DATA MINING - HOMEWORK 4
Graph Spectra

SERGHEI SOCOLOVSCHI, ANGEL IGARETA

28 November 2020

Introduction

This project aims to implement and test the spectral graph clustering algorithm as described in the paper "On Spectral Clustering: Analysis and an algorithm" [1]. Using the implementation of the K-eigenvector algorithm, two sample graphs will be analyzed:

- **Sample Graph 1:** Dataset prepared by Ron Burt [2], where he dug out the 1966 data collected by Coleman, Katz, and Menzel on medical innovation.
- **Sample Graph 2:** A synthetic graph provided for the assignment.

In addition to the basic implementation of the algorithm where k has to be specified, an additional estimation of the cluster size was added, by analyzing the gaps between the eigenvalues. This allows the user to just provide the graph as input and the implementation will select the best estimation of k.

Tools

Two implementations of the algorithm are provided, using both Matlab for simpler implementation and Python in a Jupyter Notebook together with the libraries NumPy, NetworkX, Scipy, and NLTK. In both implementations, the results and steps are equivalent.

Implementation

In this section, the Matlab implementation of the problem will be described. To perform a spectral graph clustering, the following steps were performed:

1. **Load datasets:** Both of them represent a directed graph in the format of a data file (.dat) with two unnamed columns, which represent vertex1 and vertex2 of each of the edges in the graph.
2. **Create a graph from the edge list**. The method '*graph'* was used to load the edge list, the object returned represents undirected graphs, so no preprocessing was needed in Matlab.
3. **Extract adjacency matrix (A) from the graph**. The function '*adjacency'* was used to return a sparse adjacency matrix for the graph, where the row and column indices represent the nodes and the values represent the absence or presence of a connection between the nodes. To transform the matrix to a dense one, the method *full* can be used.
4. **Generating the degree matrix (D)** by summing the value of every row in the adjacency matrix, representing the degree of each node, and storing the result on a diagonal sparse matrix.
5. **Generate Laplacian matrix L** following the expression suggested in the paper:

$$L = D^{-\frac{1}{2}} * A * D^{-\frac{1}{2}}$$

6. **Extract eigenvalues and eigenvectors** from the Laplacian matrix L, using the '*eig'* function.
7. **Estimate k** by finding the index of the eigenvalue that has the biggest eigengap with its predecessor. When an eigenvalue is large, a significant amount of information is represented with its eigenvector. If the next eigenvalue is very different than its predecessor, it means that adding a new cluster would not be significant, as most of the information is represented by the previous eigenvectors. That is the reason to estimate k, it is needed to look for the biggest difference in eigenvalues.
8. **Select the k-largest eigenvalues and their relative eigenvectors**, resulting in the Eigen matrix X.
9. **Normalize the rows of the Eigen matrix X**, using the *'normr'* method.
10. **Apply the k-means clustering algorithm**, passing the normalized matrix and the estimated cluster size K as parameters.
11. **Plot the selected graph** and distinguish nodes from different clusters using the '*highlight'* function.

Execution

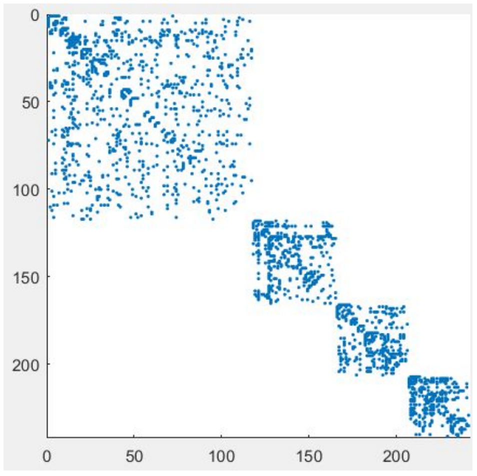
Depending on the desired implementation to run, the instructions are:

- **Matlab**: Adjust the variable *'file'* with the local path of the dataset to be analyzed and execute the code. The results would be three plots: sparsity pattern of adjacency matrix, eigenvalues, and the graph with the estimated clusters. Note that the program will automatically estimate the cluster size.
- **Jupyter Notebook**: To execute the Python implementation, this time the cluster size needs to be specified by changing the variable *'cluster_size'* and the id of the dataset needs to be indicated with the variable *'dataset_id'* (using the sample graph 2 as default).

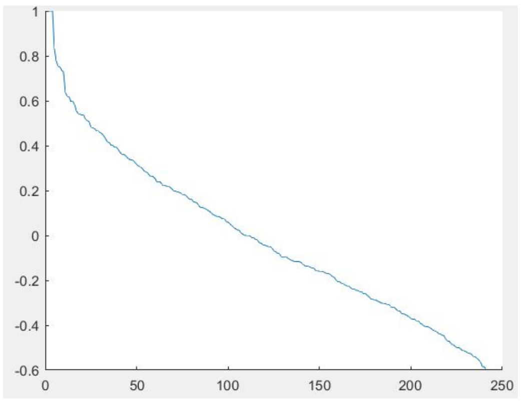
Results

Graph 1

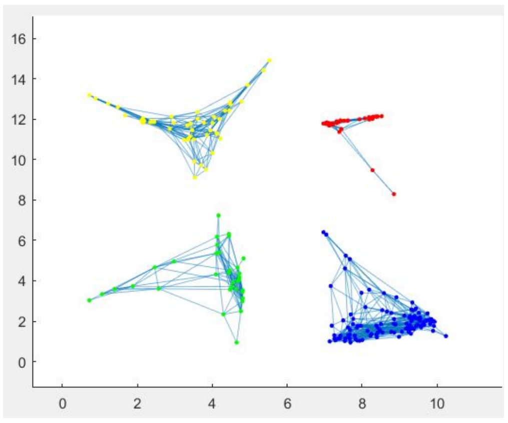
Sparsity Pattern: plotting the adjacency matrix helps to visualize the number of the communities present in the graph. In this case, 4 well-distinguished communities can be seen.



Eigenvalues: plotting all the eigenvalues from the largest to the smallest one, the biggest eigengap can not be observed clearly through the graph but in this case, it would correspond to the difference between the fourth and fifth largest eigenvalues.

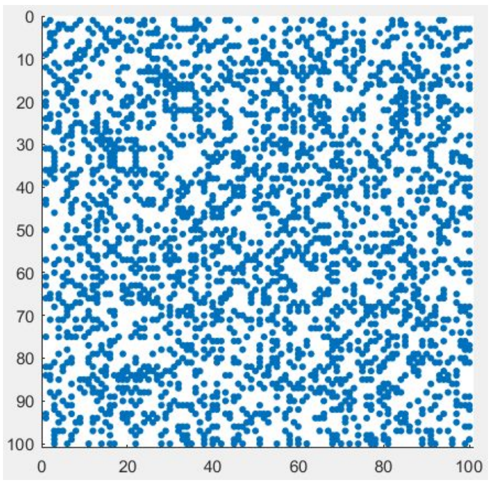


Spectral Clustering: the output given by applying the k-spectral-clustering algorithm to the first dataset. As predicted, the dataset represents **four** isolated communities.

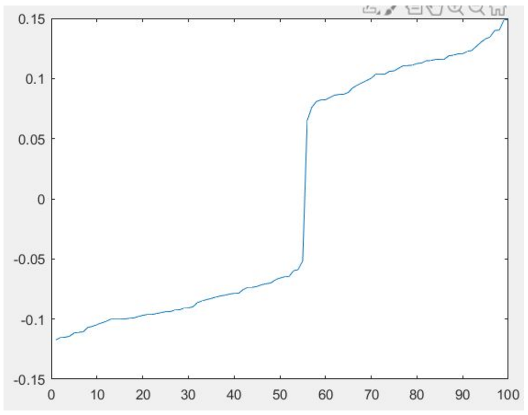


Graph 2

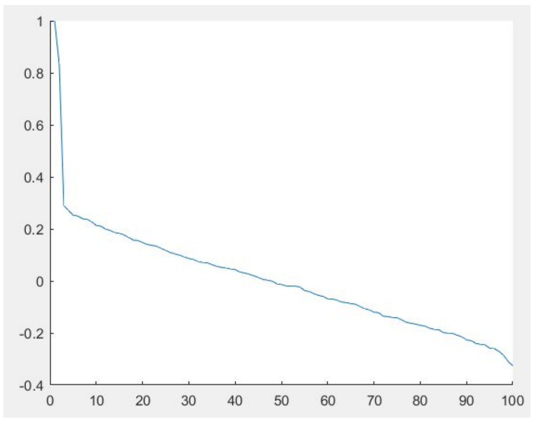
Sparsity Pattern: in this case, the plotting of the adjacency matrix is not useful for visualizing the communities present in the graph. This could mean that there is a high degree of conductivity among the clusters.



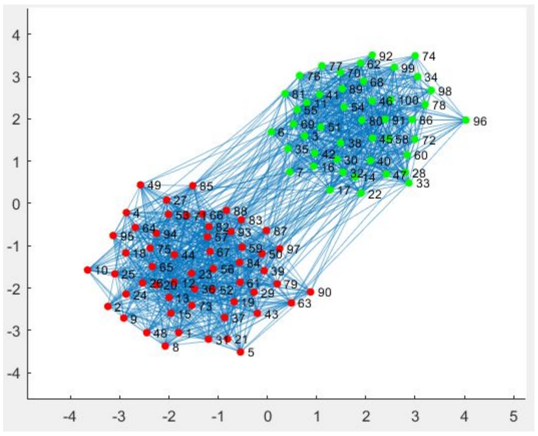
Fiedler Vector: by analyzing the first eigenvector with non-null eigenvalue, it is possible to find out that there are at least 2 communities present in the graph.



Eigenvalues: as in the previous dataset, the following plot shows all the eigenvalues of the graph, sorted from in the descendent order, in this case, the biggest eigengap is found between the third and second largest eigenvalues.



Spectral Clustering: the output of the implemented spectral clustering algorithm, which proves the presence of two highly interconnected clusters in the graph.



References

[1] Ng, Andrew Y., Michael I. Jordan, and Yair Weiss. "On spectral clustering: Analysis and an algorithm." *Advances in neural information processing systems*. 2002.

[2] Dataset from Coleman, James, Elihu Katz, and Herbert Menzel. "The diffusion of an innovation among physicians." *Sociometry* 20.4 (1957): 253-270.