Application of Eigenvalues and Eigenvectors algorithms in real life problems

Edoardo Fantolino

Abstract

This document contains my idea on the assignment for the course of Computational Linear Algebra for Large Scale Problems.

This document is addressed to Professor Stefano Berrone and Martina Busetto.

Keywords: eigenvalues, eigenvectors, graph theory, Markov chain.

1 Introduction

In this homework, we have to explore the so called ranking problem, the problem of ranking a large number of elements. In particular, we have to deal with a web pages structure. Basically, we simulate a small web with 2500 web pages linked to each others. We need to use different mathematical tools to extrapolate some useful information out of this set of web pages.

2 The directed graph of our Web, Markov Chain and Perron-Frobenius

The directed graph represent the connection between our different web pages. We can associate to a directed graph an adjacency matrix. We will call this adjacency matrix G. The entry of G are define as fallow:

$$g_{ij} := \begin{cases} 1 & \text{if there is a link from page j to page i} \\ 0 & \text{otherwise} \end{cases}$$

The information about our problem are stored into a .txt file called edges_file_matlab. I imported this information in a sparse matrix. Then as required, ($\star$1) I have plotted the sparsity pattern of this matrix. You can see the results in Fig.1.

($\star$2) The j-th column of the G matrix represent all the pages that we can reach from page j. Basically, if the i-th entry of this column vector has as value 1, that means that we have a link in page j that allow us to go in page i. If this entry is equal to zero, we do not have a link from page j to page i, that means that we cannot directly go from page j to page

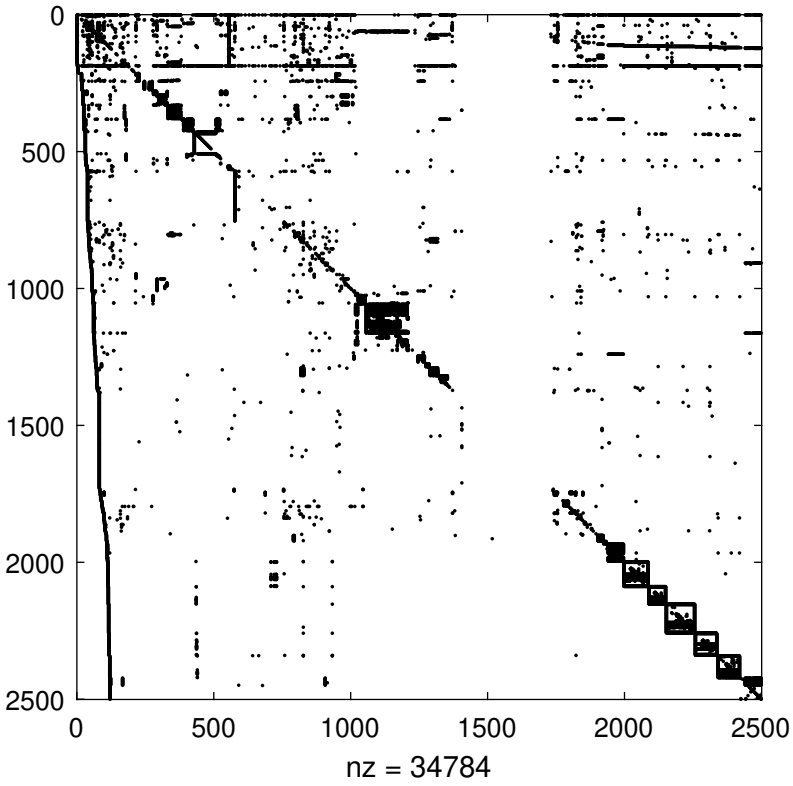


FIGURE 1: G matrix sparsity pattern.

i. The number of nonzeros elements of the G matrix represent the total number of links in our web.

($\star$3) Now, we need to compute the indegree and outdegree of each web page. The indegree of a page k is given by the number of nonzeros elements in the k-th row. While the outdegree of the page k is given by the number of nonzeros elements in the k-th column.

($\star\star$4) If our stochastic matrix is primitive we can certainly say that there exists and is unique the stationary probability distribution vector thanks to the Perron-Frobenius theorem. We can say that because this theorem guarantees us that:

- There exists an eigenvector such that all components are positive. This is clearly a probability distribution vector, because we can normalize it in a way that the sum of the components of this vector will be 1. Basically, we will divide this eigenvector by the sum of its components to obtain a suitable result and we will not ruin the relationship with his respective eigenvalue.

  Proof:

$$Ax = \lambda x$$

$$kAx = k\lambda x$$

$$A(kx) = \lambda(kx)$$

this mean that even kx is an eigenvector of the eigenvalue λ. In fact, the most important aspect regarding an eigenvector is its direction and not its magnitude. In fact when we find an eigenvector in reality we find a direction in which lies an infinite number of eigenvectors.

- There are no others positive eigenvectors except positive multiples of the previous one. All the other eigenvectors must have at least one negative or non-real component. That means that the probability distribution vector is unique because all the other eigenvectors have some negative components and that means that they do not represent probabilities. Negative probability don't make sense.

3 In search of the most representative matrix and eigen analysis

(⋆5) In request number 5, we should construct the hyperlink matrix M. This matrix is defined as follow:

$$m_{ij} := \begin{cases} \frac{g_{ij}}{c_j} & \text{if } c_j \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

We understand that the sparsity pattern of matrix G and matrix M is equal because given the definition of the hyperlink matrix we change only the values of the non zeros elements. Basically, we go from a non weighted adjacency matrix to a weighted one. The number of non zeros elements remain the same, and the entries do not change position in the matrix. We can have a proof by plotting the sparsity pattern with the MATLAB spy function.

(⋆6) A stochastic matrix is a non-negative square matrix if and only if every column adds to 1. Our hyperlink matrix is not a stochastic matrix because we have some column that are completely empty. so, the sum of this columns is not 1. Basically the problem arise from the webpages that have an outdegree equal to 0.

I think that in general, given a dataset W describing a portion of the Web, the hyperlink matrix will not be stochastic because, as we experience every day, it is not so rare to find a page that don't have link for other pages. In fact, sometimes we have to use the "back arrow" that allow us to go to the previous page.

Then we can say that this is not really the best matrix to represent our Markov chain because it seems that we could be trapped in a cycle of webpages and never get out of it, while in the browser we have the searching bar at the top that allow us to jump from one page to another without the need of a link in the current page.

To try to fix the problem that arise from matrix M, we define a new matrix $\tilde{M}$. This new matrix is defined as follow:

$$\tilde{m}_{ij} \coloneqq \begin{cases} \frac{g_{ij}}{c_j} & \text{if } c_j \neq 0 \\ \frac{1}{n} & \text{otherwise} \end{cases}$$

($\star$8) With the modified hyperlink matrix we obtain a stochastic matrix because now if we take the columns one by one and we sum the elements we will obtain as result the number 1. A kind of drawback is that before we had 34784 non zeros elements and now we have 2482284 non zeros elements.

($\star\star$9) For the request number nine we need to prove that all the stochastic matrix has

1. $\lambda_1 = 1$ as eigenvalue
2. $\lambda_1 > |\lambda_2| \geq \cdots \geq |\lambda_n|$

For the point number 1 keep always in mind that:

$$Ax = \lambda x, \quad det(A - \lambda I) = 0$$

Let's start with noticing that if we sum the columns of the hyperlink matrix $\tilde{M}$ we obtain all 1 as shown above:

$$\tilde{M} = \begin{bmatrix} \tilde{m}_{11} & \tilde{m}_{12} & \cdots & \tilde{m}_{1n} \\ \tilde{m}_{21} & \tilde{m}_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{m}_{n1} & \cdots & \cdots & \tilde{m}_{nn} \end{bmatrix}$$

$$\sum_{i=1}^{n} \tilde{m}_{i1} = 1, \quad \sum_{i=1}^{n} \tilde{m}_{i2} = 1, \quad \cdots \quad , \quad \sum_{i=1}^{n} \tilde{m}_{in} = 1$$

Now, we take $\tilde{M}$ and we subtract the identity matrix:

$$(\tilde{M} - \lambda I) = (\tilde{M} - I), \quad \lambda = 1$$

Then, if we sum along the columns of the new matrix we always obtain 0 as result.

$$(\tilde{M} - I) = \begin{bmatrix} \tilde{m}_{11} - 1 & \tilde{m}_{12} & \cdots & \tilde{m}_{1n} \\ \tilde{m}_{21} & \tilde{m}_{22} - 1 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \tilde{m}_{n1} & \cdots & \cdots & \tilde{m}_{nn} - 1 \end{bmatrix}$$

$$\sum_{i=1}^{n} \tilde{m}_{i1} = 0, \quad \sum_{i=1}^{n} \tilde{m}_{i2} = 0, \quad \cdots \quad , \quad \sum_{i=1}^{n} \tilde{m}_{in} = 0$$

That means that the resulting matrix is not a full rank matrix, because we have a linear combination of the rows that give us a null vector. The coefficient of this linear combination are simply the value 1 for each row vector.

$$rank(\tilde{M} - I) \neq n$$

$$det(\tilde{M} - I) = 0 \longrightarrow \lambda = 1$$

To explain the point number 2, we can use the Perron-Frobenius Theorem. We have shown that the matrix $\tilde{M}$ has $\lambda \in \mathbb{R}^+$, and $\lambda = 1$. The Perron-Frobenius Theorem guarantee that if our matrix is real primitive non negative and squared, and if we find an eigenvalue that is a positive real number then any other eigenvalue in absolute value is strictly smaller than λ. The stochastic matrices are by nature, construction and definition real primitive non negative and squared, so the point number two holds true.

($\star$10) The first method that popped into my mind to satisfy the request number 10 is the power method because it allow us to find the dominant eigenvalue. Furthermore it is quite fast. Obviously we can find a different value of lambda if we change the tolerance, the number of iterations and the starting point. We increase the computational time if we increase the precision. Anyway, I have found $\lambda_{max} = 1.0000022$ while MATLAB gave us basically the exact result, that means $\lambda = 1$. The relative error obtained is 2.24×10^{-6}

It can be proven that the modified hyperlink matrix $\tilde{M}$ is in general not primitive. So, we have to further adjust the matrix. Now we consider the Google matrix A defined as follow:

$$ij := \begin{cases} \alpha \frac{g_{ij}}{c_j} + \delta & \text{if } c_j \neq 0 \\ \frac{1}{n} & \text{otherwise} \end{cases}$$

where $\alpha = 0.85$ and $\delta := \frac{1-\alpha}{n}$.

($\star$11) If we sum the values on the columns of the Google matrix we will obtain always 1, so the matrix is stochastic. Then, we can observe that we do not have any entries that is equal to zero, so the Google matrix is primitive. With these two fact, we can say that the PageRank vector exists and it is unique. We can have a further proof checking the code on MATLAB.

($\star\star$13) If we assume that each page has the same probability to be chosen as the starting page, the initial probability distribution vector is given by the column vector t having all the component equal to $\frac{1}{n}$. So, If I consider the sequence $At, A^2t, \cdots, A^kt, \cdots$ I think that this sequence will tend to the vector which has as value the respective limiting probabilities, that means that is the PageRank vector that we are looking for. The entries of the vector that we find for a k sufficiently large represent the probability of entering a given page in the long run.

($\star\star$ 14) We need to implement a method that allow us to compute the PageRank of our matrix. I decided to use the power method because it allow us to find the dominant eigenvalue and eigenvector and this is exactly what we are looking for.

($\star\star$ 15) Given that our matrix G is a sparse matrix, in requirement number 15 we need to improve the algorithm that we used before in order to take advantage of the sparsity characteristic. In the previous point we used the power method for the computation of

the dominant eigenvector. The classic power method is an iterative method and the main computation is:

$$x_{(k+1)} \longleftarrow Ax_{(k)}$$

In order to preserve the sparsity I exploited the hint given in the assignment and so I decompose the matrix A as:

$$A = \alpha GD + ez^T$$

(you can find the definition of e, x and D in the assignment text).

Then, we know that:

$$A = Ax = (\alpha GD + ez^T)x = \alpha GDx + ez^T x$$

So, I changed the power method as follow:

$$x_{(k+1)} \longleftarrow \alpha GDx_{(k)} + EZx_{(k)}$$

We use the power method over the sparse matrix αGD and we add a correction EZx.

(★★★16) Here we need to try to avoid any matrix-matrix or matrix-vector multiplication. So we have to reason in terms of columns and linear combinations.

$$\begin{bmatrix} | & | & & | \\ q_1 & q_2 & \cdots & q_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ . \\ . \\ x_n \end{bmatrix} = q_1x_1 + q_2x_2 + \cdots + q_nx_n$$

In this way if we scan our starting matrix column by column and we can avoid the computation of the components that has a empty columns.

So, inside the cycle, we have to cases:

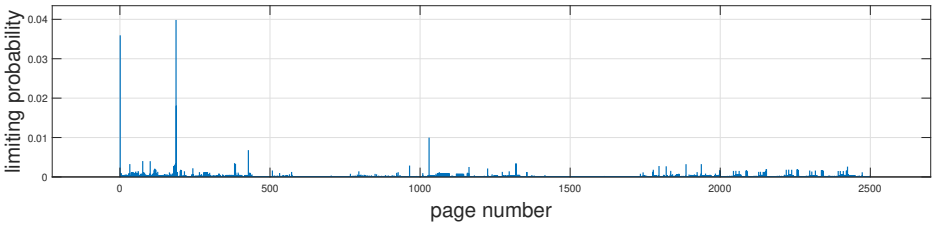
1. $c_j \neq 0 \implies$ we scale the column vector q_j of the matrix with the corresponding vector component v_j and we add it to the vector that will be our result
2. $c_j = 0 \implies$ we skip the computation and we go directly to the next iteration

I think that this technique can be useful when we take into consideration a web that has a consistent number of pages that do not have link to other pages because in this way the out-degree will be zero and we can speed up our computation.

(⋆ ⋆ 17) To solve the point number 17, I used the combination of two algorithms: the power method and the power deflation. This is because the power method allow us to compute the dominant eigenvalue while the deflation method allow us to set to zero the greater eigenvalue and then compute the new dominant one.

(⋆ ⋆ 18) From theory, we know that for a generic matrix A, the rate of convergence of $\lambda_1^{(k)}$ to λ_1 for $k \to \infty$ is the same of $\left|\frac{\lambda_2}{\lambda_1}\right|^k \to 0$ for $k \to \infty$. So we can use the previous result (power method and power deflation) to calculate an approximation of the rate of convergence.

(⋆19) As we can see from the Tab.1, the rank doesn't depend only on the overall value of in-degree and out-degree but probably the score is related even to the rank of the pages that are linked to the page taken into consideration. For example, page 1030 don't have great value of in-degree and out-degree but still it is in 4^{th} position.



The full bar chart is presented in the script wile here I prefer to reduce the number of pages to the 12 most important

rank	page	PageRank	out-degree	in-degree
1	187	$3.98 \times 10^{-}2$	13	1112
2	1	$3.59 \times 10^{-}2$	179	1109
3	188	$1.81 \times 10^{-}2$	3	283
4	1030	$9.97 \times 10^{-}3$	0	29
5	428	$6.80 \times 10^{-}3$	1	70
6	76	$4.03 \times 10^{-}3$	31	38
7	101	$4.00 \times 10^{-}3$	14	16
8	382	$3.50 \times 10^{-}3$	0	160
9	186	$3.49 \times 10^{-}3$	1	35
10	1320	$3.41 \times 10^{-}3$	1	30
11	1319	$3.41 \times 10^{-}3$	1	30
12	385	$3.31 \times 10^{-}3$	1	41

TABLE 1: Table for the request number 19

References

Abhijit Gosavi. Simulation-Based Optimization. Parametric Optimization Techniques and Reinforcement Learning. Springer (2003).

Martina Busetto. Matlab Exercises: eigenvalues and eigenvectors (2020).

Stefano Berrone. 2020-11-18-Note-11-28 [eigenvalues and eigenvectors] (2020).