Project 2 NLA: PageRank computations

Given a webpage network with n webpages and a link matrix G (defining a direct graph), we define the PageRank (PR) score x_k of the page k as

$$x_k = \sum_{j \in L_k} \frac{x_j}{n_j} \tag{1}$$

where

$$L_k = \{\text{webpages with link to page } k\},$$
$$n_j = \#\text{outgoing links from page } j,$$

The relation (1) can be rewritten as a fixed point equation $x = Ax$, being $A \in \mathbb{R}^{n \times n}$. If the web network does not contain dangling nodes (i.e. nodes with no outgoing links) then the matrix A is column stochastic. It follows that $1 \in \text{Spec}(A)$ in this case. If unique, the eigenvector of eigenvalue 1 is the so-called PR vector. Our goal is to investigate how to compute the PR vector.

Task 1: Solve the exercises 1,4 and 10 of Ref. [1].

There are two problems to take into account:

- For disconnected networks the PR vector is not unique.
- If the network has dangling nodes then the matrix A is column substochastic (and has no eigenvector of eigenvalue 1).

To address these problems one considers matrices of the form

$$M_m = (1 - m)A + mS,$$

where $0 \leq m \leq 1$ is a damping factor[1]. The matrix S is defined as follows. Consider $e = (1, \ldots, 1)^t$, and consider $z = (z_1, \ldots, z_n)^t$ to be the vector given by $z_j = m/n$ if the column j of the matrix A contains non-zero elements, $z_j = 1/n$ otherwise. Then $mS = ez^t$.

The matrix M_m is column stochastic and has a unique PR vector. Our goal is to compute this PR vector. We propose the following three strategies:

1. **Solving a linear system.** Consider the link matrix $G = (g_{ij})$, where $g_{ij} = 0$ or $g_{ij} = 1$ according to existence or not of link between the pages i and j. Define $c_j = \sum_i g_{ij}$ the out-degree of the page j. Writing $A = GD$ where G is the link matrix and $D = \text{diag}(d_{11}, \ldots, d_{nn})$ where $d_{jj} = 1/c_j$ if $c_j \neq 0$ and $d_{jj} = 0$ otherwise, one has

   $$M_m x = x \Leftrightarrow (Id - (1 - m)GD)x = \nu e$$

   for a suitable $\nu \in \mathbb{R}$. Taking $\nu = 1$ one reduces the PR computation to solve the system $(Id - (1 - m)GD)x = e$ and then scale x so that $\sum x_i = 1$. This can be implemented taking advantage of the sparse properties of the matrix of the system.

[1] We shall consider $m = 0.15$ in the numerical computations, see [2]

2. **Power method (adapted to PR computation).** The algorithm reduces to

   (a) Compute $M_m = (1-m)GD + S$.

   (b) Iterate $x_{k+1} = M_m x_k$ until $\|x_{k+1} - x_k\|_\infty <$ `tol`. Note the advantage of the sparse structure in this case, one has

   $$x_{k+1} = (1-m)GDx_k + ez^t x_k,$$

   and the right term reduces to multiply e by a real number while the left one exploits the sparse structure of G.

3. **Power method without storing the matrices.** The algorithm consists in

   (a) From the vectors that store the link matrix G obtain, for each $j = 1, ..., n$, the set of indices L_j corresponding to pages having a link with page j.

   (b) Compute the values c_j from the sets L_j.

   (c) Iterate $x_{k+1} = M_m x_k$ until $\|x_{k+1} - x_k\|_\infty <$ `tol` using the algorithm explained below.

Task 2: Implement the three previous algorithms to compute the PR vector. Use them for the examples in [1] to check they work properly.

Task 3: Download the file p2p-Gnutella30.mtx from the Sparse Matrix collection [3] Then use the previous methods to compute the PR vector. Report the results obtained. Investigate the role of `tol` in the computation and the components of the PR vector.

Report the results obtained. Check the order of convergence of the power method implemented, according to the theoretical expectations. Discuss (no implementation required) which are the possible strategies to deal with a larger webpage network (e.g. the `web-Google.mtx` file of google links, also available in [3]) according to the required resources in each implementation of the PR algorithm.

References

[1] K. Bryan and T. Leise. *The $ 25,000,000,000* Eigenvector: the Linear Algebra behind Google.*

[2] S. Brin and L. Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* Computer Networks and ISDN Systems. 30: 107117.

[3] T. Davis. *The University of Florida Sparse Matrix Collection.* `http://www.cise.ufl.edu/research/sparse/matrices/`

Due date: December 11th (at noon). The delivery must contain the implemented code and a brief discussion of the results. Please create a .tar.gz with the required files and upload it to the Campus Virtual.

How to perform an iterate of the power method without storing matrices?

Our goal is to compute the iterates $x_{k+1} = M_m x_k$, where $M_m = (1-m)A + mS$, without considering the matrix M_m. Below we denote M_m by $M = (M_{i,j})_{i,j=1,\dots,n} \in \mathbb{R}^{n\times n}$ and x_k by $x = (x_1, \dots x_n)^T$. One has

$$Mx = \begin{pmatrix} M_{1,1} & \dots & M_{1,n} \\ \vdots & & \vdots \\ Mn,1 & \dots & M_{n,n} \end{pmatrix} \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} M_{1,1}x_1 + \dots + M_{1,j}x_j + \dots + M_{1,n}x_n \\ \vdots \\ M_{n,1}x_1 + \dots + M_{n,j}x_j + \dots + M_{n,n}x_n \end{pmatrix}$$

If $c_j = 0$ then $d_{j,j} = 0$ and the jth column of A is a column of zeros (recall that $A = GD$ where $D = \mathrm{diag}(d_{11}, \dots, d_{nn})$). From this it follows that $M_{i,j} = 1/n$ for all $1 \le i \le n$. Then, the righthand part of Mx in the previous computation can be rewritten as

$$\begin{pmatrix} \displaystyle\sum_{j|c_j\neq 0} M_{1,j}x_j + \frac{1}{n}\sum_{j|c_j=0} x_j \\ \vdots \\ \displaystyle\sum_{j|c_j\neq 0} M_{n,j}x_j + \frac{1}{n}\sum_{j|c_j=0} x_j \end{pmatrix}$$

Consider j such that $c_j \neq 0$ and denote by $\tilde{A} = (1-m)A$. Then,

$$\tilde{A}_{i,j} = \begin{cases} 0 \text{ if } g_{i,j} = 0 \\ (1-m)/c_j \text{ if } g_{i,j} = 1 \end{cases}$$

Using that $g_{i,j} = 0$ if, and only if, $i \notin L_j$, the product Mx can be implemented as follows

```
xc=x
x=0
for j in ranfe (0,n):
    if(c[j]==0):
         x=x+xc[j]/n
    else:
         for i in L[j]:
             x[i]=x[i]+xc[j]/c[j]
x=(1-m)*x+m/n
```

Task 4: Explain why the previous code computes Mx.