PageRank algorithm

Procheta Sen

UNIVERSITY OF LIVERPOOL

PageRank algorithm: Markov chain perspective

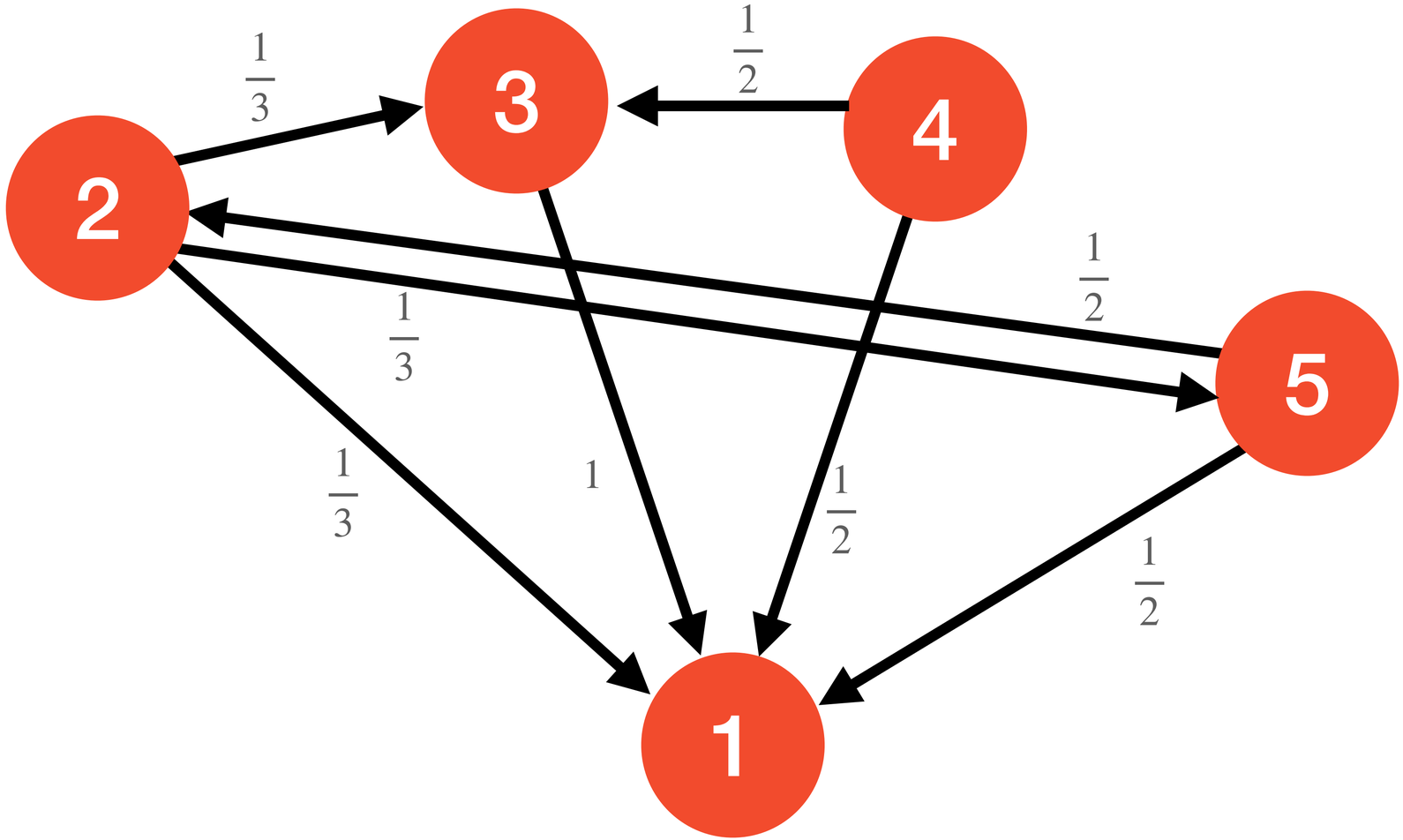
In the Markov chain model:

- each **node** (webpage) of the graph is a regarded as a **state**
- an **arcs** (hyperlink) is a **transition**, which leads from one state to another state
- at a fixed state all transitions are equally probable

This framework models a **random Web surfer**:

- an imaginary surfer is randomly clicking on links
- all links have an equal probability of getting clicked
- “back” button on the browser is not used and the surfer does not type in an URL
- after the surfer made many clicks, what is the probability that they end up on a particular webpage?

PageRank algorithm: Markov chain perspective



For vertex x, each transition from x to one of its neighbours is $\frac{1}{O_x}$

PageRank algorithm: Markov chain perspective

- Let $\overline{A}$ be the **state transition probability matrix**:
  $\overline{A}_{ij}$ is equal to the transition probability that the surfer in state (page) i will move to state j, i.e.

$$\overline{A}_{ij} = \frac{1}{O_i}, \text{ if } (i,j) \in E, \text{ and}$$
$$\overline{A}_{ij} = 0, \text{ otherwise}$$

- Let $\overline{P}_0 = (P_0(1), P_0(2), \ldots, P_0(n))^T$ be an initial probability distribution vector that a surfer is at each state (page).

PageRank algorithm: Markov chain perspective

Then we have

$\sum_{i=1}^{n} P_0(i) = 1$ — initially the surfer is in one of the states (pages)

$\sum_{j=1}^{n} \overline{A}_{ij} = 1$ — the surfer always makes a click when in state i

The latter equation is not quite true for some Web pages because they have no out-links. Such pages (nodes) are called **dangling pages** (nodes).

If matrix $\overline{A}$ satisfies that equation for every $i = 1,\ldots,n$, we say that $\overline{A}$ is the **stochastic matrix** of Markov chain.

PageRank algorithm: Markov chain perspective

Assume for the time being that $\overline{A}$ is a stochastic matrix.

A standard question about Markov chains:

Given the initial probability distribution $\overline{P}_0$ at the beginning, what is the probability that m steps/transitions later the Markov chain (random surfer) will be at each state j?

The probability that the random surfer is in state j after 1 transition can be computed as

$$P_1(j) = \sum_{i=1}^{n} \overline{A}_{ij}(1) P_0(i),$$

where $\overline{A}_{ij}(1)$ is the probability of going from i to j in 1 transition, and $\overline{A}_{ij}(1) = \overline{A}_{ij}$.

PageRank algorithm: Markov chain perspective

In the matrix form this can be written as

$$\overline{P}_1 = \overline{A}^T \overline{P}_0.$$

In general, the probability distribution after k steps/transitions is:

$$\overline{P}_k = \overline{A}^T \overline{P}_{k-1}$$

This looks very similar to our previous equation $\overline{P} = \overline{A}^T \overline{P}$!

PageRank algorithm: Markov chain perspective

By the Ergodic Theorem of Markov chains,

a finite Markov chain defined by **the stochastic transition matrix** A has a unique **stationary probability distribution** if A is **irreducible** and **aperiodic**.

The stationary probability distribution means that after a series of transitions $\overline{P}_k$ will converge to a steady-state probability vector $\overline{\Pi}$ regardless of the choice of the initial probability vector $\overline{P}_0$, i.e.

$$\lim_{k \to \infty} \overline{P}_k = \overline{\Pi}$$

PageRank algorithm: Markov chain perspective

When we reach the steady-state, we have $\overline{P}_k = \overline{P}_{k-1} = \overline{\Pi}$, and thus $\overline{\Pi} = \overline{A}^T\overline{\Pi}$.

That is $\overline{\Pi}$ is the **principal** eigenvector of $\overline{A}^T$ with eigenvalue 1.

(Known fact: A **stochastic matrix** always has an eigenvalue 1. All other eigenvalues are in absolute value smaller or equal to 1)

If we treat $\overline{\Pi}$ as the PageRank vector $\overline{P}$ we obtain our previous equation

$$\overline{P} = \overline{A}^T\overline{P}$$

PageRank algorithm: Markov chain perspective

We want our matrix $\overline{A}$ to satisfy the following 3 conditions:

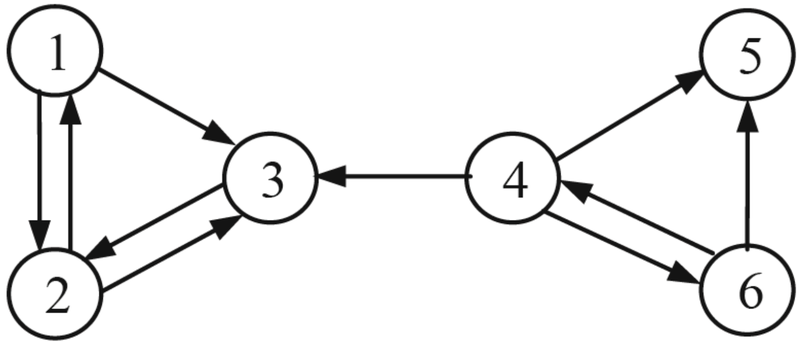
1. To be **stochastic**
2. To be **irreducible**
3. To be **aperiodic**

None of these conditions is satisfied for the real web graph.

However, we can modify matrix $\overline{A}$ so that it satisfies all three conditions!

Modification I: make $\overline{A}$ **stochastic**

For every dangling page i, add a complete set of outgoing links. The transition probability of going from i to every page is $1/n$



Hyperlink graph

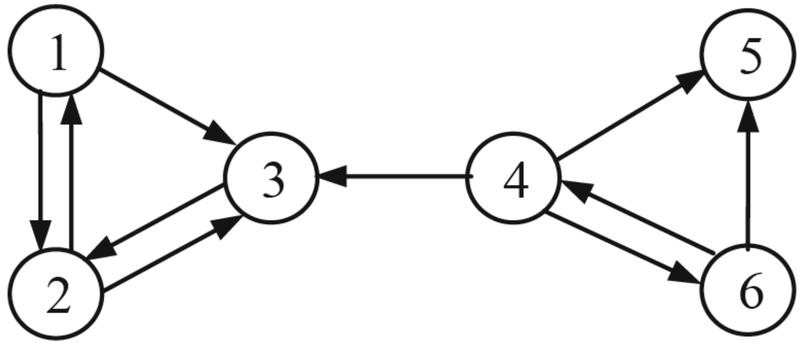
$$\overline{A} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}.$$

Transition probability matrix

5 is a dangling page, and hence $\overline{A}$ is **not** a stochastic matrix

Modification I: make $\overline{A}$ **stochastic**

For every dangling page i, add a complete set of outgoing links. The transition probability of going from i to every page is $1/n$



Hyperlink graph

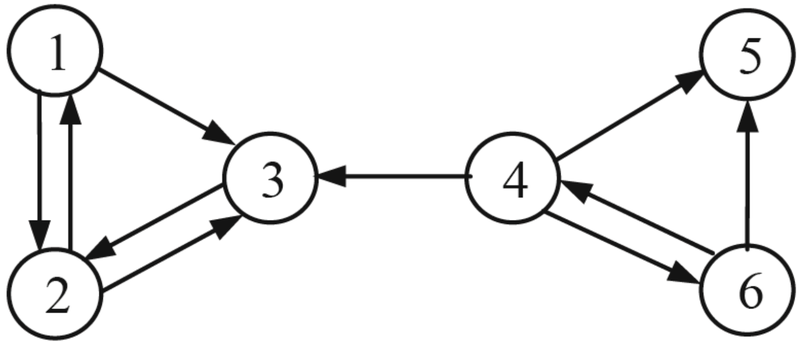
$$\overline{A} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}.$$

Transition probability matrix

5 is not dangling anymore, and $\overline{A}$ becomes a stochastic matrix

Modification II: make $\overline{A}$ **irreducible** and **aperiodic**

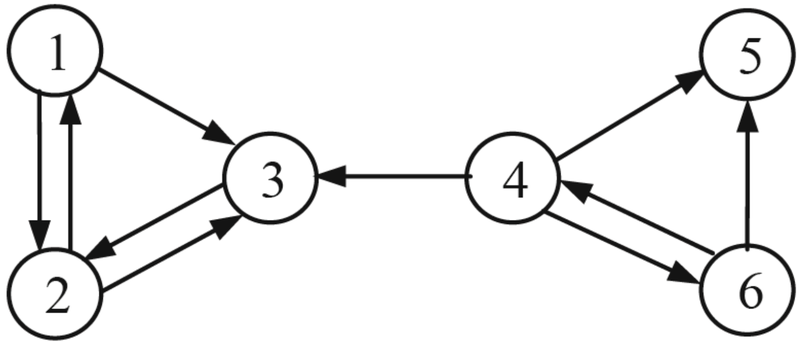
Add a link from each page to every page and give each such link a small transition probability controlled by a parameter d.



$$\overline{A} = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 1/3 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 \\ 0 & 0 & 0 & 1/2 & 1/2 & 0 \end{pmatrix}.$$

Modification II: make $\overline{A}$ **irreducible** and **aperiodic**

Add a link from each page to every page and give each such link a small transition probability controlled by a parameter d.



$$(1-d)\frac{\overline{E}}{n}+d\overline{A}^T = \begin{pmatrix} 1/60 & 7/15 & 1/60 & 1/60 & 1/6 & 1/60 \\ 7/15 & 1/60 & 11/12 & 1/60 & 1/6 & 1/60 \\ 7/15 & 7/15 & 1/60 & 19/60 & 1/6 & 1/60 \\ 1/60 & 1/60 & 1/60 & 1/60 & 1/6 & 7/15 \\ 1/60 & 1/60 & 1/60 & 19/60 & 1/6 & 7/15 \\ 1/60 & 1/60 & 1/60 & 19/60 & 1/6 & 1/60 \end{pmatrix}$$

where d is a number between 0 and 1, and $\overline{E}$ is a $n \times n$ square matrix of all 1's.

$(1-d)\frac{\overline{E}}{n}+d\overline{A}^T$ is **stochastic** (but transposed), **irreducible**, and **aperiodic**

Apply the power iteration algorithm to the modified $\overline{A}$

We can now apply the **power iteration algorithm** to the modified matrix (that now satisfies the 3 conditions!).

The algorithm will compute the **principal** eigenvector for the matrix.

The resulting vector is the PageRank vector $\overline{P}$. This leads to the PageRank algorithm.

PageRank algorithm

PageRank(An n-vertex graph without dangling vertices: $G = (V, E)$; damping factor: d; tolerance ε)

1: $P_0(i) = 1/n$ `for all` $i = 1, \ldots, n$; // initial probability distribution (i.e. initial PageRank values)

2: $k = 1$

2: **repeat**

3: $P_k(i) = \frac{1-d}{n} + d \cdot \sum_{(x,i) \in E} \frac{P_{k-1}(x)}{O_x}$, `for all` $i = 1, \ldots, n$; // update PageRank values

4: $k = k + 1$

5: **until** $||\overline{P}_k - \overline{P}_{k-1}|| \leq \varepsilon$

9: **return** $\overline{P}_k = (P_k(1), P_k(2), \ldots, P_k(n))$ // the final PageRank values

Remarks

- In practice, a typical value for the damping factor d is 0.85
- PageRank algorithm can be applied to any graph, not limiting to web graph, to induce a ranking for the vertices
- PageRank algorithm is one of many algorithms that are based on the idea of random walks in a graph
- Since we are only interested in the ranking of the pages, the actual convergence may not be necessary, and the algorithm can be terminated after some maximum number of iterations even before achieving the tolerance ε
- In [1], it is reported that on a database of **322 million links** the algorithm converges to an acceptable tolerance in roughly **52 iterations**.

[1] Brin, S. and P. Lawrence. The anatomy of a large-scale hypertextual web search engine. Computer Networks, 1998, 30(1-7): p. 107-117.