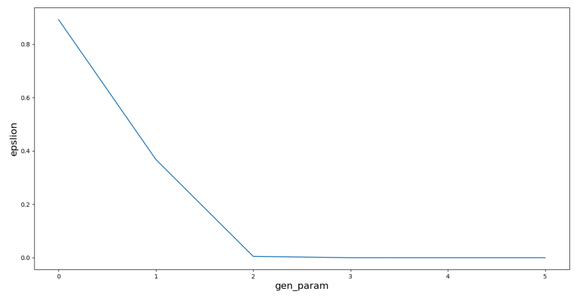
(1.2) This parameter helps us test the function *how_to_choose_epsilon* and understand the process. Indeed, increasing the gen_{param} parameter, the threshold ϵ of an ϵ-Graph decreases in order to find the maximal ϵ threshold which keeps the graph fully connected. Conversely, if gen_{param} decreases, the last node is closer to the others and allows ϵ to get bigger (being more restrictive) and still construct a fully connected graph. Therefore, if the parameter is very large, it will be very difficult for the graph to get fully connected. Hence, the threshold put on the similarity matrix will be very low ($\epsilon \to 0$).

gen_param	5	4	3	2	1	0
threshold ϵ	2.61×10^{-12}	8.9×10^{-9}	1.23×10^{-4}	4.9×10^{-3}	0.35	0.93

Table 1 – Analyzing the sensitivity of the optimal ϵ for a fully connected graph w.r.t. the parameter

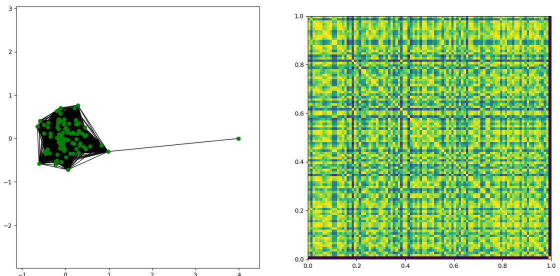


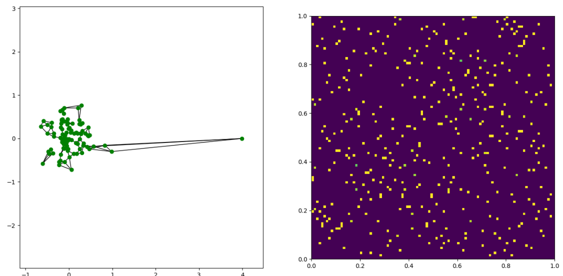
An example of ϵ-Graph with a parameter $gen\_param = 2$

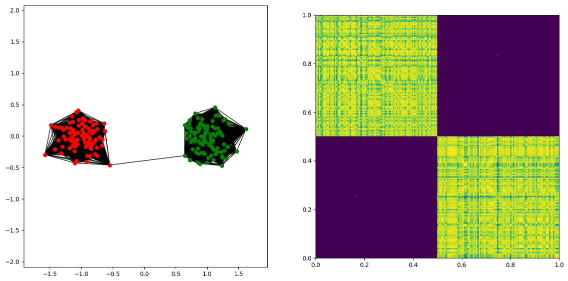


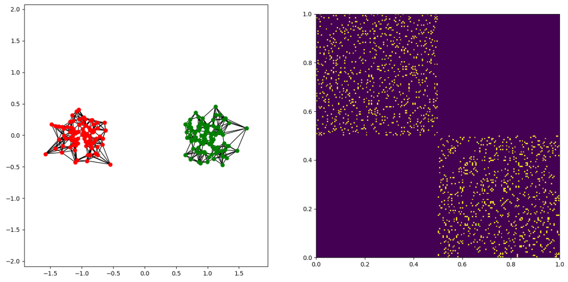
Dependence in gen_param of the highest threshold ϵ such that the graph is fully connected

(1.3) Plotting the similarity graphs with ϵ-Graph and $k-NN$ Graph methods, we see that it is easier to build a k-NN connected graph when the data is compact (a small k manages to link all the nodes). Conversely, the ϵ connected graph works better when the data is separable. Indeed, with the separability property, the ϵ optimization will manage to fully connect the graph whereas the kNN will fail to connect the two separated entities (blobs for example). We display below two cases (with $N_{sample} = 200$) where the $k-NN$ graph works better (worst_case_blob scenario) and another where the ϵ graph is more efficient.


ϵ graph in worst case blob with a parameter $gen\_param = 3$

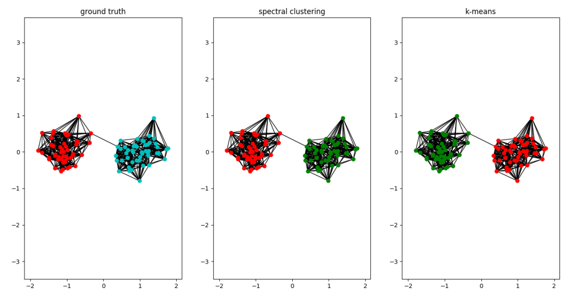

$k-NN$ graph in worst case blob with a parameter $gen\_param = 3$ and manages to build a sparse connected similarity matrix with $k = 3$


ϵ graph in two blobs of variance 0.2 where building multiple edges in each blob is necessary to get the graph fully connected


$k-NN$ graph in two blobs of variance 0.2 where the graph fails to connect the blobs when $k = 7$ (The graph gets connected only when $k \geq 100$

2 Spectral Clustering

(2.1) We generate two connected and compact blobs with $k = 16$ and $\sigma^2 = 0.5$.Using the unnormalized Laplacian matrix of the graph, we can recover the eigenvectors corresponding to the smallest eigenvalues of this matrix. Taking the two smallest eigenvalues into account (the eigenvectors associated to the 0 eigenvalue and the first non-zero eigenvalue). We then use the k-means algorithm considering $y_i \in \mathbb{R}^2$ (here $k_{eigenvalues} = 2$) to cluster those points into clusters denoted $C_1, ..., C_k$. Note that we could have used to *Relaxing Balanced Cuts* seen in the lecture by computing the sign of the spectral coordinate.



$k-NN$ graph clustering in two blobs ($k = 16$)

(2.2) Decreasing the number of considered nearest neighbors in the graph to $k = 10$ makes the graph separated into two connected blobs. In this case, the two eigenvectors we are considering are both associated to zero. This is simpler than the connected case because in this case each eigenvector becomes an indicator for its cluster. Running again the k-means algorithm on the rows of the two first columns of the eigenvector matrix, we assign the resulting labels to our original sample and display the clustering.

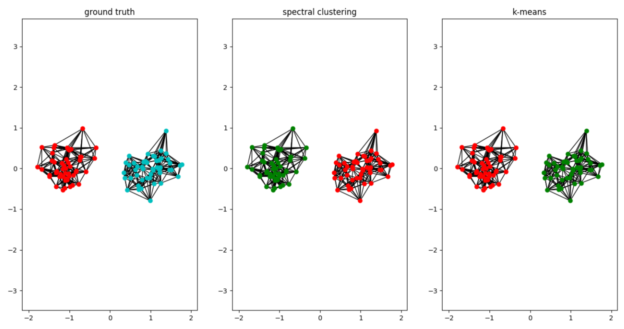


FIGURE 1 – $k - NN$ graph clustering two separated blobs ($k = 10$)

(2.3) In this task, we generate a dataset with 4 blobs of variance $\sigma^2 = 0.03$. We use the elbow rule to detect $k_{eigenvalues}$, the number of eigenvalues we should take into account in our adaptative clustering model. Therefore, our optimal number of eigenvalues is given by the first big gap between two consecutive eigenvalues. Implementing this heuristic function, we can choose the eigenvectors for our clustering and display the results.

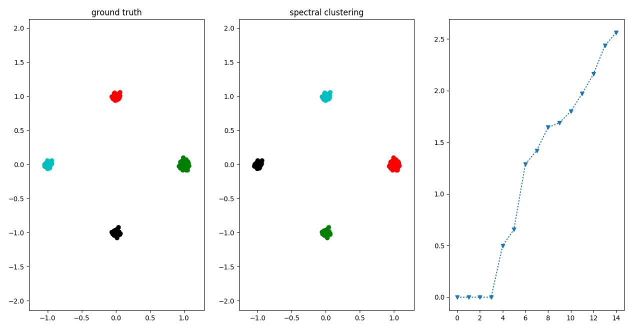


FIGURE 2 – $k - NN$ graph clustering 4 blobs ($\sigma^2 = 0.03, k = 10$) and elbow rule

(2.4) Inreasing the variance of the blobs to $\sigma^2 = 0.2$, the four blobs become closer and connected and the gaps between eigenvalues become less highlighted but the adaptative spectral clustering still manages to cluster efficiently the data into the corresponding labels.

(2.5) We used for the clustering the k-means algorithm. Thresholding could work in some situations but k-means genuinely works better in the situations illustrated here.

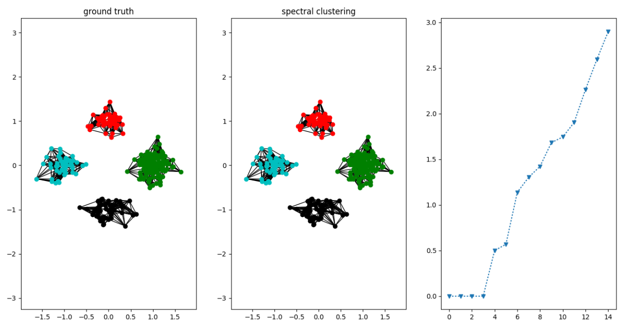


FIGURE 3 – $k - NN$ graph clustering 4 blobs ($\sigma^2 = 0.2, k = 10$) and elbow rule

(2.6) The multiplicity of the eigenvalue $\lambda_0 = 0$ tells us how many connected components we have in our model. Therefore, we can validate our graph construction with a sanity check, since if we really intend to clusterize the data into k groups, we need to have k eigenvectors associated to λ_0 in the graphs Laplacian spectrum.
Moreover, looking at the distribution of eigenvalues provides us with an idea of whether our clusters are well defined or not. Indeed, the larger the gap of the bend is, the better our clusters will be well defined.

(2.7) The k-Means algorithm fails to label correctly the two moons clusters. The centroids toward which the algorithm converges get stable without learning the structure of the given data. Conversely, the spectral clustering using a $k - NN$ graph's Laplacian provides us with an accurate clusterization of the two moons. This difference is inherent to the connectivity versus compactness issue. The k-Means fails in data which require more connectivity than compactness and gets great results when the data are compact.

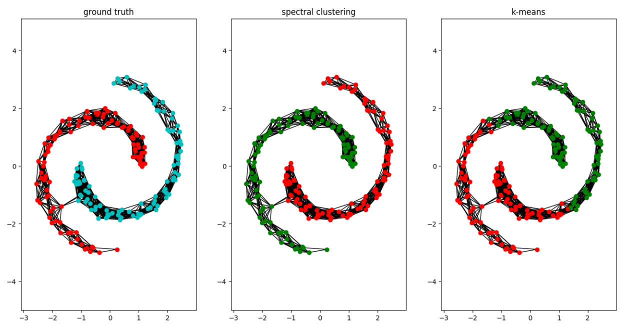


FIGURE 4 – Clustering two moons data with a $k - NN$ graph ($k = 5$) and k-Means

(2.8) In this question, we compare spectral clustering using the unnormalized Laplacian matrix and the random-walk Laplacian matrix (L and L_{rw}) of the point and circle dataset. For this dataset structure, we notice that the unnormalized Laplacian fails to cluster for the first time into two coherent clusters.
This difference of performance is explainable by the fact that the random-walk model takes more into account the connectivity of the graph by normalizing and having more homoneneous clusters. The resulting clusterization of the random-walk Laplacian gives a satisfying and robust result. More generally, if

the graph is very regular and most vertices have approximately the same degree, then all the Laplacians are very similar to each other, and will work equally well for clustering. However, if the degrees in the graph are very broadly distributed, then the Laplacians differ considerably.

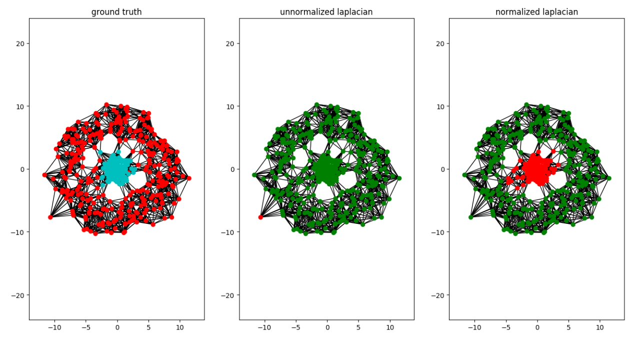


FIGURE 5 – Clustering point and circle data with a $k - NN$ graph ($k = 15$), comparing unnormalized and random-walk Laplacians.

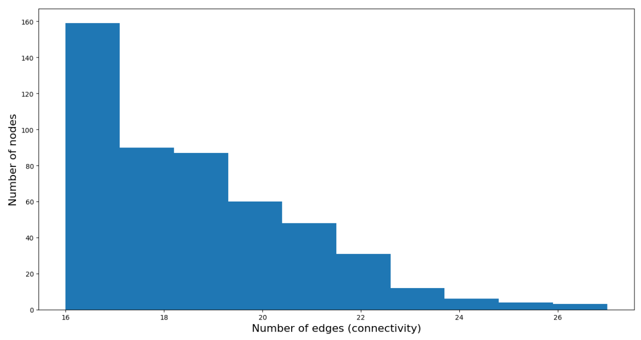
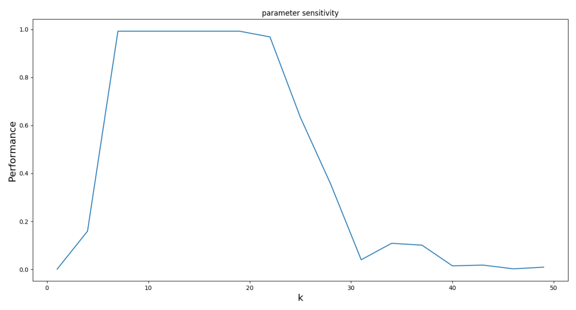
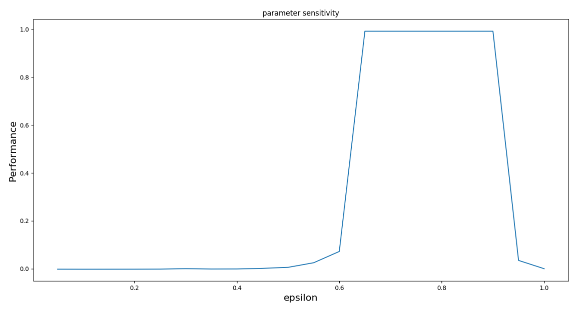


FIGURE 6 – Histogram of nodes' degree distribution for a point and circle dataset ($k = 15$).

(2.9) In the following figures, we display the performance (the ARI index) of the spectral clustering while varying one of the parameters of our constructed graphs (ϵ or k). The results prove that there exists a region of stability in which the clustering works efficiently but still performs poorly outside of this region. Therefore, the spectral clustering is highly dependent on the parameters we provide for the graph's connectivity and the eigenvalues selection.



ARI index of performance in clustering data with a $k - NN$ graph depending on k



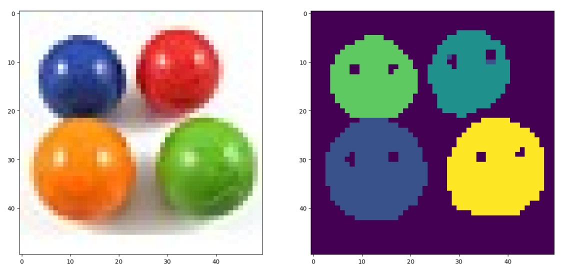
ARI index of performance in clustering data with an ϵ graph depending on ϵ

(2.10) If the ground truth wasn't provided, we could try to evaluate the quality of our clustering by evaluating its stability over multiple executions or look for other unsupervised performance indices to evaluate the accuracy of the clustering. One idea would be to look at the silhouette value which is a measure of how similar an object is to its own cluster (cohesion) compared to other clusters (separation). This provides a method to validate the consistency within the clusters.

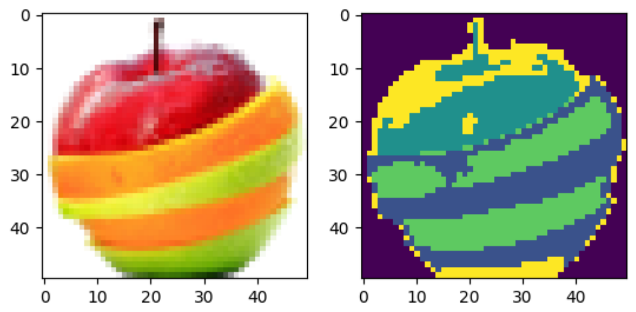
3 Image Segmentation

(3.1) This third section provides us with an application of spectral clustering in computer vision and image segmentation. We build a $k-NN$ graph for each of the given input images ($four\_elements$ and $fruit\_salad$). As explained in the documentation, the images are 50×50 pixels RGB images. Therefore, we transform the data into a X_r matrix in $\mathbb{R}^{2500\times3}$.

Considering the $four\_elements$ image, we count 5 different colors in the image (4 differently colored balls and the background color). Therefore, we construct a $k-NN$ graph (with $k = 40$) and apply the adaptative spectral clustering on the image, asking for 5 different labels. The results are very satisfying (increasing k helped in the clusterization clearness). Considering now the second input image, we construct similarly the spectral clustering with $k = 15$ and $num_{classes} = 5$ (corresponding to white, red, dark red, orange and green colors).



Spectral clustering of 4 balls in an image using $k-NN$ graph's Laplacian



Spectral clustering of a fruit salad in an image using $k-NN$ graph's Laplacian

(3.2) There exists many tricks related to computer vision and image proccessing to blur the image or apply a transformation to its pixels to reduce the picture's dimensionality and hence the number of nodes in the constructed graph. Furthermore, we can think of other spectral clustering tricks such as Fast Approximate Spectral Clustering and Fast kernel spectral clustering methods. Otherwise, we can think of the potential sparsity of the matrix. Indeed, considering the $k-NN$ graph we know that our matrix will have many zeros when (i, j) are not in the k nearest neighbors of each other. Therefore, we can think of an other way of stocking the data and proccessing it instead of building a $N \times N$ matrix which slows down the spectral clustering's computation.

(3.3) During our implementation of spectral clustering, we used the *eig* function and not the *eigs* one. The first one computes all the spectrum of our Laplacian matrix, computing the N eigenvectors and their respective eigenvalues, whereas *eigs* only computes a number of those eigenvalues and eigenvectors based on a parameter given by the user. In practice, for large graphs, it is useless to compute all the eigenvalues and *eigs* should be prefered since it reduces the complexity by shrinking the dimensionality to the $P < N$ first eigenvectors we chose for building our clusters.

4 Conclusion

In this work, we learnt graph construction properties and differences between $k - NN$ and ϵ graphs. We managed to cluster successfully a great panoply of data structures (blobs, two moons, point and circle, ...) using the Laplacian matrix and symmetrization in some cases. We also compared the spectral clustering algorithm with the K-Means and analyzed the properties of spectral clustering with respect to the Laplacian's spectrum. Finally, good results have been presented in the third section, showing the importance of spectral clustering and its multiple applications, for instance in computer vision and image processing.