Numerical Linear Algebra
A.A. 2025/2026

Lecturer: Prof. P. F. Antonietti TA: Dott. M. Botti

Challenge 2 - Eigenvalue problems applied to clustering and social networks

Goal: Apply linear algebra techniques and eigensolvers to data clustering.
Data: Download `social.mtx` containing the adjacency matrix of a simulated social network.

Overview

Many practical applications deal with large amounts of data. One of the standard problems in data science is to group this data in clusters containing similar items. In many cases, clustering can be easily accomplished by the human brain (take a look at the attached pictures).

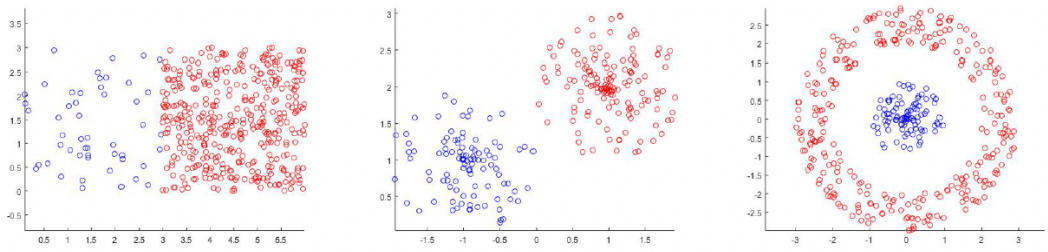

Figure 1: Examples of clustering problems.

In general, creating an algorithm which allows us to separate the data into clusters is a difficult task. The data in many cases is multidimensional and cannot be visualized easily. Even deciding how many clusters to select is not necessary an easy task. There are currently several widely used algorithms which allow for the separation of the data points by similarity. All these algorithms have their own advantages and disadvantages.

In this project, we will use the spectral clustering algorithm based on the Fiedler's vector. We will look at the graph representing an individual's profile on a social network. The profile has a total of 351 *friends*. Some of these friends know each other and are friends with each other as well. This friendship network can be represented as a graph with nodes being friends and edges showing the friendship connection between the individuals. The goal of clustering is to separate this graph into components which are tightly connected within themselves and have fewer connections to the outside (the so-called communities). The method explored here is based on the theorem establishing that for a connected graph (meaning that there is a path between any two vertices) the eigenvector corresponding to the second smallest strictly positive eigenvalue of the graph Laplacian allows the graph to be split into two maximally connected components.

Tasks

The tasks of this challenge are also reported in `https://forms.office.com/e/dDHf0j75xw`.

- We start by considering an example of a small graph. Using `Eigen`, create a new adjacency matrix A_g corresponding to the graph below. **Report the Frobenius norm of A_g.**

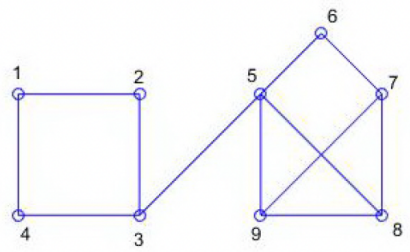

Figure 2: An example of a small graph

- Construct the vector $\boldsymbol{v}_g$ such that each component v_i is the sum of the entries in the i-th row of matrix A_g. Compute the graph Laplacian by performing the following operations:

$$L_g = D_g - A_g,$$

where D_g is diagonal matrix with the vector $\boldsymbol{v}_g$ on the main diagonal and zeros everywhere else. Compute the matrix-vector product $\boldsymbol{y} = L_g\,\boldsymbol{x}$ with $\boldsymbol{x} = [1, 1, \ldots, 1]^T$ and **report the Euclidean norm of $\boldsymbol{y}$. Is matrix L_g symmetric positive definite**?

- Using the proper eigensolver available in the `Eigen` library, find the eigenvalues and eigenvectors of the matrix L_g. **Report the smallest and largest computed eigenvalues**.

- Observe that one of the eigenvalues of the matrix L_g is zero (this corresponds to an eigenvector proportional to $\boldsymbol{x}$). All other eigenvalues of the matrix are positive. **Report the smallest strictly positive eigenvalue of L_g and the corresponding eigenvector**. Check that the positive entries corresponds to the nodes $\{1; 2; 3; 4\}$ and the negative ones to $\{5; 6; 7; 8; 9\}$ (or viceversa) which would be an intuitive clustering choice.

- Apply the previous technique to a larger graph for which a clustering strategy is not obvious. Using `Eigen`, load the adjacency matrix A_s stored in the file `social.mtx` corresponding to a network of 351 Facebook friends. The $(i; j)$-th element is equal to 1 if the i-th and j-th friends are also friends with each other and is equal to 0 otherwise. **Report the Frobenius norm of the matrix A_s.**

- Repeat the procedure we performed previously on the smaller graph. Create the matrices D_s and L_s which are the vector of row sums of the matrix A_s and the Laplacian of the corresponding graph, respectively. Check the symmetry of L_s and **report the number of nonzeros entries in L_s.**

- In order to make the graph laplacian matrix invertible, add a small perturbation to the first diagonal entry of L_s, namely $L_s(1, 1) = L_s(1, 1) + 0.2$. Export L_s in the `.mtx` format and move it to the `lis-2.1.10/test` folder. Using the proper iterative solver available in the LIS library compute the largest eigenvalue of L_s up to a tolerance of 10^{-8}. **Report the computed eigenvalue and the iterations counts**.

- Find a `shift` $\mu \in \mathbb{R}$ yielding an acceleration of the previous eigensolver. **Report μ and the number of iterations required to achieve a tolerance of 10^{-8}.**
- Use the proper iterative solver available in the LIS library to identify the second smallest positive eigenvalue and the corresponding eigenvector setting a tolerance of 10^{-10}. **Report the computed eigenvalue and the iteration counts**.
- Sort the vertices of the graph according to the positive and negative coefficients of the eigenvector computed in the previous point. **Report the number n_p of positive entries and the number n_n of negative entries**.
- In Eigen, construct the permutation matrix P which reorders the nodes of the network according to the previous clustering strategy (first the n_p nodes associated to positive coefficients, then the n_n nodes associated to negative ones). Computed the reordered adjacency matrix $A_{ord} = P\ A_s\ P^{\mathrm{T}}$. **Report the number of nonzero entries in the non-diagonal blocks $A_{ord}(1 : n_p, n_n + 1 : n_p + n_n)$ and $A_s(1 : n_p, n_n + 1 : n_p + n_n)$.**
- **Comment the obtained results**.