PageRank implementations

The goal of this project is to code two different implementations of the PageRank algorithm.

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

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

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. In this case $1 \in \mathrm{Spec}(A)$. If unique, the eigenvector of eigenvalue 1 is the so-called PR vector. However, there are two problems to take into account:

1. For disconnected networks the PR vector is not unique.
2. 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

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

where

- $0 \leq m \leq 1$ is a damping factor. We shall consider $m = 0.15$ [1]
- $mS = ez^t$, where $e = (1, \ldots, 1)^t$ and $z = (z_1, \ldots, z_n)^t$ is the vector given by

$$z_j = \begin{cases} m/n \text{ if the column } j \text{ of the matrix } A \text{ contains non-zero elements} \\ 1/n \text{ otherwise} \end{cases}$$

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

We want hence to compute the PR vector of M_m. We shall consider the dataset p2p-Gnutella30.mtx from the Sparse Matrix collection `http://www.cise.ufl.edu/research/sparse/matrices/`

Let $G = (g_{ij})$ the link matrix, that is, g_{ij} is either 0 or 1 according to the existence or not of link between the pages i and j. Then $n_j = \sum_i g_{ij}$ is the out-degree of the page j. Let $D = \mathrm{diag}(d_{11}, \ldots, d_{nn})$ where $d_{jj} = 1/n_j$ if $n_j \neq 0$ and $d_{jj} = 0$ otherwise. Then $A = GD$.

Ex 1. Compute the PR vector of M_m using the power method (adapted to PR computation). The algorithm reduces to iterate $x_{k+1} = (1-m)GDx_k + ez^t x_k$ until $\|x_{k+1} - x_k\|_\infty < \texttt{tol}$.

---

[1] S. Brin and L. Page. *The Anatomy of a Large-Scale Hypertextual Web Search Engine.* Computer Networks and ISDN Systems. 30: 107–117.

Even if one can exploit the sparse structure of G in the implementation of the previous algorithm, for a large dataset the memory requirements easily exceed the available resources. An alternative could be to perform the iterates of the power method without storing the matrices:

1. 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.

2. Compute the values n_j as the lenght of the set L_j.

3. Iterate $x_{k+1} = M_m x_k$ until $\|x_{k+1} - x_k\|_\infty < \texttt{tol}$ using the idea explained below.

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 $M = (M_{i,j})_{i,j=1,\ldots,n} \in \mathbb{R}^{n\times n}$ and x_k by $x = (x_1, \ldots x_n)^T$. One has

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

If $n_j = 0$ then $d_{j,j} = 0$ and the jth column of A is a column of zeros (recall that $A = GD$ where $D = \text{diag}(d_{11}, \ldots, d_{nn})$). 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

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

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

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

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

Ex 2. Compute the PR vector of M_m using the power method without storing matrices.