Homework 4: Graph Spectra

Mauro Pungo, Viktoria Sartor

December 2021

1 Introduction

This report briefly presents the results obtained using the algorithm presented on the paper ***"On spectral clustering: Analysis and an algorithm"*** by A. Ng, M. Jordan and Y. Weiss [1].

The paper discusses an algorithm to partition a graph into K communities using a spectral approach, ie, the study of the eigenvalues and eigenvectors from matrices derived from the graph's edges structure.

2 The algorithm

The paper presents a spectral clustering algorithm more stable that the "out-of-the-box" methods such as KMeans and other classical spectral clustering approaches.

Given a set of points $S = \{s_1, \ldots, s_n\}$ in $\mathbb{R}^l$ that we want to cluster into k subsets:

1. Form the affinity matrix $A \in \mathbb{R}^{n \times n}$ defined by $A_{ij} = \exp(-||s_i - s_j||^2/2\sigma^2)$ if $i \neq j$, and $A_{ii} = 0$.
2. Define D to be the diagonal matrix whose (i,i)-element is the sum of A's i-th row, and construct the matrix $L = D^{-1/2}AD^{-1/2}$.[1]
3. Find $x_1, x_2, \ldots, x_k$, the k largest eigenvectors of L (chosen to be orthogonal to each other in the case of repeated eigenvalues), and form the matrix $X = [x_1 x_2 \ldots x_k] \in \mathbb{R}^{n \times k}$ by stacking the eigenvectors in columns.
4. Form the matrix Y from X by renormalizing each of X's rows to have unit length (i.e. $Y_{ij} = X_{ij}/(\sum_j X_{ij}^2)^{1/2}$).
5. Treating each row of Y as a point in $\mathbb{R}^k$, cluster them into k clusters via K-means or any other algorithm (that attempts to minimize distortion).
6. Finally, assign the original point s_i to cluster j if and only if row i of the matrix Y was assigned to cluster j.

Figure 1: The algorithm, as presented in the paper

The algorithm consists in 3 steps made of substeps: Pre-processing, Decomposition and Grouping.

Let $\mathcal{G}(V, E)$ be a graph, V the set of its nodes/vertices and E the set of edges such that E = {(i,j) | an edge between node i and j exists }

2.1 Pré-processing

In this step, we build **A**, a matrix representation for $\mathcal{G}$. $A \in \mathbb{R}^{nxn}$ is the Affinity matrix. It captures similarities between nodes on the graph. In this algorithm this similarity depends on the distance between the points (here represented as nodes).

Step 1 of figure 1 defines the affinity between nodes i and j as $A_{ij} = e^{-\frac{||s_i - s_j||^2}{2\sigma^2}}$. For unweigthed graphs, it becomes clear that the affinity matrix is in fact the adjacency matrix where $A_{ij} = 1$ if an edge exists between nodes i and j. If the graph is also undirected then $A_{ij} = A_{ji}$.

From A we can build the diagonal matrix $D \in \mathbb{R}^{nxn}$ of degrees where $D_{ii} = \sum_{j|(i,j)\in E} A_{ij}$.

2.2 Decomposition

In this step we compute the spectrum ,eigenvalues and eigenvectors, of G.
But instead of computing the spectrum of A, we compute that of the graph's Laplacian L. A common definition for the Laplacian is $L = D - A$ but in this algorithm, it's defined (step 2) as $L = D^{\frac{1}{2}} A D^{\frac{1}{2}}$

The reason we compute the spectrum of L is that its eigenvalues reveal global graph structures that are not evident from the edges structure. Namely, the eigenvalues give important information about the graph's connectivity, if the k first eigenvalues of L are 0, then the graph has K connected components. Conversely, if the 2nd smallest eigenvalue , λ_2 is greater than 0, then $\mathcal{G}$ is connected (K = 1 components).

2.3 Grouping

By using the first K eigenvectors, after normalizing them, as points in $\mathbb{R}^K$ we can the run any classical clustering algorithm on this space. We chose to use the default Scipy's implementation of KMeans, in Python. Finally we just replace the i-th eigenvector in $Cluster_j$ by node i.

3 Results

In order to test the performance of the algorithm we used both graph datasets from Canvas. The results presented are for both datasets.

3.1 Graph Sparsity

Below we show the sparsity of A for both datasets.

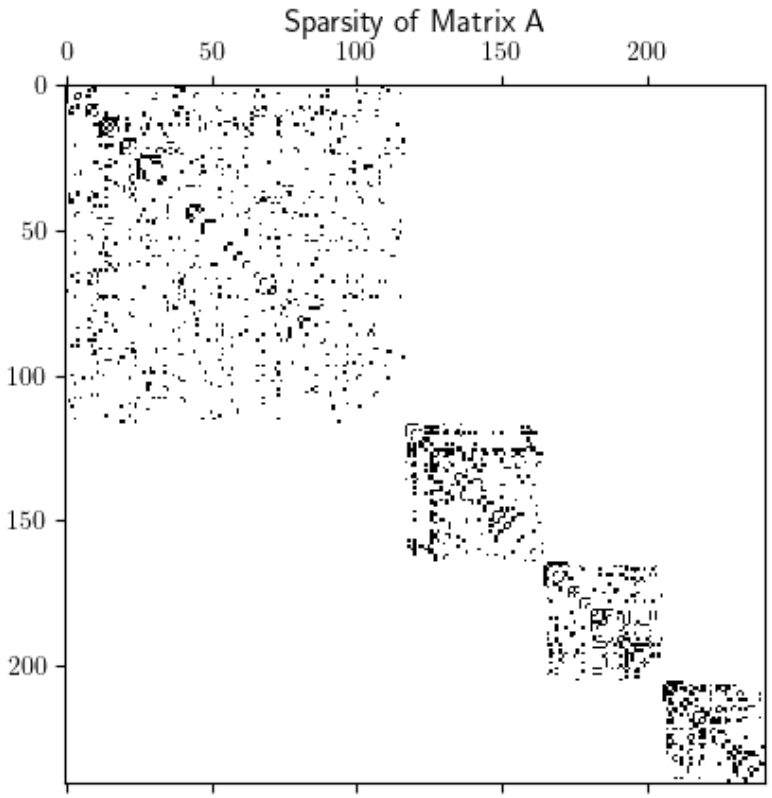

Figure 2: Matrix A's sparsity for dataset example1.dat

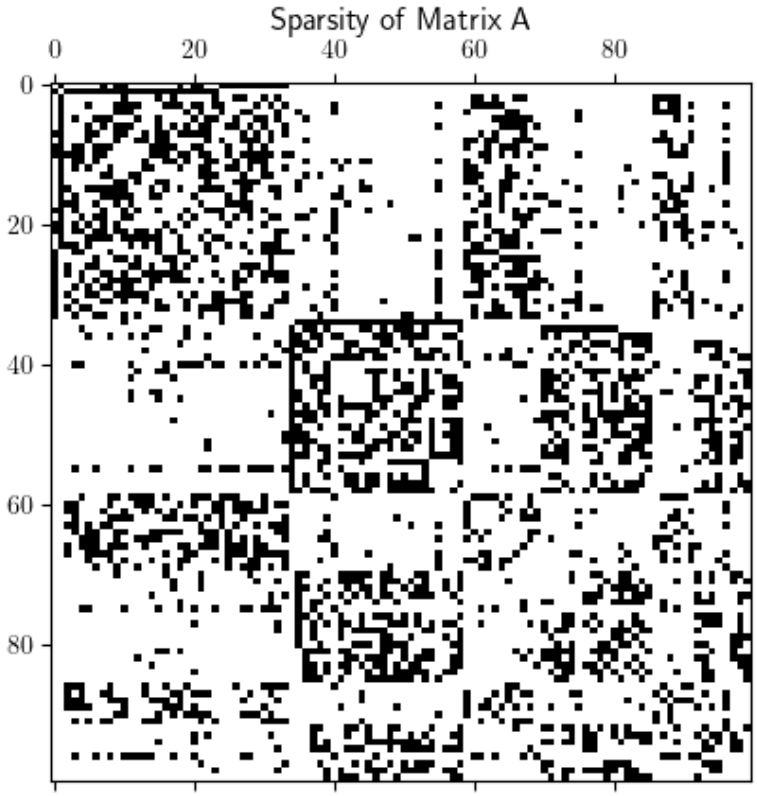

Figure 3: Matrix A's sparsity for dataset example2.dat

3.2 Graph Spectrum

Results see below.

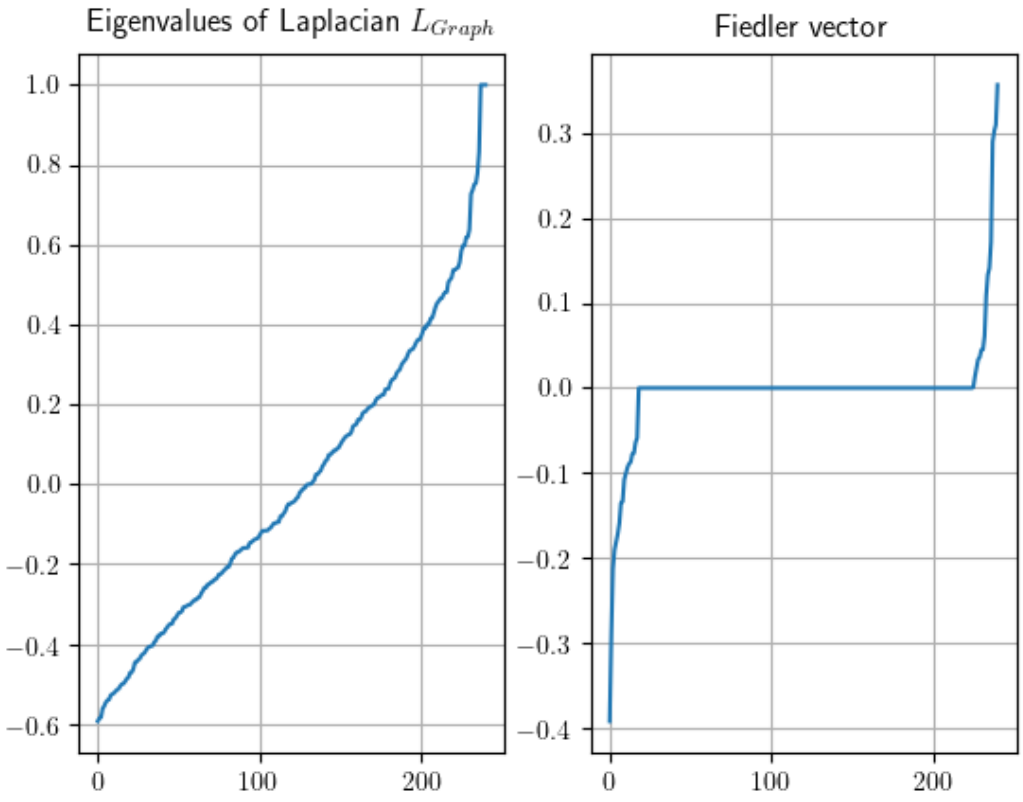

Figure 4: Graph Spectrum for example1.dat dataset

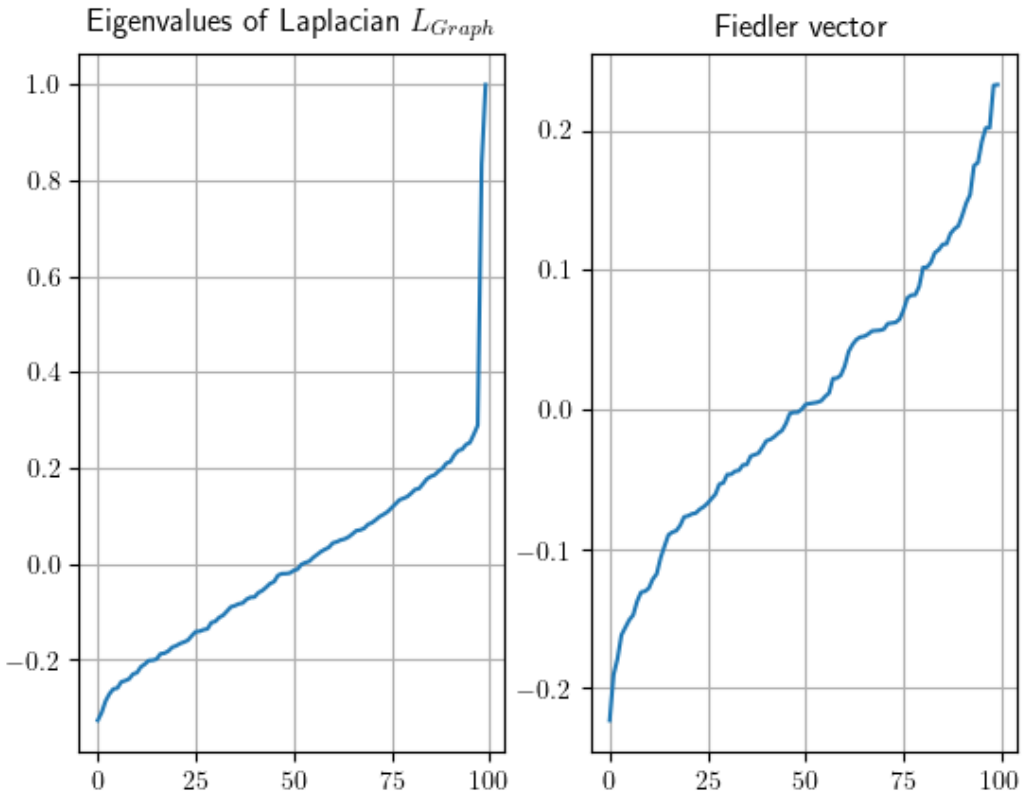

Figure 5: Graph Spectrum for example2.dat dataset

3.3 Graph K-partitions

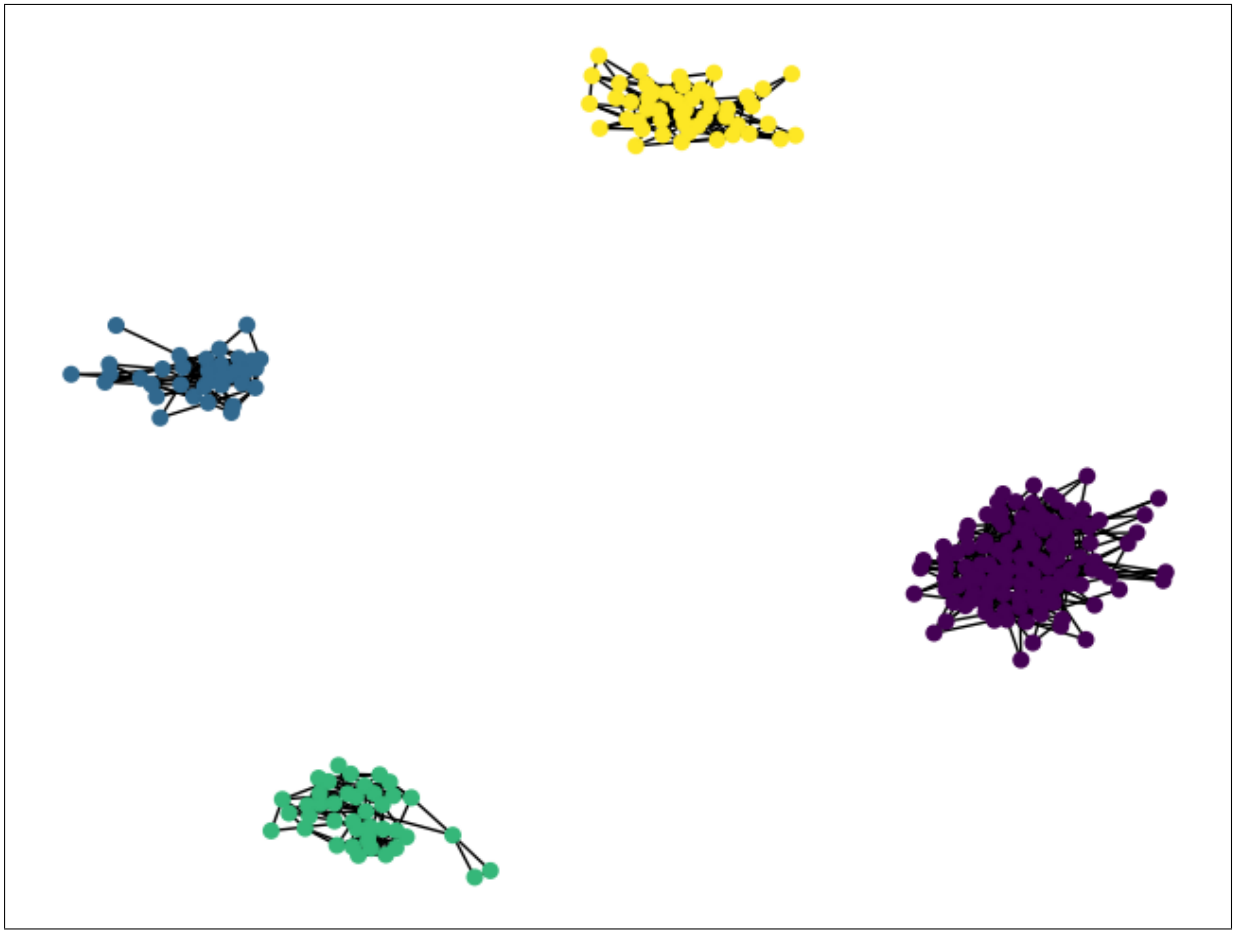

Figure 6: Graph's partitions for dataset example1.dat using K = 4 clusters

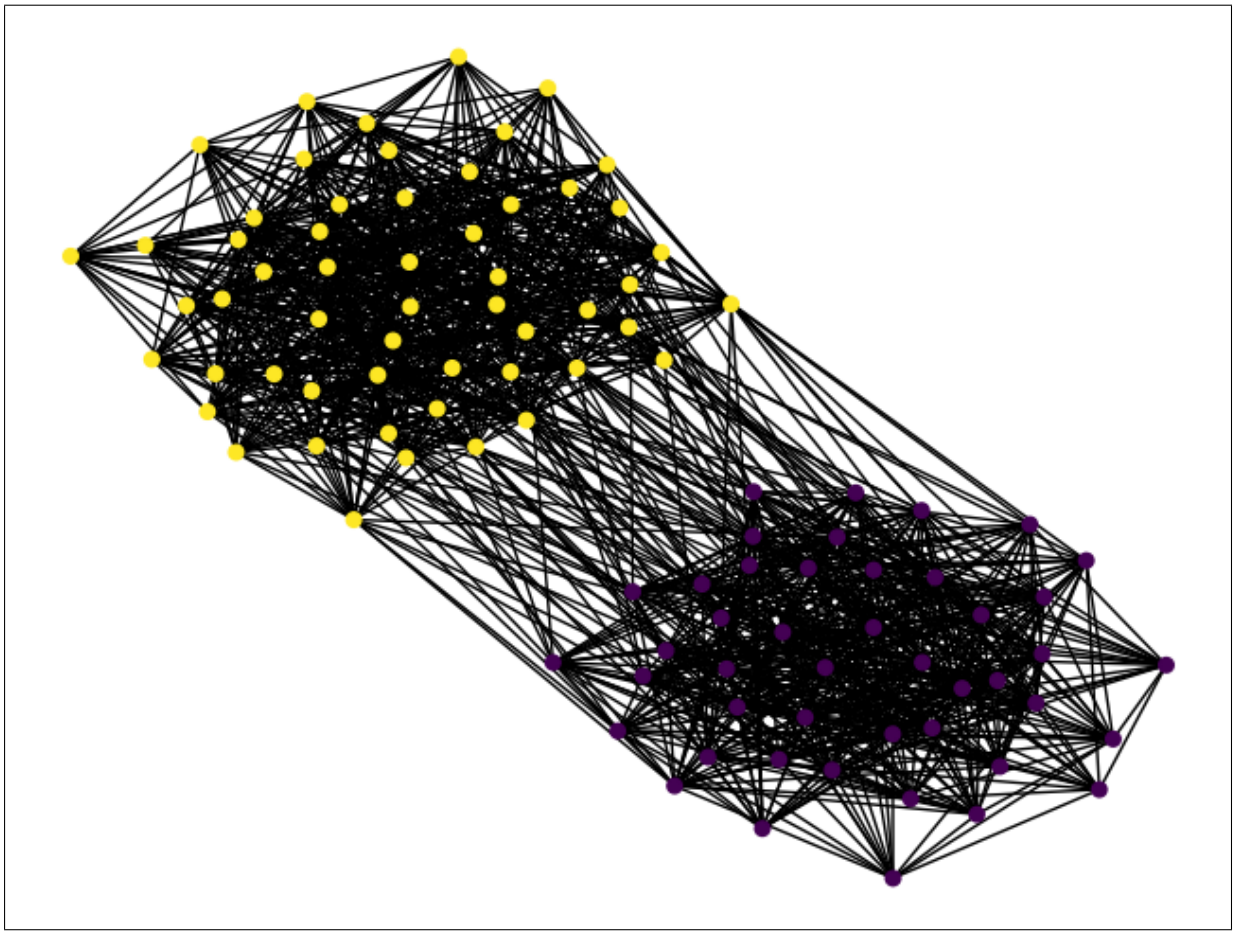

Figure 7: Graph's partitions for dataset example1.dat using K = 2 clusters

4 Build instructions

Syntax: python graph_spectra.py [ARG_FLAG1] [ARG_VALUE1]
For further help just run *python graph_spectra.py -h*
Run script *graph_spectra.py* in the same directory where the data directory is stored.

Argument parser:

```
—dataset choices: 'example1.dat', 'example2.dat'
—partitions default: 4

All flags with double hyphens!
```

References

[1] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. "On Spectral Clustering: Analysis and an algorithm". In: *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*. Ed. by Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani. MIT Press, 2001, pp. 849–856. URL: https://proceedings.neurips.cc/paper/2001/hash/801272ee79cfde7fa5960571fee36b9b-Abstract.html.