Homework 4 Report – Graph Spectra

Data Mining – ID2222 – Homework Group 23

Adriano Mundo (mundo@kth.se), Edoardo Realini (erealini@kth.se)

November 30, 2020

1. Introduction

The goal of the *homework assignment 4* was to implement and test the spectral graph clustering algorithm [1]. We had to implement our own version of the K-eigenvector algorithm and analyse two sample graphs, provided as .dat files on the course Canvas website. The two datasets are: a real graph prepared by Ron Burt and collected from physicians in Illinois and a synthetic graph, respectively.
The following report explains the implementation and how to run the code for the task described above. The implementation has been done by using Python. We leveraged on the knowledge from the scientific paper [1] and the lectures [2].

2. Implementation

The code is organized in classes. For each task of the assignment there is a class or multiple classes with the respective methods. They are used for running the *Graph Spectra* algorithm in the *main.py* file. In this section we will explain what each file contains and what is used for.

- ***GraphMatrixReader:*** this python file contains the implementation of the csv file parser, the creation of lists for the graph, the creation of graph with respect to the dependencies and of the adjancency matrix.

  The main attributes are:
  - A *NumPy* ndarray version of the matrix
  - The number of nodes
  - The graph object created with the package *NetworkX*
  - The values of start and end nodes list

  The main methods are:
  - *init*: the constructor of the object *GraphMatrixReader* which instantiates the new objects and set the attributes to the default values
  - *gen_ matrix:* the function by taking as input the row indices, columns indices and the shape, hence from the input data creates the adjacency matrix
  - *gen_ nx_ graph:* the function creates the *NetworkX* graph needed for visualization of the clusters
  - *read_ simple_ graph:* the method reads a graph as csv file with the two columns of start and end nodes and store it as a sparse matrix. The result is the adjacency matrix of the given graph and to that this function leverages on the *gen_ matrix* function previously defined.
- ***SpectralGraphClustering:*** this file contains the implementation of the Graph Spectra algorithm presented in [1]. This class makes use of the *NumPy* and *SciPy* library.

The main attributes are:

- The graph matrix and the affinity matrix.
- The respective D, L, X, Y matrices.
- The labels and the clusters

The main methods are:

- *two_ rows_ affinity*: the function by taking as input two rows and the value of sigma, return the value of the affinity between the two rows.
- *compute_ affinity*: the function simply calculates the whole affinity matrix as stated by the formula in the paper where $A_{ii}=0$
- *compute_ D:* the function calculates the degree matrix which is a diagonal matrix where each element is equal to the sum of the values of the corresponding row in the affinity matrix.
- *compute_ L:* the function computes the Laplacian matrix by multiplying the inverse of D with the affinity matrix multiplied again with the inverse of D. Everything is implemented as dot product between matrices.
- *compute_ X:* the function creates the matrix X with stacking the eigenvectors in columns where the number of eigenvectors has been received as input for the funtion
- *compute_ Y:* the function creates the matrix Y by normalizing each of X's row to have unit length.
- *clusterize_ Y:* clusterize each row of the Y matrix into k clusters via K-means for minizing distorsion by taking as input the number of clusters.
- *clusterize_ graph:* entry point for the class, this method calls the sequence of actions and creates a dictionary by taking as input the number of k clusters, sigma and the number of eigenvectors

- ***main****:* this file is the entry point of the project. In this file the graph matrix reader object is constructed by the *GraphMatrixReader* class and through the *read_ simple_ graph* method call is filled with the data read from the dataset. Subsequently, the object *SpectralGraphClustering* is constructed and by calling the function *clusterize_ graph* algorithm is run on the graph. In the end it prints the labels, the clusters and show an image of the clusterized graph.

The program has been tested by using the two datasets available on Canvas (real and synthetic data).

3. How to Run the Code

In this section it is explained how to run the code provided with this short report. The recommended environment is Ubuntu >= 18.04 LTS. It is also requested to have installed on the machine Python3 with the following dependencies: *NumPy, SciPy* ,*NetworkX* and *Matplotlib.*

The pipeline can be run by using the following command in the directory *src.*

```
$ python3 main.py
```

It is possible to use command line arguments to set the number of clusters to find, the sigma parameter for the computation of the affinity, the number of top K eigenvectors to keep, the filename of graph in data project folder.

The default configuration is the one below (for example1.dat).

```
$ python3 main.py -k 4 -s 5 -k_ev 4 -f example1.dat
```

The default configuration is the one below (for example2.dat).

```
$ python3 main.py -k 2 -s 5 -k_ev 2 -f example2.dat
```

4. Results

By running the program, it is possible to see the following output which shows each operation correctly executed. There are two results blocks for the Graph Spectra algorithm. The first one is for example1.dat, while the second is for example2.dat. In addition, a plot of the clusters identified from the algorithms are shown.

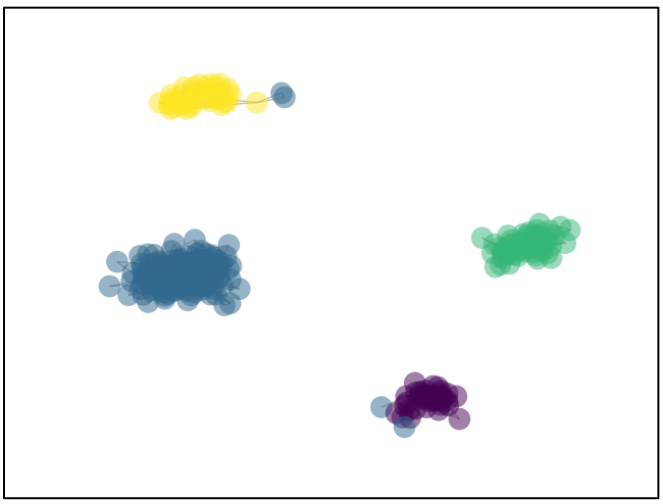
```
Spectral Clustering on Graphs
Adriano Mundo & Edoardo Realini
KTH Royal Institute of Technology - 2020

Running clustering algorithm with parameters:
Number of clusters:  4
Sigma for affinity computation:  5.0
Number of Top k Eigenvectors:  4

--------------------------------------------------------------------------------
--
Computing data from file:  ../data/example1.dat
Reading graph from file ../data/example1.dat with 241 nodes.

Labels:  [1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0]

Clusters:  {0: [207, 208, 209, 210, 211, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222,
223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 240, 241],
1: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24,
25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93,
94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
113, 114, 115, 116, 117, 180, 181, 212, 239], 2: [118, 119, 120, 121, 122, 123, 124, 125,
126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143,
144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161,
162, 163, 164, 165], 3: [166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178,
179, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198,
199, 200, 201, 202, 203, 204, 205, 206]}
```



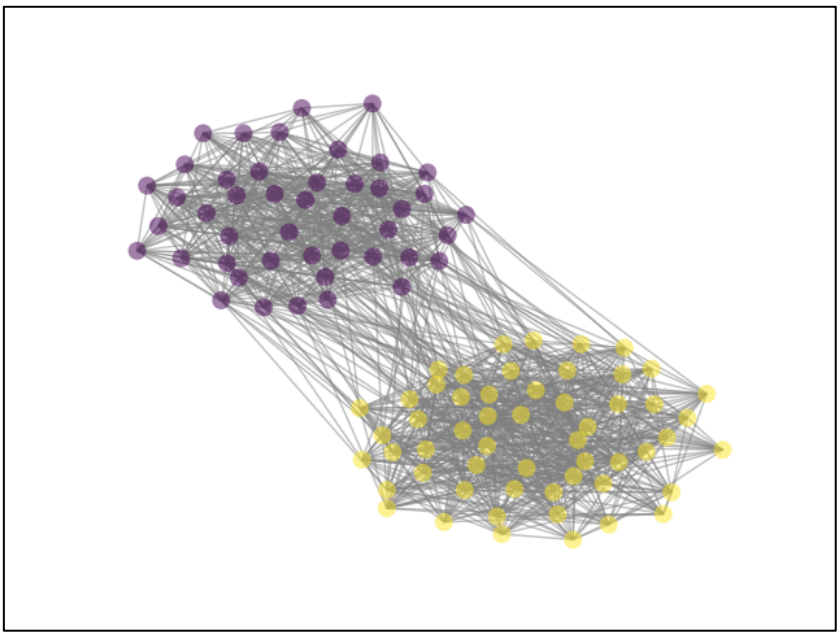
```
Spectral Clustering on Graphs
Adriano Mundo & Edoardo Realini
KTH Royal Institute of Technology - 2020

Running clustering algorithm with parameters:
Number of clusters:  2
Sigma for affinity computation:  5.0
Number of Top k Eigenvectors:  2

--------------------------------------------------------------------------------------------
--
Computing data from file:  ../data/example2.dat
Reading graph from file ../data/example2.dat with 100 nodes.

Labels:  [1 1 0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1 1 1 1 0 1 1 1 1 1 0 1 0 1 0 0 0 0 1 1
 0 1 0 0 0 1 1 0 0 0 1 1 1 0 1 1 0 0 1 1 0 1 0 1 0 1 1 1 1 1 1 0 0 0 1 0 1 0
 1 0 0 0 1 0 0 1 1 1 1 0 1 1 0 1 0 0 1 1 1 0 1 0 0 0]

Clusters:  {0: [3, 6, 7, 11, 14, 16, 17, 22, 28, 30, 32, 33, 34, 35, 38, 40, 41, 42, 45,
46, 47, 51, 54, 55, 58, 60, 62, 68, 69, 70, 72, 74, 76, 77, 78, 80, 81, 86, 89, 91, 92, 96,
98, 99, 100], 1: [1, 2, 4, 5, 8, 9, 10, 12, 13, 15, 18, 19, 20, 21, 23, 24, 25, 26, 27, 29,
31, 36, 37, 39, 43, 44, 48, 49, 50, 52, 53, 56, 57, 59, 61, 63, 64, 65, 66, 67, 71, 73, 75,
79, 82, 83, 84, 85, 87, 88, 90, 93, 94, 95, 97]}
```



References

[1] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. 2001. On spectral clustering: analysis and an algorithm. In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01). MIT Press, Cambridge, MA, USA, 849–856.

[2] Graph fundamentals, Graph models, Link analysis and Spectral Clustering lectures, ID2222, Data Mining, KTH Royal Institute of Technology