POLITECNICO DI TORINO

Master's Degree in
Mathematical Engineering

Angelini Sergio A. **, matricola 300873**
Mail: s300873@studenti.polito.it
Gallo Vittorio**, matricola 300829**
Mail: s300829@studenti.polito.it
Grobbo Filippo**, matricola 305723**
Mail: s305723@studenti.polito.it

Final essay for the course of Computational Linear Algebra

Homework Spectral Clustering

Academic Year 2022-2023

Problem Overview

In this report we introduce our approach to the homework about Spectral Clustering. The main task in clustering analysis is grouping objects into groups (called clusters) according to an arbitrary similarity/dissimilarity criterion. Spectral clustering is a technique that puts together different tools:

- graphs theory
- properties of matrices eigenvalues and eigenvectors
- K-means algorithm

In few words the algorithm can be summarised in the following pseudocode:

Algorithm 1

function Spectral clustering
Input
x_i: set of points ($x_i \in R^n$, i=1,..,N)
Output
c_j: set of clusters (j=1,..,K)
Define a similarity function
Compute the similarity graph
Construct the Laplacian matrix of the graph
Compute the smallest eigenvalues according to some criterion
Construct the matrix U that contains as columns eigenvectors corresponding to the smallest eigenvalues
Consider the rows of U as a new set of points y_i (i=1,..,N) and compute K-means on them
Associate the clusters obtained for y_i to the corresponding x_i (1 to 1 correspondence)
end function

Proposed approach

For this project we use the following elements:

- Circle and Spiral datasets
- similarity function $s_{ij} = exp\left(-\frac{\|X_i - X_j\|^2}{2\sigma^2}\right)$ with $\sigma = 1$
- K-nearest-neighborhood similarity graph. In simple words each node is connected to the k closest nodes in terms of distance and in both directions in order to obtain an undirected graph. The value of k is variable, we choose 10 most of the times.
- Elbow method for computation of the smallest eigenvalues.

After some ad hoc initializations, computation of K-nearest-neighborhood is done thanks to the function KNN_similarity_graph:

```
%% Function that computes the weight matrix of the KNN-similarity graph.

function W = KNN_similarity_graph(data, similarity_function, K)
% KNN_similarity_graph: Function that computes the weight matrix of the
%                       KNN-similarity graph.
%
%
% INPUTS: data: the matrix representing our data,
%         similarity function: function handle of two variables for ...
    create
%                              the KNN graph,
%         K: number of neighbours for the KNN algorithm.
%
% OUTPUTS: W: n_data by n_data matrix representing the weight matrix ...
    of the
%            KNN-graph.

          % Finding the number of data.
          [n,¬] = size(data);

          % Allocating in a sparse way the weight matrix of the graph
          % because there will be only K elements for each row; the ...
             sparsity
          % of this matrix will be K/n. The first allocation has all zero
          % values.
          W = spalloc(n,n,K*n);

          % Setting a tolerance useful for the similarity function.
          tol = 1e-8;

          % Loop for the calculation of the similarity function for ...
             all the
          % data.
          for i = 1 : n

              S = zeros(1,n);

              % Calculate only the similaries for the next data ...
                 because of
              % the simmetry of the weight matrix.
              for j = i + 1 : n

                  Sij = similarity_function(data(i, :), data(j, :));

                 if Sij ≥ tol % For the Gaussian similarity function,
                              % the minimum is 0.
                    S(j) = Sij;

                 end

              end

              % Now take only the K nearest neighbours of each element.
```

```
49                [vals, pos] = maxk(S, K);
50
51                % Here we find the first K elements and put them in the ...
                      weight
52                % matrix.
53                h = sparse(pos, 1,vals, n, 1);
54                W(:, i) = h;
55
56            end
57
58            % Takes bottom half of W to make W symmetric.
59            W = triu(W.',1) + tril(W);
60
61  end
```

Basically, we sparsly allocate the similarity matrix W and then calculate values of similarity function above a fixed threshold. After this, for each node of the matrix we consider only the K nearest neighbors and fill the i-th column of W in a sparsly way. Finally we return as W only the upper and lower diagonal part of the W previously obtained, since loops are not taken into account.
Having the similarity graph we can visualize its sparsity structure, compute Laplacian matrix and its eigenvalues. Figure 1 gives some plots for the spiral dataset.

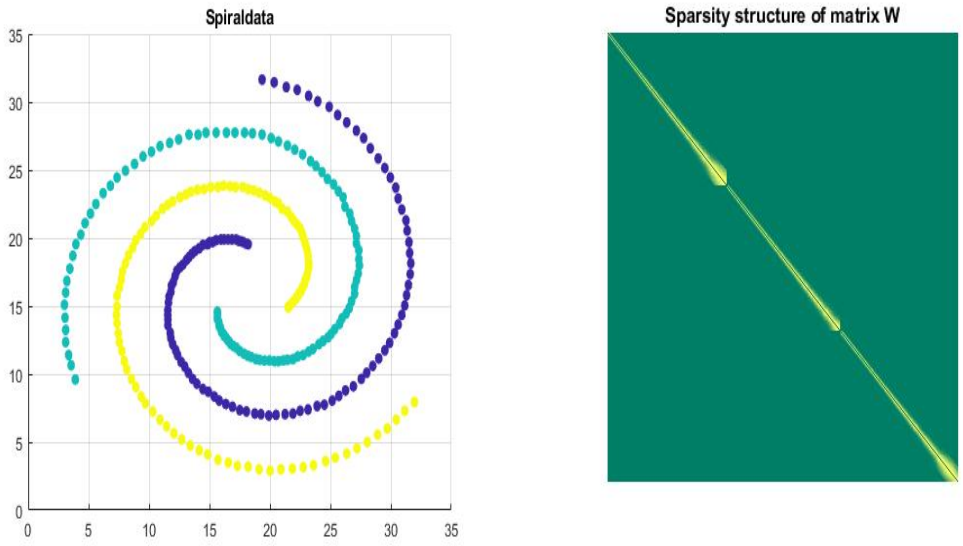

Figure 1. Spiral dataset information.

Regarding the elbow method, it essentially consists of plotting eigenvalues and picking the number of eigenvalues that corresponds to the first elbow of the curve. This will be used as number of eigenvectors to take into consideration and will determine the width of matrix U. In a nutshell, the number can be interpreted as the dimension into which we

want to project our starting points. Since the elbow method is an heuristic, we decide to leave the choice of the number to code's user, even if a suggestion is made by us. Essentially, our suggested value is obtained taking cumulative sum of the absolute value of eigenvalues. Once this is done, a vector is obtained and the chosen number is the index associated to the first maximum value of the vector. Intuition behind our reasoning is that when cumulative sum has the first peak it means that we find the elbow. For Spiral and Circle dataset it is simple to observe that the correct number of smallest eigenvalues to pick is the real number of connected components (which is also the suggested value). For example, Figure 2 shows that the function increase more passing from $x = 3$ to $x = 4$, rather than passing from $x = 2$ to $x = 3$. This is why eigenvectors associated to the 3 smallest eigenvalues are taken in consideration.

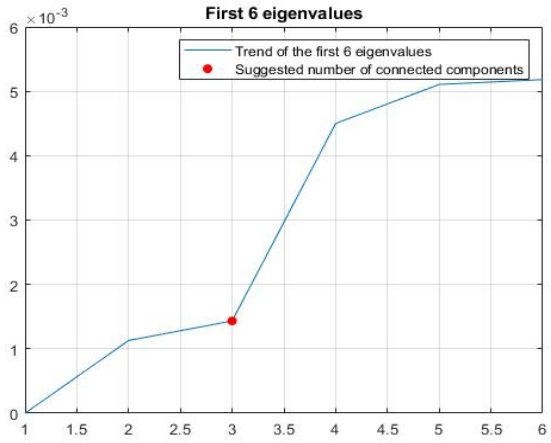

Figure 2. Elbow method for Spiral dataset.

After all these considerations, if the resulting number from the elbow method is 2 or 3, we make two additional analysis about eigenvectors. These information are summarized in Figure 3. On the left it can be seen the plot of the components of each eigenvector. Along x axis in fact there are the indexes of each eigenvector component while on the y axis the respective value assumed by the component. From this plot we can state that eigenvectors associated to the 3 smallest eigenvalues have quite different behaviour, since looking for example to fixed indexes, we observe opposite value on the y axis for each eigenvector. Instead, the plot on the right shows eigenvectors represented in the space spanned by the columns of U. As a proof of concept we can see that, in the Spiral dataset, eigenvectors associated to the 3 smallest eigenvalues are well separated. This is giving us a visual proof that spectral clustering is going to work well because in the projected space points are well separated. It is useful to observe that the plot on the left can be done also in case of higher number of clusters.
The final part of our approach consists in applying K-means on the matrix U previously obtained and plot original points divided by clusters. The same is done applying different clustering techniques that are known in literature, such as DBSCAN, Hierarchical

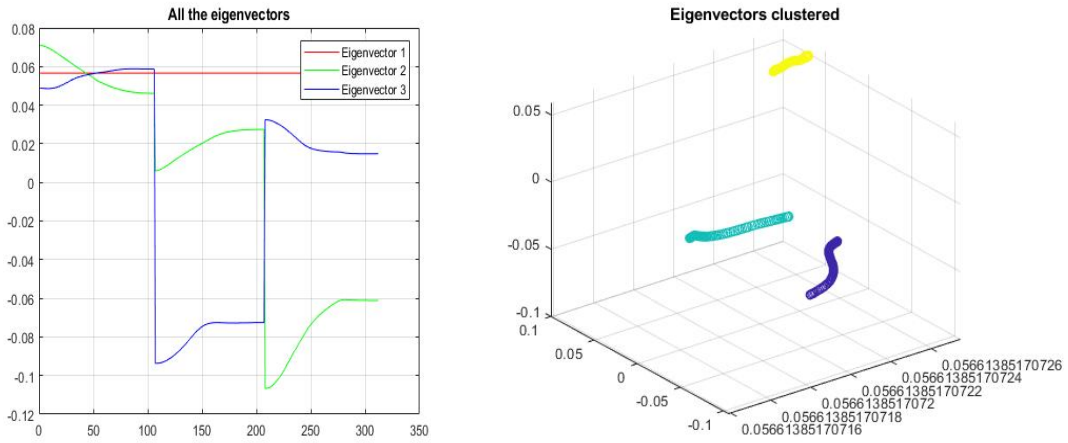

Figure 3. Eigenvectors plots.

clustering and simply K-means.

Results

Plotting the comparison with other methods and using the same number of clusters for every techniques (number of clusters is the one given in input), we obtain the results showed in Figures 4 and 5. In both cases we can conclude that Spectral clustering is better with respect to other methods, since only Hierarchical clustering obtains also a good result, but just for the Spiral dataset.

Spectral clustering for 3D datasets

Looking in literature for 3D datasets we find 7 possible different datasets in [1]:

- Atom
- EngyTime
- GolfBall
- Hepta
- Target
- Tetra
- TwoDiamonds

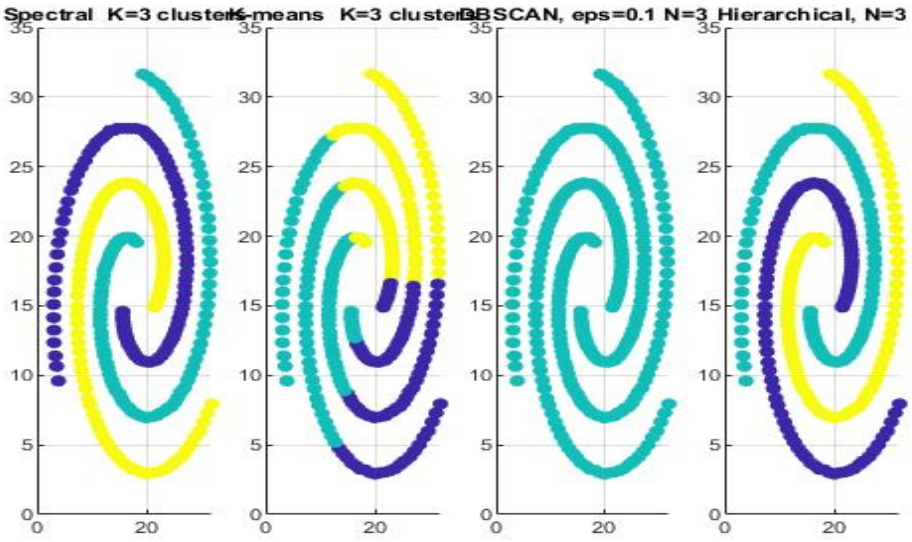

Figure 4. Spiral dataset results.

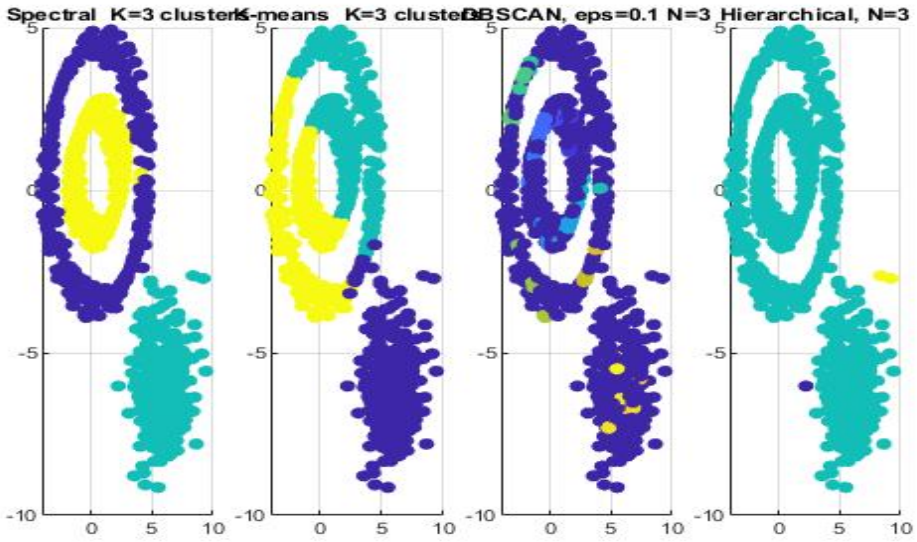

Figure 5. Circle dataset results.

We repeat the same process of previous sections for both the Laplacian and normalized symmetric Laplacian matrices. It was at this point of the work that we decide to give to code's user the choice of number of smallest eigenvalues. In fact, trying to automatize the elbow method is something very easy for toy 2D datasets. When things become more complicated (higher dimensions) it is difficult to discern an elbow if all eigenvalues are in the order of magnitude of 0 and some are also negative (due to numerical approximation). This is the case for example of Atom dataset. As shown on the left of Figure 6, the majority of eigenvalues are 0 (this could be seen increasing the range on the x axis). Even if it is almost clear that we have an elbow corresponding to $x = 2$, Spectral clustering does not work so well on this dataset (as shown on the right of Figure 6).

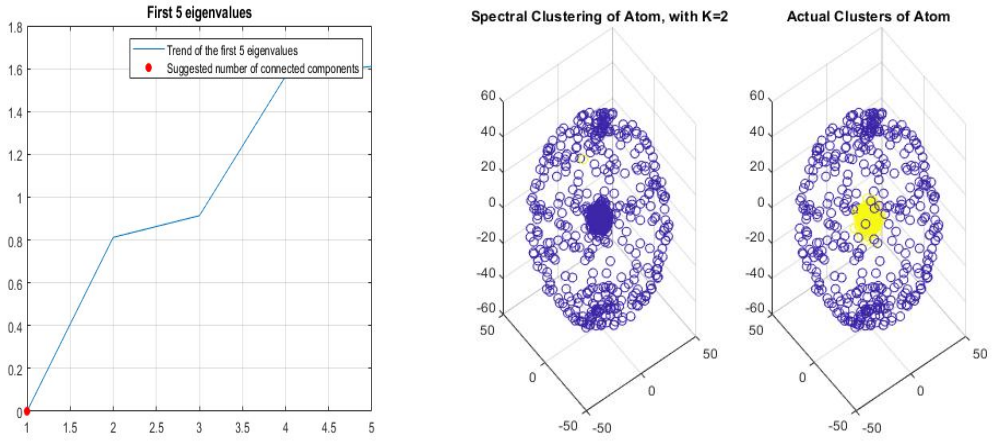

Figure 6. Atom dataset plots.

For this reasons, for each 3D dataset, we decide to tune a little bit some parameters and obtain results in table 1. SC state for Spectral clustering, L is the Laplacian matrix, L_s is the corresponding normalized symmetric version and the other parameters are called with same names used in Matlab code.

Table 1. Results for 3D datasets

	Atom	**EngyTime**	**GolfBall**	**Hepta**	**Target**	**Tetra**	**TwoDiamonds**
SC functions with L	yes	yes	yes	yes	no	yes	yes
SC functions with L_s	yes	yes	yes	yes	no	yes	yes
KNN	10	2	10	10	10	10	10
comp_conn	4	3	3	9	6	6	4
rescaling	yes	no	no	no	yes	no	no
real number of clusters	2	2	1	7	6	4	2
suggested number of clusters	incorrect	correct	correct	correct	correct	correct	incorrect

To summarize, we observe that in the majority of 3D datasets our algorithm performs correctly. The issue about Atom dataset was solved applying a rescaling to the points. In fact, as it could be seen on the right of Figure 6, initial points in the 'nucleus' of the

atom are very far from other points. This implies having low values of similarity function. Rescaling points helps to maintain proportion and obtain values with no problems in evaluating similarity function. Side effect of this rescaling is that our suggested number of clusters is not correct anymore. The main reason is that it becomes more difficult to discern smallest eigenvalues, since they are almost all 0. However, after trying some values, we find the right one that separates points in correct groups (see Figure 7). This makes us believe that our algorithm about Spectral clustering is accurate, while the suggestion for the elbow method can be improved. The same happens in EngyTime L_s case, where we need also to force $KNN = 2$ to obtain good results. The only dataset we are not able to separate is Target. Looking at eigenvalues plot we observe they have more or less a parabolic pattern. This implies that, even if we change the scale of eigenvalues plots, we are not able to find any elbow because essentially there are none. Another possible reason for issues in this dataset can be related to the **presence of outliers**, as it could be seen from plot of points with real clusters (see Matlab code).

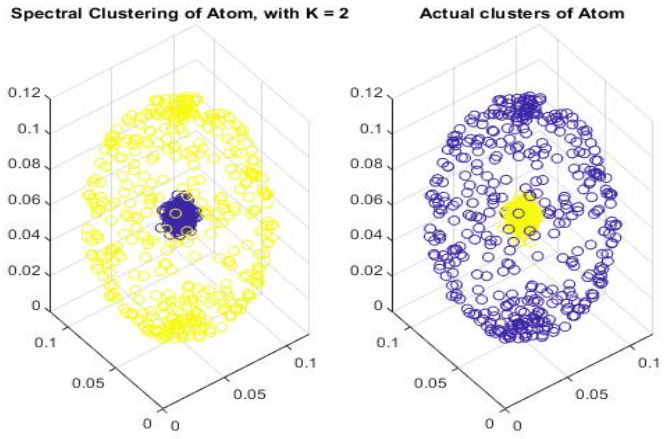

Figure 7. Atom results after rescaling and trying different values of clusters instead of the suggested one.

Relation between the eigenvalues of L and L_s

The relation between the eigenvalues of the Laplacian matrix and the normalized symmetric Laplacian matrix can be calculated as follows; we start from two eigenvalues problems:

$$Lx = \lambda_L x \qquad L_s v = \lambda_s v \tag{1}$$

Furthermore we know that one random eigenvalue of L and L_s can be written as the Rayleigh quotient w.r.t. the associated eigenvector (in addition we can assume that the corresponding eigenvectors are normalized without any loss of generality):

$$\mathcal{R}_L(x) = \lambda_L = x^T L x \qquad \mathcal{R}_{L_s}(x) = \lambda_s = v^T L_s v \tag{2}$$

Now we can calculate the ratio between the two eigenvalues obtaining:

$$\frac{\lambda_s}{\lambda_L} = \frac{v^T L_s v}{x^T L x} = \frac{v^T v - v^T D^{-\frac{1}{2}} A D^{-\frac{1}{2}} v}{x^T L x} = \frac{1 - \mathcal{R}_{D^{-\frac{1}{2}} A D^{-\frac{1}{2}}}(v)}{\mathcal{R}_D(x) - \mathcal{R}_A(x)} \tag{3}$$

and so it results that:

$$\lambda_s = \lambda_L \left[\frac{1 - \mathcal{R}_{D^{-\frac{1}{2}} A D^{-\frac{1}{2}}}(v)}{\mathcal{R}_D(x) - \mathcal{R}_A(x)} \right] \tag{4}$$

It can be also proven numerically for all the previous examples.

Relation between the eigenvectors of L and L_s

Regarding this paragraph we can confirm that zero is eigenvalue for both the matrices; it follows from the fact that if we apply the vector made by all 1 in the indices corresponding to a single connected component of the corresponding similarity graph, we obtain the following result:

$$L_s = D^{-\frac{1}{2}} L D^{-\frac{1}{2}} \Rightarrow L_s D^{\frac{1}{2}} = D^{-\frac{1}{2}} L \tag{5}$$

$$L_s D^{\frac{1}{2}} \mathbf{1}_{comp\_conn} = D^{-\frac{1}{2}} L \mathbf{1}_{comp\_conn} = \mathbf{0} \tag{6}$$

As a consequence, the corresponding eigenvector is $D^{\frac{1}{2}} \mathbf{1}_{comp\_conn}$.

Bibliography

[1] M. C. Thrun and A. Ultsch. Clustering benchmark datasets exploiting the fundamental clustering problems. *Data in Brief*, 30:105501, 2020.