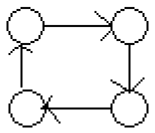
Random Walks on Directed Graphs

CS 4850: Mathematical Foundations for the Information Age
Lecture 25 : March 23, 2009
Scribes: Sean Sullivan (sps27), Hyun Park (hp256)

What properties must the directed graph have?

Strongly Connected : If the graph isn't strongly connected, the random walk might get stuck forever in a subset of the nodes or encounter a node with no outlinks. This is avoided by adding a fictitious node called "restart" that all other nodes point to. "restart" also has edges back to each of the original nodes. In the random walk, take the restart edge with probability 0.15 at each step.

Aperiodic : The greatest common denominator of all cycle lengths is 1. If this isn't the case, the random walk might not converge to a steady-state set of node probabilities.



A random walk on the graph above will not converge. If another cycle of length 3 were added, then the problem would go away.

There is similar terminology used in the literature for Markov Processes –

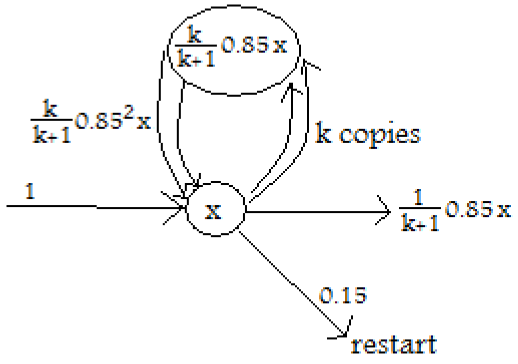
Persistent : If you ever reach a vertex, the walk will return eventually. (this means the walk is in a strongly connected component containing no outgoing edges)

Irreducible : The graph is just 1 single strongly connected component

Ergodic : A vertex is ergodic if it is aperiodic and persistent. A graph is ergodic if all of its vertices are ergodic.

Manipulating a graph to increase PageRank

Consider a vertex that has 1 incoming edge and 1 outgoing edge (and also the edge to restart). We'll say the incoming edge has weight 1 on it, so the PageRank of the vertex is 1. How much could this PageRank be increased by creating k dummy webpages that are linked to and from the existing vertex?



Now we just set up an equation and solve for the PageRank:

$$x = 1 + \frac{k}{k+1} 0.85^2 x$$

$$x = \frac{k+1}{k(1-0.85^2)+1}$$

So as k gets large, how much does this help?

$$\lim_{k\to\infty} x = \frac{1}{1-0.85^2} \cong 3.6$$

Computing PageRank

Given an adjacency matrix A, normalize the rows so that the entries in each row sum to 1.

We then wish to solve:

$p^T = p^T A\,(1-\alpha) + \alpha\, r^T$, where r is the restart vector, normally all entries are 1/n

This solves to:

$p = \alpha(I - (1-\alpha)A^T)^{-1} r$

Finding Spam Vertices

We might be able to identify spam vertices by comparing their hitting time to the return time (note: return time = 1/PageRank). If the return time is much lower than the hitting time, it might be a spam vertex with many short cycles.

Compute the restart time: At each node, there is a probability 0.15 of a restart. This means that the expected time between restarts is 1/0.15 = 6.67.

What kind of bounds can be get relating the return time to the hitting time?

First, the return time must be at most the restart time + hitting time, because you can return to the vertex by restarting and then going through the hitting process all over again.

The fastest possible return would be if there was a cycle of length 2 containing the vertex. So we get a lower bound on the return time:

$$Return\ time \geq 2(1-\alpha)^2 + (\alpha + (1-\alpha)\alpha) \times (hitting\ time)$$

At each step, we have a probability $\alpha = 0.15$ of restarting, so the 2 possibilities for the length 2 cycle are either the return time is 2 if both hops succeed, or the return time is the hitting time if one of the 2 hops failed.

Combining these 2 inequalities, we find:

$1.45 + 0.278 \times E(hitting\ time) \leq E(return\ time) \leq 6.67 + E(hitting\ time)$

If the return time is at the lower end of this range, then we become suspicious about the vertex potentially be spam.