PageRank algorithm

October, 14th 2021

Pierre-Jean Charpin & José Lagès

You have to hand in your TP report before ***December, 9th 2021****. Send the report and your* ***commented*** *Python code to the following email address: pj.charpin@femto-st.fr. The use of LaTeX is mandatory. Your results should be commented on and illustrated (if possible) with carefully produced figures. Your report should be written in such a way as to keep its weight as low as possible. For this purpose, you should use vector images for your illustrations (.pdf, .svg, .eps format) which will, in principle, be much lighter than the "usual" .jpg, .png, ... If you have any questions or remarks, please do not hesitate to send me an email at the above address.*

1 The "basics"

The aim of this section is to give a reminder, or to discover the Python language through simple examples that will make use of most of the tools needed for this TP.

1.1 Vectors and matrices

The *numpy* library is a very useful, even indispensable, module of python. It has many useful functions and allows you to create vectors and matrices (arrays of numbers).

Exercise : Define a matrix and a vector using *numpy* as follows :

```
1: import numpy as np
2: A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
3: b = np.array([-3, -2, -1])
```

Exercise : Do the matrix/vector product Ab with the help of the *dot* function from *numpy* and display the result.

1.2 Functions

The Python language offers the user the possibility to create functions whose purpose is to perform operations on variables given as input to these functions. The way to define a function is as follows (in this example, the function nameOfTheFunction takes 2 variables as arguments and sums them) :

```
1: def nameOfTheFunction(var1,var2)
2:     res = var1 + var2
3:     return(res)
```

Exercise : Define the function that to x corresponds $\sin(x)$. Apply it to the calculation of the sine of π and then to the sine of the vector $(0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}, 2\pi)$.

Any triangle can be characterised by its three vertices $(x_1; y_1)$, $(x_2; y_2)$ and $(x_3; y_3)$. The area of this triangle can be calculated using the following formula :

$$\mathrm{Area} = \frac{1}{2}|(x_1 - x_3)(y_2 - y_1) - (x_1 - x_2)(y_3 - y_1)| \tag{1}$$

Exercise : Write a function that returns the area of a triangle given the coordinates of its three vertices as arguments (for example [[0, 0], [0, 1], [1, 0]])

1.3 Plotting functions

Exercise : Import the module *matplotlib.pyplot* and use the *plot* function of this module to plot the graphs of the following functions. To plot the graph of a function f on an interval [a,b], you must first "discretize" it; that is to say, define a list of points $(x_i, f(x_i))$ which will allow the graph of the function to be approximated by a broken line. Of course, the more we increase the number of points, the closer the broken line is to the graph (in a certain sense). More concretely, we start by constructing a vector X which discretizes the interval [a,b], considering a set of equidistant points in [a,b]. To do this we can use the function *linspace* of the module *numpy*.

- $f(x) = \cos^2(x)$ with $x \in [0, 5]$
- On the same figure, the fonctions : $f(x) = \cos^2(x)$, $g(x) = \cos(2x)$ and $h(x) = \cos(x^2)$.

Consider using different symbols for the different graphical representations and include a legend.

1.4 Matrix diagonalization

The module *linalg* of *numpy* has the function *eig* to calculate the eigenvalues and eigenvectors of a square matrix.

Exercise : Diagonalize (find the eigenvalues and eigenvectors) the following M matrix and check your result using the website Wims.

$$\mathrm{M} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 16 & 0 & 0 & 0 \\ 0 & 0 & 9 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{2}$$

The function *argsort* in the module *numpy* is used to determine the indices of a list so as to sort the list in ascending order. For example, the function *argsort* applied to the list L=[3,5,1,8,4] will give the following result: [2,0,4,1,3] because the smallest element of L is 1 which is located in the third position, then 3 which is in the first position etc.

Exercise : Sort the list of eigenvalues of M in ascending order using the function *argsort*.

Exercise : What do the commands *print(M[2,:])* and *print(M[:,1])* do ?

1.5 Read/write files

The function *loadtxt* of the module *numpy* allows you to extract information present in a text file and to arrange it in a matrix. Conversely, the function *savetxt* allows you to write a variable present in your python environment into a text file.

Exercise : Load the file *"matrix.txt"* into a matrix Mat using the function *loadtxt*, calculate $\mathrm{Mat}^2 + 2\mathrm{Mat} - 3\mathrm{Id}$ and write your result in a text file *"res.txt"* (the numbers must be written in integer format).

2 PageRank : Introduction and general case

2.1 The PageRank vector

Networks are ubiquitous: social interactions, ie, persons related by acquaintances, can be modeled by social networks, the World Wide Web (WWW) is a collection of myriads of hyperlinked web-pages, billions of neurons communicate through electrical signals, ten thousand of genes interact with each other, economical sectors transform goods from other sectors and supplies goods to other sectors...

In many cases, (large amount of) data can be regarded as a network.[a] Eg, let us consider "Les Misérables" by V. Hugo, and let us consider the characters as the nodes of a network. We can think about many ways to relate each character to the others. A simple way is the following one: let us consider that if a character B is cited in the book just after a character A then a link $A - B$ is created between the two characters. Following this procedure for all the characters, we construct an undirected network. If we consider that A is cited before B, we can add this additional information by assigning a direction to the link which is now $A \to B$. Following this procedure for all the characters, we construct a directed network. If we count the number of times in the book that character B is cited just after A, it can serve to give a weight to the $A \to B$ link. Following this procedure for all the characters, we construct a directed and weighted network.

Once data is modeled by a network, a measure of centrality allows to determine how central or important is a node. For a social network, such a measure tells us which person is the most influential. For the WWW, it allows to rank the web-pages by relevance. For "Les Misérables", it tells which character plays the most central role in the story. Many measure of centrality exist. The most simple one is the degree centrality. The degree of a node is the number of links going to or going from this node. For degree centrality, the nodes of the network are ranked by their degree. According to this simple measure, the most (less) important node is the most (less) connected node. Note that the notion of importance is relative to the centrality measure we use. For two different centrality measures, the ranking of the nodes, and consequently their importance in the network, can be different.

Here, we will focus on the PageRank centrality which was at the heart of the Google search engine at its beginning in 1998 [1]. The associated algorithm is very powerful as it is able to rank billions of nodes in a reasonable number of computational steps. Let us consider a network with N nodes which are connected each other with ℓ links. Let us assign a number or a label to each node, eg, $1, 2, \ldots, N$. The network structure is encoded in the adjacency matrix A the elements of which are

$$A_{ij} = \begin{cases} 1 & \text{if } j \text{ points to } i \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

For an undirected network, the matrix A is symmetric, ie, $A_{ij} = A_{ji}$, $\forall (i,j)$. In their 1998 seminal paper [1], Brin and Page, co-founders of Google, described the PageRank algorithm without using the matrix formalism. They also omitted to talk about Markov chains [2], a class of stochastic models, introduced in 1906, to which the PageRank algorithm belongs. The adjacency matrix A is converted into a stochastic matrix S the elements of which are

$$S_{ij} = \begin{cases} A_{ij}/k_j^{\text{out}} & \text{if } k_j^{\text{out}} \neq 0 \\ 1/N & \text{otherwise.} \end{cases} \tag{4}$$

Here, $k_j^{\text{out}} = \sum_{i=1}^{N} A_{ij}$ is the out-degree of the jth node. Otherwise stated, k_j^{out} is the number of ways to leave the node j. By extension, we can also define the in-degree of the ith node as $k_i^{\text{in}} = \sum_{j=1}^{N} A_{ij}$. The value k_i^{in} gives the number of ways to jump to the ith node. We easily observe that for all column j, $\sum_{i=1}^{N} S_{ij} = 1$, ie, each column of the stochastic matrix S is normalized to unity. The stochastic matrix element S_{ij} is the transition probability from node j to node i. A random surfer, condemned to forever wander from node to node on the network, has a probability S_{ij} to jump to the node i if it is initially localized on the node j. If the number of ways out from the node j is k_j^{out}, then the random surfer can jump from node j to any of the adjacent nodes with the same probability $S_{ij} = 1/k_j^{\text{out}}$. If the node j is a dangling node, ie, there is no way out from node j, then one allows the random surfer to jump, with probability $1/N$, to any node of the network. Let us assume that the random surfer starts its journey from the node j_0. The vector $\mathbf{p}^{(1)} = S\mathbf{p}^{(0)}$, with $p_i^{(0)} = \delta_{ij_0}$, gives the probability distribution describing the localization of the random surfer after a jump, ie, an iteration of the stochastic process.[b] The ith component of the vector $\mathbf{p}^{(n)} = S^n\mathbf{p}^{(0)}$, ie, $p_i^{(n)}$, is the probability that the random surfer ends to the node i after n iterations.[c] The vector $\mathbf{p}^{(\infty)}$ gives the probability distribution after an infinite journey of the random surfer. This probability distribution is not unique as it depends on the initial probability distribution $\mathbf{p}^{(0)}$, ie,

[a] In mathematics, a network is called a graph. A node of a network is a vertex of a graph, and a link between two nodes of a network is an edge between two vertices of a graph. It is just a renaming.

[b] The Kronecker symbole δ_{ij} is equal to 1 if $i = j$ and is equal to 0 otherwise.

[c] Since the vector components of $\mathbf{p}^{(n)}$ are probabilities, we have $\left\|\mathbf{p}^{(n)}\right\|_1 = \sum_{i=1}^{N} p_i^{(n)} = 1$.

from where the random surfer starts its journey. In order to obtain an unique solution of the random walk on the network, we define the Perron-Frobenius operator G the elements of which are

$$G_{ij} = \alpha S_{ij} + (1-\alpha)\, v_i. \tag{5}$$

Here, the parameter $\alpha \in [0.5, 1[$ is the damping factor and $\mathbf{v}$ is a preferential vector. The equation (5) tells us that at each iteration, the random surfer follows, with the probability α, the structure of the network, or jump to any node, with the probability $(1-\alpha)$, according to the preferential vector $\mathbf{v}$. In the genuine article [1], the damping factor is taken as $\alpha = 0.85$ and the preferential vector components are $v_i = 1/N$, $\forall i$. The Perron-Frobenius operator G is called the Google matrix [3]. This damping factor allows the random surfer to be never stuck in a sub-network and, hence, only one solution exists $\mathbf{P}$ for the infinite journey probability distribution, $G^{\infty}\mathbf{p}^{(0)} = \mathbf{P}$ for all initial probability distribution $\mathbf{p}^{(0)}$. The vector $\mathbf{P}$ gives the steady state probability distribution of the stochastic process encoded by the stochastic matrix G since

$$G\mathbf{P} = \mathbf{P}. \tag{6}$$

The ith component of the PageRank vector $\mathbf{P}$ is proportional to the number of times the random surfer reach the node i. It is possible to define the PageRank index K. We assign the rank $K = 1$ to the node i_1 such as

$$P_{i_1} = \max_{i \in [1,\ldots,N]} \{P_i\},$$

the rank $K = 2$ to the node i_2 such as

$$P_{i_2} = \max_{i \in [1,\ldots,N]-\{i_1\}} \{P_i\},$$

the rank K to the node i_K such as

$$P_{i_K} = \max_{i \in [1,\ldots,N]-\{i_1,\ldots,i_K\}} \{P_i\},$$

and the rank $K = N$ to the node i_N such as

$$P_{i_N} = \min_{i \in [1,\ldots,N]} \{P_i\}.$$

The nodes of the network are then ranked by descending order of their values of the PageRank probabilities. The node to which $K = 1$ ($K = N$) is assigned is the most (less) important or central node of the network according to the PageRank centrality measure.

A possible algorithm for calculating the vector $\mathbf{P}$ is the power iteration algorithm whose pseudo-code is presented below 2.1. This algorithm is based on the previously stated result: "there is only one solution $\mathbf{P}$ for the probability distribution of the infinite journey, $G^{\infty}\mathbf{p}^{(0)} = \mathbf{P}$ for any initial probability distribution $\mathbf{p}^{(0)}$. "

Algorithm 1 Power iteration method

Data : $\mathrm{P}_{\mathrm{init}}$, ε, G^*
Result : CheiRank vector $\mathbf{P} = G^{\infty}\mathbf{p}^{(0)}$

```
while G*P − P ≥ ε do
    P = G*P
end
```

This algorithm is translated in the following way: as long as the product $\mathrm{G^*P}$ is greater than P to the nearest ε (ε being a real number as small as one wishes), one continues to iterate the matrix product.

PageRank calculation: direct diagonalization

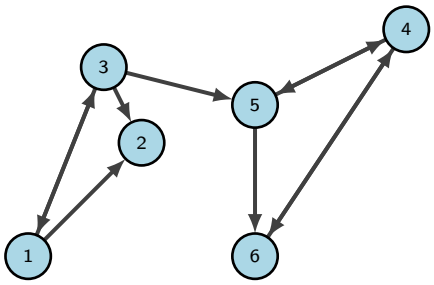

Figure 1: Network 1.

For the network shown above,

- classify the nodes according to PageRank centrality by solving the eigenvalue problem (6). We will take the original value $\alpha = 0.85$ and start from node 3,
- what happens to the PageRank probabilities and PageRank rankings when α varies in the range $[0.5, 1[$,
- compare PageRank centrality with degree centrality,
- make relevant figures highlighting your results

2.2 The CheiRank vector

Another vector is interesting to take into account, it is the CheiRank vector $\mathbf{P}^*$ constructed in the same way as the PageRank vector $\mathbf{P}$ but considering the matrix G^* constructed from A^t. In contrast to the PageRank vector which informs us about the probability of reaching a given node after an infinite number of moves, the CheiRank vector informs us about the level of **communication** of a node. The highest (low) CheiRank value therefore characterises the most (least) communicating node.

Still considering the previous network :

- rank the nodes according to CheiRank centrality, from the most communicating to the least communicating, by solving the power iteration problem (2.1), considering the G^* matrix
- comment on this ranking
- does the ranking evolve according to the arrival vector $\mathbf{P}^*$? Why ?
- plot, at each iteration, a histogram of the CheiRank vector as a function of the nodes (when executing your code, we should therefore see a small animation of the evolution of the CheiRank value as a function of the iteration)

3 PageRank : Pokémon, link them all !

The battle system in the video game Pokémon can be seen as a directed network. In this universe, pokémons are warrior animals that can be classified into well-defined types. These types are generally related to elements such as earth, water, fire, air. A pokémon can have different types of attacks. If an attack of type t_1 hits a pokémon of type t_2 , this attack can be boosted or even totally cancelled depending on the types involved.

Here is a summary table of the rules concerning weaknesses and strengths for the different types.

We then see a directed network whose nodes are the types. A link $\mathrm{t}_1 \to \mathrm{t}_2$ means that an attack of type t_1 is launched on a pokémon of type t_2 . The weight of this link defines the bonus or malus that the attack will suffer.

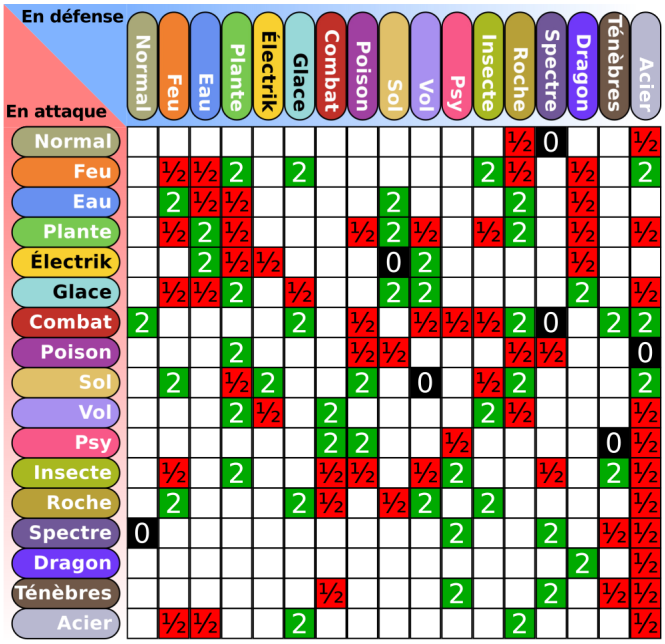

Figure 2: Summary of the impact of attacks according to their type. 0 means that the attack has no effect, 1/2 that the targeted type has medium resistance, 2 that the attack is super effective against the targeted type and the white boxes represent a normal resistance of 1.

Considering the network of pokemon types,

- What does the adjacency matrix A represent in a network ?
- same question for the stochastic matrix S.
- Write an executable and carefully commented out python script, taking as arguments the files *'nodes.txt'* and *'links.dat'* to determine the PageRank and CheiRank of the network associated with the pokémons. Be careful when filling in the adjacency matrix A! Use the definition 3 correctly.
- summarise your results in two text files generated by your script (one for PageRank and the other for CheiRank) with the following form :
  • First line : ε used and number of nodes in the network
  • Second line : empty line
  • Then a table with five columns (with separator) : Rank – ID – Type – PageRank/CheiRank (power iteration) – PageRank/Cheirank (diagonalization)
- For each ranking, give a meaning of the first and last node (what do PageRank and CheiRank represent, are they consistent with the table 2 ?) Open question.
- (Bonus) Using the python module *networkx*, draw the network corresponding to the problem, taking into account the weights of each of the nodes (each link between nodes can be of a different colour depending on the value of this link).

References

[1] S. Brin and L. Page, *The Anatomy of a Large-Scale Hypertextual Web Search Engine*. In: Seventh International World-Wide Web Conference (WWW 1998), April 14-18, 1998, Brisbane, Australia

[2] A.A. Markov, Rasprostranenie zakona bol'shih chisel na velichiny, zavisyaschie drug ot druga, Izvestiya Fiziko-matematicheskogo obschestva pri Kazanskom universitete, 2-ya seriya, 15, 135 (1906) (Extension of the limit theorems of probability theory to a sum of variables connected in a chain)

[3] A.N. Langville and C.D. Meyer, *Google's PageRank and Beyond: The Science of Search Engine Rankings*, Princeton University Press (2006)