Numerical Linear Algebra: 22nd December 2023

Project 3: PageRank implementations

Dafni Tziakouri

In the course of this project, we aim to develop two distinct algorithms for executing Page Rank. Both algorithms leverage the Power Method, with one algorithm utilizing matrix storage and the other eschewing it. It has been observed that computing the Page Rank vector incurs significant computational expenses, especially with large matrices.

To assess the performance of these algorithms, we will examine two sparse matrices, outlined below:

* Martr\_G (matrix): This function creates a matrix G representing the probability of jumping from one page to another.
* Matr\_L (nrows, ncols): This function creates a matrix L containing indices of non-zero elements in each column, serving as a workaround for not storing matrices.

Additionally, we generated two functions for each method—one to preserve the matrices and another to operate without saving them:

* PowerMethod\_Storing(G, m, Tol): This function implements the power method for computing the PageRank vector using the method of storing matrices.
* PowerMethod\_NotStoring( matrix, m, Tol): This function implements the power method without storing matrices, addressing memory limitations for large datasets.

To execute the functions mentioned above, users are prompted to enter a damping factor within the range [0, 1] for various exploratory purposes, with a suggested value of 0.15. Additionally, users are required to input tolerance within the range [1e-04, 1e-10], although it is advisable to utilize 1e-05 as the recommended tolerance value. In addition, the user can select the preferred method for each iteration.

Now, let's delve into the outcomes for each method across different damping factors and tolerance options, considering each method individually.

**Exercise1: Power method storing matrices**

Let’s imagine we have a webpage network with n webpages and a link matrix called G (which defines a direct graph). Then, we can define a PageRank score of the page k as:

= {Number of pages with a link to page}, and = {Number of outgoing links from page}.

We can rewrite the equality as a fixed-point equation with format, with being.

If the webpage does not contain any nodes with no outgoing link (dangling nodes), we can say that matrix A is column stochastic. In this case, . The eigenvector of eigenvalue 1 can be called PR vector in case of being unique.

But we want to code an algorithm which can handle not unique PR vectors for disconnected networks or column substochastic matrices A in case we have dangling nodes on the networks.

We will consider the following: , where is a damping factor (we consider ) and , where the vector given by

The matrix is column stochastic and has a unique PR vector.

In the following table, we will display the outcomes for each available option:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method 1: Power method with storing matrices | | | | | | |
| Damping | 0.15 | | 0.25 | | 0.50 | |
| Tolerance | 1e-05 | 1e-10 | 1e-05 | 1e-10 | 1e-05 | 1e-10 |
| Time Consumption ( in seconds) | | | | | | |
| Dataset | 26.665 | 27.044 | 26.339 | 26.885 | 26.547 | 27.153 |

When employing the Power Method with Matrix Storage, it becomes evident that the time consumption is not significantly influenced by the damping and tolerance values. The dataset values remain consistent even when altering the variables.

**Exercise 2: Power method not storing matrices**

When working with large datasets, most of the times we will have storage problems. We can solve that by iterating without storing the matrices.

To do so, we will consider the following:

Following the same procedure as before in table below, we will display the outcomes for each available option:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Method 2: Power method without storing matrices | | | | | | |
| Damping | 0.15 | | 0.25 | | 0.50 | |
| Tolerance | 1e-05 | 1e-10 | 1e-05 | 1e-10 | 1e-05 | 1e-10 |
| Time Consumption ( in seconds) | | | | | | |
| Dataset | 8.392 | 25.854 | 6.813 | 18.685 | 4.198 | 10.624 |

When matrices are not stored in the Power Method, a notable decrease in overall time consumption is observed. Moreover, significant variations arise when adjusting the Damping and Tolerance variables. For instance, with Tolerance set at 1e-05, the time consumption consistently proves to be less than when Tolerance is set at 1e-10. Conversely, a smaller Damping factor leads to higher time consumption, while progressively increasing the Damping factor results in decreased time consumption.

In conclusion, the choice of matrix storage in the Power Method has a discernible impact on time consumption. When matrices are stored, the damping and tolerance values exhibit minimal influence on the overall time required, with dataset values remaining consistent despite variable adjustments. On the other hand, opting not to store matrices results in a significant reduction in time consumption. Notably, variations in time consumption are evident when modifying the Damping and Tolerance variables, emphasizing their influence on the efficiency of the Power Method. Specifically, lower tolerance values contribute to reduced time consumption, while higher damping factors lead to increased efficiency, highlighting the nuanced interplay of these variables in optimizing the algorithm's performance.