Math 345 - Notes
PageRank algorithm
December 5, 2018

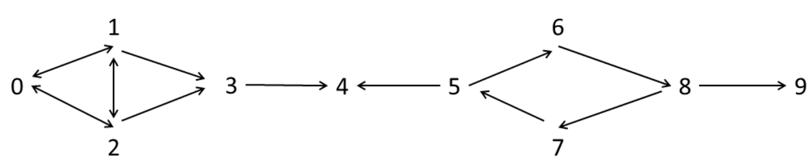
PageRank Algorithm

The Page rank algorithm was the initial Google webpage ranking algorithm. It assumes 100% page rank in a *directed* network. The algorithm computes the probability that a person clicking at random on links will arrive at a particular page. The probability, at any step, that the person will continue is given by the damping factor $\alpha \in (0,1)$.
Suppose there are N nodes in the network.

Algorithm:

- Let $\alpha \in (0,1)$ be the damping factor.
- Use discrete uniform distribution to initialize the rank of each data point. That is, at time 0, each node is assigned rank $\frac{1}{N}$.
- At each step, α of a node's rank is evenly distributed among its outgoing links and the remainder of its rank gets distributed to all the nodes (including itself).
- Iterate until the resulting rank vector stabilizes.

For our example with 10 nodes, let's add directions to edges:



Let $\alpha = 0.84$ and initialize the nodes with rank $r_0(i) = \frac{1}{10}$ for $0 \leq i \leq 9$.
Let's look at step $t+1$. Each node will update to rank of at least

$$\sum_{i=1}^{N} \frac{(1-\alpha)r_t(i)}{N} = \frac{1-\alpha}{N} = .016$$

$$
\begin{aligned}
r_{t+1}(0) &= \frac{\alpha}{3} r_t(1) + \frac{\alpha}{3} r_t(2) + (.016) \\
r_{t+1}(1) &= \frac{\alpha}{2} r_t(0) + \frac{\alpha}{3} r_t(2) + (.016) \\
r_{t+1}(2) &= \frac{\alpha}{2} r_t(0) + \frac{\alpha}{3} r_t(1) + (.016) \\
r_{t+1}(3) &= \frac{\alpha}{3} r_t(1) + \frac{\alpha}{3} r_t(2) + (.016) \\
r_{t+1}(4) &= \frac{\alpha}{1} r_t(3) + \frac{\alpha}{2} r_t(5) + \frac{\alpha}{1} r_t(4) + (.016) \\
r_{t+1}(5) &= \frac{\alpha}{1} r_t(7) + (.016) \\
r_{t+1}(6) &= \frac{\alpha}{2} r_t(5) + (.016) \\
r_{t+1}(7) &= \frac{\alpha}{2} r_t(8) + (.016) \\
r_{t+1}(8) &= \frac{\alpha}{1} r_t(6) + (.016) \\
r_{t+1}(9) &= \frac{\alpha}{2} r_t(8) + \frac{\alpha}{1} r_t(9) + (.016)
\end{aligned}
$$

Note that the ranks add up to 1:

$$
\sum_{i=0}^{9} r_{t+1}(i) = \left[\alpha \sum_{i=0}^{9} r_t(i) \right] + (1 - \alpha) = 1
$$

For example, at step 1,

$r_1(0) = .028 + .028 + .016 = .072$

$r_1(1) = .042 + .028 + .016 = .086$

$r_1(2) = .042 + .028 + .016 = .086$

$r_1(3) = .028 + .028 + .016 = .072$

$r_1(4) = .084 + .042 + .016 + .084 = .226$

$r_1(5) = .084 + .016 = .1$

$r_1(6) = .042 + .016 = .058$

$r_1(7) = .042 + .016 = .058$

$r_1(8) = .084 + .016 = .1$

$r_1(9) = .042 + .016 + .084 = .142$

The algorithm runs until there is no change in the ranks.

So what are we really finding? If you have encountered Markov chains, then we can interpret the network as a Markov chain with transitions given by weights depending on α. Looking for stability in the network translates to finding a stationary distribution for the Markov chain. For our example, the transition matrix associated with the Markov chain is

$$P = \begin{bmatrix} .016 & .436 & .436 & .016 & .016 & .016 & .016 & .016 & .016 & .016 \\ .296 & .016 & .296 & .296 & .016 & .016 & .016 & .016 & .016 & .016 \\ .296 & .296 & .016 & .296 & .016 & .016 & .016 & .016 & .016 & .016 \\ .016 & .016 & .016 & .016 & .856 & .016 & .016 & .016 & .016 & .016 \\ .016 & .016 & .016 & .016 & .856 & .016 & .016 & .016 & .016 & .016 \\ .016 & .016 & .016 & .016 & .436 & .016 & .436 & .016 & .016 & .016 \\ .016 & .016 & .016 & .016 & .016 & .016 & .016 & .016 & .856 & .016 \\ .016 & .016 & .016 & .016 & .016 & .856 & .016 & .016 & .016 & .016 \\ .016 & .016 & .016 & .016 & .016 & .016 & .016 & .436 & .016 & .436 \\ .016 & .016 & .016 & .016 & .016 & .016 & .016 & .016 & .016 & .856 \end{bmatrix}$$

P is a stochastic matrix, meaning that each row adds up to 1. In the language of Markov chains, we are looking for a unit length row vector π such that

$$\pi P = \pi.$$

If you would rather work with column vectors, then transpose both sides of the equation and find a unit length column vector $\mathbf{v} = \pi^T$ such that

$$P^T \mathbf{v} = \mathbf{v}.$$

This means that $\mathbf{v}$ is the eigenvector of P^T associated to the largest eigenvalue 1 (this is the largest eigenvalue due to the fact that P is stochastic.) We need to ensure that $\mathbf{v}$ is of length one, so we might have to rescale if we use a calculator! Using computational software, or a calculator, we can find that

$$\mathbf{v} = \begin{bmatrix} 0.042244 \\ 0.046865 \\ 0.046865 \\ 0.042244 \\ 0.441189 \\ 0.045488 \\ 0.035105 \\ 0.035105 \\ 0.045488 \\ 0.219407 \end{bmatrix}$$

Another way to find π is to look at the rows of $\lim_{n\to\infty} P^n$. Under certain circumstances, each row of this limiting matrix will be π, such as in our example.

On the other hand, the PageRank Algorithm should also converge to this probability distribution vector, which we computed here exactly.

Remark: Note the similarity with the eigenvector centrality discussed last lecture. The difference is that here we are working in a directed graph and only α of a node's weight is distributed to its (directed) neighbors.