Evaluation Of Spectral Clustering methods.

Malik Hacini

May 20, 2024

Contents

1 Introduction

This document presents the methods, experiments and results achieved for spectral clustering on synthetic and real world datasets. All of the project was written in Python. You will find the code referenced in this presentation in the ”src” folder of the project. For more information on spectral clustering, refer to PH.

2 Code architecture

The code for this project is divided into 2 main Python files :

- graphs.py
- spectralclustering.py

The goal of the structure is to be versatile: to perform a new experiment, you do not need to modify the code, as the different parameters give a lot of flexibility.

2.1 graphs.py

This file contains everything needed to the construction of a graph from a dataset. There is a "Graph" Python class meant to create a graph based on a list of data points. With class attributes and methods, you can then access all of the spectral clustering related objects of the graph : adjacency matrix and laplacians (with all kinds implemented).

One key feature of these graphs is the similarity function used. A "Similarity" Python class exists for this purpose. However, in most cases , we use the gaussian kernel similarity function, hence it is the default one in our implementation.

Lastly, there is a symmetrizing method parameter. It is the method used for symmetrizing the adjacency matrix in the case of classical spectral clustering. The choices are "mean, or, and". Please refer to the source code for more details. Overall, this file is all you need to construct a graph based on your dataset.

Exemple. Consider a dataset "data" where we want to build a 10-nn graph G using the gaussian kernel of standard deviation $\frac{1}{2}$ and symmetrizing the adjacency matrix by the 'mean' method, the call would be :

```
1               G=Graph(data,10,'knn','mean',1/2)
2
```

We can then easily access the graph adjacency matrix and laplacians using the Graph class attributes and methods.

2.2 spectralclustering.py

This file contains everything needed to perform spectral clustering on a dataset, it's main function being simply called "spectral clustering". Understanding this functions's parameters is the only real thing you need to conduct experiments using spectral clustering. Here is the docstring associated with it :

```
1       def spectral_clustering(data,k_neighbors,n_eig,laplacian,g_method='
    g_knn',sym_method='mean',sigma=None,use_minibatch=False,eigen_only=False
    ,clusters_fixed=False,return_matrix=False,labels_given=np.array([None]))
    :
2           """Performs spectral clustering on a dataset of n-dimensional
    points.
3           Inputs :
4           data (ndarray): The dataset, a
```

```
5         k_neighbors (int): Number of neighbors you want to connect in the
      case of a k-nn graph.
6         n_eig (int): Number of eigenvectors to calculate. Used to compute
      the number of clusters
7         laplacian (string) : The laplacian to use between [un_norm , sym ,
      rw]
8         sym_method (string): The method used to symmetrize the graph matrix
       in the case of an asymmetric adjacency matrix.
9         sigma (float): Standard deviation for the gaussian kernel.
10        use_minibatch (bool) : Choice of the k-means algorithm. True might
      lead to better performance on large datasets. Default = False.
11        eigen_only (bool) : If True, the function will only returns the
      eigenvalues and eigenvectors and not compute the full clustering.
      Default = False.
12        clusters_fixed (int) : The number of clusters in your data. If
      unknown, leave by default and the eigengap heuristic will be used.
      Default = False.
13        return_matrix (bool) : True <=> returns the adjacency matrix
      alongside the clustering results. Use if you want to visualize the graph
      . Default = False.
14        labels_given (ndarray) : The correct labels of your data. If given,
       used to reorder the labels obtained by clustering. Leave empty if
      unknown. Default = False
15
16        Returns :
17        vals (ndarray): the computed eigenvalues pf the graph laplacian.
18        labels (ndarray) : labels of the points after spectral clustering,
      ordered in the same way as the dataset.
19        matrix (ndarray) : the adjacency matrix of the graph
20        """
21
```

3 Spectral Clustering Experiments

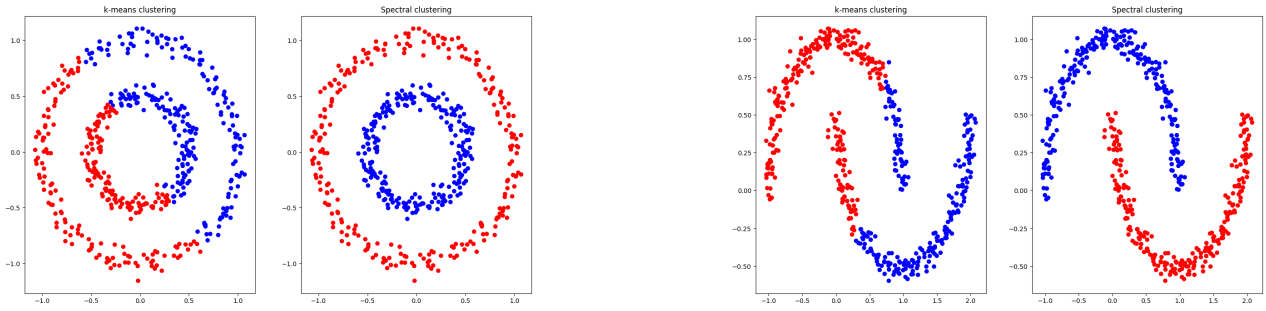
This section summarizes the experiments done.

3.1 Basic Experiments

To assert the good behavior of our algorithms, the first step was to conduct very basic experiments. We are going to use very simple toy datasets and focus on 2 important results : Clustering performance and Eigenvalues of laplacians.

3.1.1 Circles and Moons

The most basic spectral clustering example are the circles and moons dataset. These datasets fail to be clustered correctly by k-means, as the clusters do not lie in disjoint convex sets. However, spectral clustering is powerful enough to correctly cluster these datasets. Due to the simplicity of the datasets, practically any choice of settings gives perfect performance.



(a) Circles (b) Moons

Figure 1: Comparison between k-means and spectral clustering on toy datasets

In this experiment, we know the correct number of clusters because we can visualize the dataset, which isn't always the case for real datasets. To get a hint on the correct number of clusters, sectral clustering offers the eigengap heuristic. For most datasets, the quality of the eigengap will be different depending on the laplacian. However, in very simple cases like circles and moons, the k-nn graph is disconnected into the two clusters, which is the ideal theoretical case for the eigengap heuristic, thus every laplacian leads to a good eigengap :

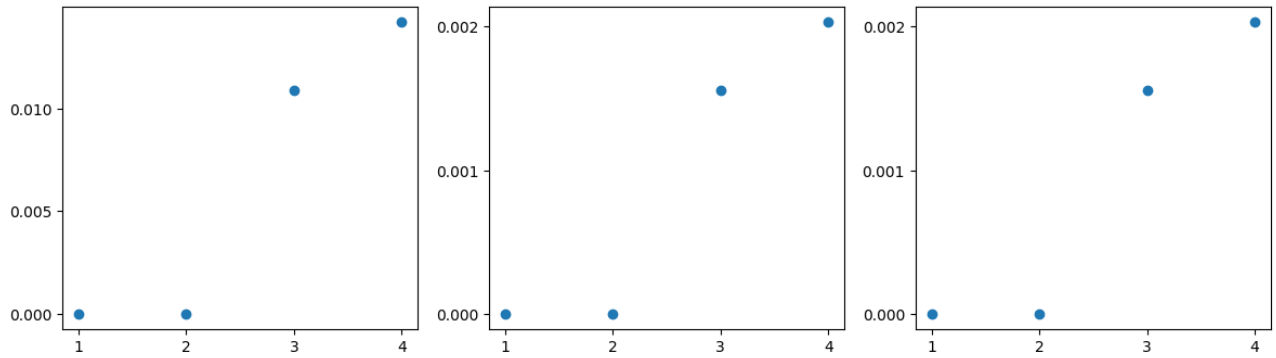


Figure 2: First eigenvalues of laplacians for the clustering of the circles dataset.
Left to right : L, L_{sym}, L_{rw}

Every laplacian has a gap in it's eigenvalues after 2 of them, meaning the graph contains exactly 2 connected components, thus 2 clusters. In order to show the difference between each laplacian, we need more sophisticated datasets.

3.1.2 Gaussian Mixture Models (GMMs)

In the goal of doing qualitative comparison between different methods, using 2-dimensional gauussian mixture datasets is a good starting point.

Multivariate Normal Distribution The multivariate normal distribution, also called "multivariate gaussian" is a generalization of the one-dimensional (univariate) normal distribution to higher dimensions. For any dimension d, it can be defined by it's PDF :

$$\begin{array}{ccc} \mathbb{R}^d & \longrightarrow & \mathbb{R}_{>0} \\ x & \longmapsto & \frac{1}{\sqrt{(2\pi)^2|\Sigma|}} e^{-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^t} \end{array}$$

Where:

- $\mu \in R^d$ is the mean of the distribution.
- Σ is a d by d square symmetric positive semi-definite matrix called the covariance matrix, because when x is treated as a random vector with each of it's coordinates x_i being a random variable, $\sigma_{i,j} = Cov(x_i, x_j)$.

The multivariate gaussian distribution is centered at μ and has an ellipsoidal shape defined by Σ.

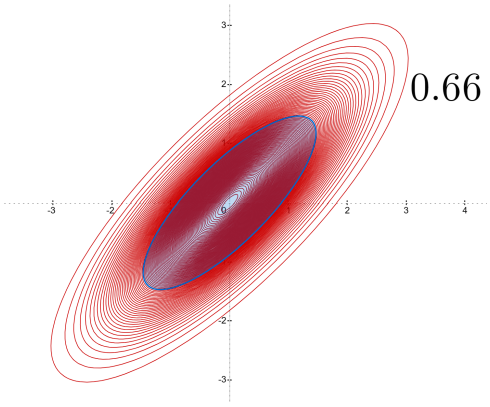


Figure 3: Level sets of a bivariate normal distribution centered at the origin. The probability of a sample landing in the blue ellipse is 0.66.

A linear combination of d-dimensional gaussians is called a gaussian mixture.

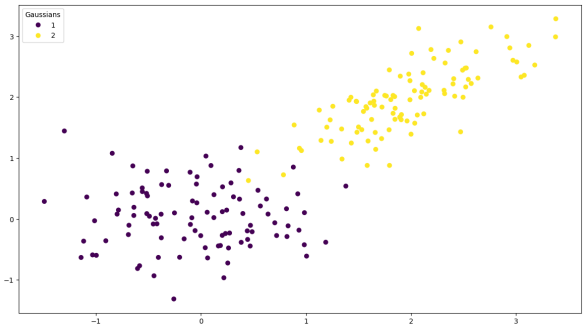
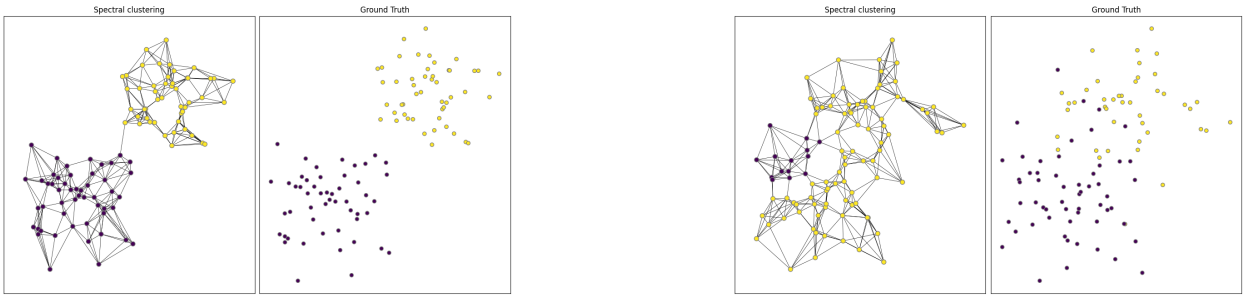


Figure 4: 200 samples from a mixture of two 2-dimensional gaussians.

Remark. In practice, since gaussians PDF decay quickly when driving away from μ, when the means are fairly spaced, the different gaussians do not really interact to form a diffrent PDF and the resulting mixture just looks like a superpostion of the respective gaussians. Thus, we do not perform the sampling with the actual PDF of the mixture, but uniformly choose between the different gaussians for each point.

Gaussian mixtures form great toy datasets on which we can test and visualize our clustering, where a goal cluster is all of the points sampled from a particular gaussian.



(a) Successful clustering (b) Unsuccessful clustering

Figure 5: Visualization of spectral clustering and it's k-nn graph on a gaussian mixture. Settings : L_{rw}, $k = 6$, $\sigma = \frac{1}{3}$ (for gaussian kernel).

On figure (a) we see that the k-nn graph well encodes the cluster structure of the data. It is fully connected but there is only one link between clusters, which makes for great performance of SC using L_{rw}.

On figure (b), the means of the two gaussians are closer apart, thus the clustering fails to partition the graph correctly due to the multiple links between the real clusters. However, increasing k creates stronger links inside of the clusters, leading to better performance :

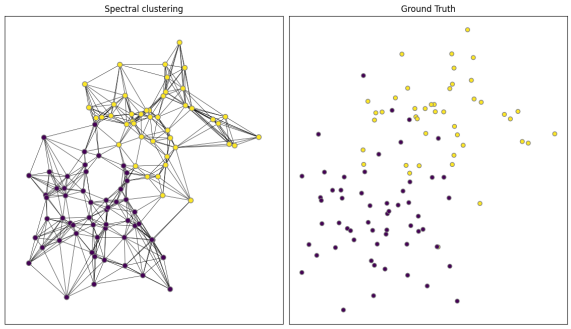
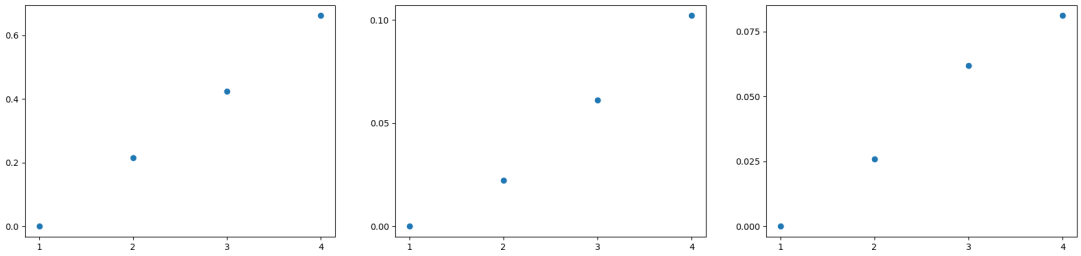


Figure 6: The performance of the clustering is increased with $k = 8$

By forcing the algorithm to form 2 clusters, we get good performance. Again, what if we didn't know the true number of clusters ? In this case, the graph is fully connected, so does the eigengap really indicates 2 clusters ?



(a) L (b) L_{sym} (c) L_{rw}

Figure 7: First eigenvalues of laplacians for the graph of **??**.

These results clearly show that the connectivity of the graph has great impact of the quality of the eigengap. Both the spectra of L and L_{rw} give no valuable information and while L_{sym} does have a "gap" after the second eigenvalue, it isn't clear enough to conclude on the true number of clusters. In conclusion, while the eigengap can be a useful criteria in the case of well separated datasets, when the graph is strongly connected to better the performance on harder datasets, the eigengap becomes meaningles Thus, we conclude that to efficiently choose k, we need to connect the graph enough to ensure good performance, but not too much to the point where the eigengap is useless, otherwise we will be unable to know the correct number of clusters. In practice, this is near impossible most of the time.

3.2 UCI datasets

Now that we know our algorithms follow the basic theoretical behaviors we want, we can benchmark them on more interesting datasets. We used the same 11 benchmark datasets from the [1, UCI ML repository] as [2, Jonckheere et.al]. These datasets are high dimensional and to correctly assert performance of the clustering, we need a pertinent quantitative metric.

3.2.1 Normalized Mutual Information (NMI)

The NMI facilitates comparisons between two different ways of partitioning a dataset, yielding a value that ranges from 0 to 1. A higher value indicates a greater degree of similarity between partitions. As an external metric, the NMI necessitates the availability of class labels for computations, implying that the ground truth is required when employing this metric. The calculation of the NMI between two partitions A and B is executed according to the following equation :

$$NMI\,(A,B) = \frac{2 * I(A,B)}{[H\,(A) + H(B)]}$$

Where $I(A,B)$ is the mutual information and H the entropy. In practice, A is the set of predicted labels and B the ground truth labels of the dataset.
NMI can fail to correctly assess the performance of the clustering when the number of clusters predicted and the ground truth number of clusters do not match. However, this won't be the case in our experiments, since we will force the clustering to build the right number of clusters. Other evaluation metrics such as the adjusted Rand Index (ARI) exist, but we limited ourselves to NMI for these experiments as it already is very well suited for comparison purposes.

3.3 Method and results

We compare the 3 standard graph laplacians : L, L_{sym} and L_{rw}. We construct,via a gaussian kernel similariy function, a k-nn graph using the optimal connectivity parameter k computed by [2, Jonckheere et.al] for each clustering. In the following results table, N is the number of samples and d their dimension. We report the results using NMI.

Table 1: Clustering performance (NMI) on UCI datasets with optimal parameters in brackets.

Dataset	N	d	k	L	L_{sym}	L_{rw}
Iris	150	3	4	80.58	80.58	74.98
Glass	214	9	6	38.59	38.92	38.95
Wine	178	13	3	86.33	86.33	83.66
WBDC	569	30	2	67.73	69.47	68.54
Control chart	600	60	6	81.17	81.17	82.94
Parkinson	185	22	2	21.96	19.13	28.89
Vertebral	310	6	3	39.26	39.26	52.06
Breast Tissue	106	9	6	54.03	54.43	54.04
Seeds	210	7	3	73.90	73.90	76.29
Image Seg.	2310	19	7	67.06	67.41	67.42
Yeast	1484	8	10	30.58	31.11	31.37
Average	–	–	–	58.29	58.34	59.92

References

[1] Dheeru, D. and Karra Taniskidou.UCI repository of machine learning databases *University of California, Irvine, School of Information and Computer Sciences*, 2017.

[2] Harry Sevi, Matthieu Jonckheere, and Argyris Kalogeratos. Generalized spectral clustering for directed and undirected graphs, 2022.