HOMEWORK:
Eigenvalues and Eigenvectors Approximation

Computational Linear Algebra for Large Scale Problems
Politecnico di Torino

A.Y. 2019/2020

Abstract

In this homework the students are required to apply the algorithms for the approximation of eigenvalues and eigenvectors studied during the course in order to analyze a dataset consisting of "friend list" from Facebook.

1 Introduction

The aim of the present homework is to investigate some properties of the undirected graph related to the dataset of "friend list" from Facebook using tools from numerical linear algebra among which numerical approximation of eigenvalues and eigenvectors. To this end all the needed information about graph theory will be provided.

2 Instructions

The numerical codes that are needed to solve the homework can be written using either Matlab or Python as programming languages. The choice is up to you. This homework must be done alone or in a group consisting of two people at most.
As a report of the work done to solve the homework, you are requested to upload[1]:

- a report (*.pdf* file) containing the answers to all the questions, the results and their detailed explanation;
- an archive (e.g. a *zip* file) containing all the implemented codes (*.py* or *.m* files) together with the *INSTRUCTIONS.txt* file that must contains

[1]Details on the uploading will be provided as soon as possible.

the explanations on how to run the codes and reproduce the results presented in the report.

3 Homework: eigenvalues and eigenvectors approximation applied to Facebook dataset

We consider the dataset consisting of "friend list" from Facebook taken from the website http://snap.stanford.edu/data/ego-Facebook.html. These data have been collected in a file named *edge_file_facebook_matlab.txt* for Matlab (or *edge_file_facebook_python.txt* for Python). The structure of this file will be described later. Facebook data have been anonymized by replacing the Facebook-internal ids for each user with a new value.
This dataset describes a network of connections among the Facebook users. In particular, we focus on the **undirected graph** underlying this structure.

A **simple undirected graph** $G = (V, E)$ consists of a non-empty set V of **vertices** and a set E of unordered pairs of distinct elements V, called **edges**.
Given a graph G we can associate to it a matrix A called the **adjacency matrix**. Its elements indicate whether pairs of vertices are adjacent or not in the graph

$$A_{ij} = \begin{cases} 1 & \textit{if vertex } i \textit{ is adjacent to vertex } j; \\ 0 & \textit{if vertex } i \textit{ is not adjacent to vertex } j. \end{cases}$$

The **spectrum of a graph** is defined as the set of the eigenvalues of its adjacency matrix A.
Our graph consists of 4039 nodes and 88234 edges. The nodes are numbered starting from 1 in Matlab and starting from 0 in Python. The connection between the nodes are described in the file *edge_file_facebook_matlab.txt* (or *edge_file_facebook_python.txt*). This file consists of two columns of integer numbers. The first column contains the numbers related to starting nodes, while the second column the numbers related to the ending nodes. However, keep in mind that the the graph is undirected, that is, edges do not have a direction. The edges indicate a two-way relationship and each edge can be transversed in both directions.

1. Construct the adjacency matrix A of the given graph in a sparse storage format.

2. Graphically visualize the sparsity pattern of the matrix A.

3. Look at the definition of the matrix A. What can you say about the spectrum of the adjacency matrix of our undirected graph? And what can you say about its eigenvectors?

A measure of the influence of a node in a network is given by the so called **eigenvector centrality**. This type of centrality takes into consideration not only how many connections a vertex has, that is, its **degree**, but also the degree of the vertices it is connected to [2]. Therefore, it measures a node's importance while giving consideration to the importance of its neighbors. The eigenvector centrality is computed starting from the eigenvector x_1 corresponding to the eigenvalue λ_1 of A having the largest absolute value.

4. Write a function that implement a suitable numerical method among those studied during the course to approximate both λ_1 and x_1. Then, consider a new vector $\tilde{x}_1$ whose components are the absolute values of the components of x_1 and normalize it such that the sum of all its components equal one.

The ith component of the vector $\tilde{x}_1$ gives the centrality score of the ith node in the graph and the scores are normalized so that the sum of all centrality scores equals one.

1. Which one of the node has the highest eigenvector centrality? What is the value of its centrality? Discuss the results.

2. Check the results comparing them with the ones obtained using the Matlab/Python functions to numerically approximate eigenvalues and eigenvectors.

The second matrix that we consider is the **Laplacian matrix** L

$$L_{ij} = \begin{cases} d_i & \text{if } i = j, \\ -1 & \text{if } i \text{ is adjacent to } j, \\ 0 & \text{otherwise,} \end{cases}$$

where d_i is the degree of the ith vertex, that is, the number of connections the vertex ith has. As it can be seen from the above definition, L is related to the adjacency matrix A. Indeed, $L = D - A$, where D is the diagonal matrix having the degree values on the main diagonal.

3. Construct the Laplacian matrix L of the given graph in a sparse storage format.

4. Verify that each row sum to zero. Given that, what can you say about the spectrum of the eigenvalues of L and its eigenvectors?

The last matrix that we consider is the **normalized Laplacian** $\mathcal{L}$

$$\mathcal{L}_{ij} = \begin{cases} 1 & \text{if } i = j, \\ \frac{-1}{\sqrt{d_i d_j}} & \text{if } i \text{ is adjacent to } j, \\ 0 & \text{otherwise.} \end{cases}$$

[2] The measure of eigenvector centrality is a key part of Google's PageRank algorithm for measuring the "importance" of web pages.

For graphs with no isolated vertices $\mathcal{L}$ is related to L according to the following relationship $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$.

5. Construct the normalized Laplacian matrix $\mathcal{L}$ in a sparse storage format.

The eigenvalues of the Laplacian can be essentially as large as desired, while it has been proved that the eigenvalues of the normalized Laplacian always lie in the range between 0 and 2. Moreover, the second smallest eigenvalue in modulus λ_2 of $\mathcal{L}$ tells us about the connectivity of the graph. If the graph has two disconnected components $\lambda_2 = 0$. If λ_2 is small this suggest that the graph is nearly disconnected, that is, it has two components that are not very connected to each other.

6. Keeping this pieces of information in mind, write a code that allows you to approximate λ_2 using the numerical algorithms studied during the course (*Hint: try to use the power method with deflation in a smart way*).

7. Discuss the connectivity of the given graph.

8. Check the results comparing them with the ones obtained using the Matlab/Python functions to numerically approximate eigenvalues.