Pagerank

September 24, 2012

Pagerank

Consider an finite, directed multigraph $G = (V, E)$ without self-loops, as in Fig. 1. We will understand G as the hyperlink structure of the web pages described by V; an arc from vertex u to vertex v describes a hyperlink from page u to page v.

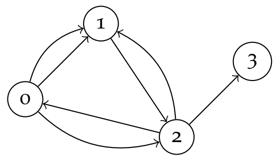

Figure 1: A directed multigraph.

The *random surfer* model is a stochastic process that aims to rank the relevance of a these pages. The states of this process are the vertices V. With good probability α, the process picks an outgoing edge at random and moves to that page. If there are no outgoing edges, the surfer picks a random page from V instead. (Note that outgoing edges are counted with multiplicities, so from vertex 0 in the example, the chance of going to 1 is twice that of going to 2.) Alternatively, with probability $(1 - \alpha)$, the surfer becomes bored and moves to a random page from V instead. The probability α is called the *damping factor*; a typically value that works well for web pages is $\alpha = \frac{85}{100}$.

This process is a finite, irreducible, and ergodic Markov chain. Its stationary distribution describes the *page rank* of each vertex. The contribution of Serge Brin and Larry Page was to realise that this value gives a good measure of the relevance of a web page, which is the main idea behind the search engine Google.

Files

Vertex names are integers $V = \{0, \dots, n-1\}$. Input files contain $|V|$, followed by u and v for each $(u, v) \in E$. The files are in the data directory are:

```
4
0 1   0 1   0 2
1 2
2 0   2 1
2 3
```

Figure 2: Input file for the graph in Fig. 1.

three.txt The 4-vertex graph from Fig. 1.

tiny.txt The 5-vertex graph from §1.6 from Sedgewick and Wayne.[1]

medium.txt The 50-vertex graph *ibid.*

wikipedia.txt The 11-vertex graph from Wikipedia's PageRank article.[2]

p2p-Gnutella08-mod.txt A 6301-vertex graph describing a file sharing network. Vertices represent hosts in the Gnutella network topology and edges represent connections between the Gnutella host, collected 8 August 2002.[3]

[1] R. Sedgewick and K. Wayne, *Programming in Java: An Interdisciplinary Approach*, Addison Wesley, 2007.

[2] "PageRank." Wikipedia, The Free Encyclopedia. Wikimedia Foundation, Inc. Accessed 16 Sep 2012.

[3] Modified from the Stanford University SNAP library, original file at snap.stanford.edu/data/p2p-Gnutella08.html. Sources: J. Leskovec, J. Kleinberg and C. Faloutsos. *Graph Evolution: Densification and Shrinking Diameters*. ACM Transactions on Knowledge Discovery from Data (ACM TKDD), 1(1), 2007. M. Ripeanu and I. Foster and A. Iamnitchi. *Mapping the Gnutella Network: Properties of Large-Scale Peer-to-Peer Systems and Implications for System Design*. IEEE Internet Computing Journal, 2002.

Deliverables

1. Implement a simulation of the random surfer model. Start in vertex 0 and follow the rules for a given number of iterations read

from the command line. Count the number of times each vertex is visited and print the relative frequencies.

2. Solve the exact same problem using linear algebra instead of simulation. That is, construct the transition probability matrix P and compute pP^r for some sufficiently high r (read from the command line) and some initial vector p of your choice. In fact, we're approximating the dominant left eigenvector, i.e., a vector satisfying $pP = p$. There are (at least) 3 ways of computing pP^r:

   (a) Let $p_0 = p$. For each $i = 1, \ldots, m$ let $p_i = p_{i-1}P$. Return p_r.

   (b) Compute $Q = P^r = P \cdot P \cdots P$ using $r - 1$ matrix products. Return pQ.

   (c) Compute P^r by iterated squaring. Assume r is a power of 2. Set $Q_0 = P$ and for $i = 1, \ldots, \log r$ compute $Q_i = Q_{i-1}^2$. Return $pQ_{\log r}$.

   Pick the one you think is fastest or easiest to implement for the small instances.

   However, to attack an instance of nontrivial size, you need to exploit the structure of the transition matrix. (Unless you like waiting a lot.) Let A denote the adjacency matrix of G. Then we have the hyperlink matrix H defined as $H_{ij} = A_{ij} / \deg(i)$. Let D denote the matrix where $D_{ij} = 0$ if $\deg(i) > 0$ and $D_{ij} = 1/n$ if $\deg(i) = 0$. Then,

   $$P = \alpha(H + D) + \frac{1-\alpha}{n}\mathbf{1},$$

   where $\mathbf{1}$ is the $|V| \times |V|$ all-1s matrix. In particular,

   $$pP = \alpha pH + \alpha pD + \frac{1-\alpha}{n}p\mathbf{1}.$$

   All three of these vector–matrix products are simpler to compute: all columns in D are identical, all columns in $\mathbf{1}$ are identical, and H is sparse (so you don't store is as a square matrix of size $O(|V|^2)$; just as a list of $O(|E|)$ nonzero values). Oh, and if you really just wrote code to compute the dot product between p and an all-1s vector, this is a good time for a break.

3. Fill out the report.

Pagerank Lab Report

by Mats Rydberg and Martin Larsson.

Transition probabilities

The transition matrix for the graph described in three.txt is[4]

[4] Fill in the right values. Set $\alpha = \frac{85}{100}$.

$$P = \begin{pmatrix} 0.0375 & 0.6042 & 0.3208 & 0.0375 \\ 0.0375 & 0.0375 & 0.8875 & 0.0375 \\ 0.3208 & 0.3208 & 0.0375 & 0.3208 \\ 0.25 & 0.25 & 0.25 & 0.25 \end{pmatrix},$$

and its 10th power is

$$P^{10} = \begin{pmatrix} 0.1784 & 0.2795 & 0.3635 & 0.1784 \\ 0.1784 & 0.2795 & 0.3635 & 0.1784 \\ 0.1784 & 0.2795 & 0.3635 & 0.1784 \\ 0.1784 & 0.2795 & 0.3635 & 0.1784 \end{pmatrix}$$

The transition matrix P can be broken down into $P = \alpha(H + D) + \frac{1-\alpha}{n}\mathbf{1}$, where

$$H = \begin{pmatrix} 0 & \frac{2}{3} & \frac{1}{3} & 0 \\ 0 & 0 & 1 & 0 \\ \frac{1}{3} & \frac{1}{3} & 0 & \frac{1}{3} \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

and

$$D = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{4} & \frac{1}{4} & \frac{1}{4} & \frac{1}{4} \end{pmatrix}$$

.

Results

The following table gives the top hits, i.e., the 10 first vertices of each graph sorted by page rank, using $\alpha = \frac{85}{100}$.

three.txt	2 (36.4%)	1 (28.0%)	0 (17.8%)	3 (17.8%)	
tiny.txt	0 (27.0%)	1 (26.0%)	3 (24.6%)	2 (15.0%)	4 (7.4%)
medium.txt	2 (21.7%)	1 (13.9%)	3 (12.3%)	4 (8.8%)	9 (1.0%)
	0 (0.9%)	5 (0.9%)	6 (0.9%)	7 (0.9%)	8 (0.9%)
wikipedia.txt	1 (40.4%)	2 (35.9%)	4 (6.7%)	3 (3.4%)	5 (3.4%)
	0 (2.9%)	6 (1.5%)	7 (1.5%)	8 (1.5%)	9 (1.5%)
p2p-Gnutella08-mod.txt	1 (0.248%)	2 (0.205%)	3 (0.170%)	4 (0.138%)	5 (0.107%)
	6 (0.050%)	8 (0.042%)	7 (0.041%)	9 (0.036%)	1769 (0.016%)

The following table gives the number of random walk steps and (scalar) multiplications needed for each graph until the results were stable to within 2 decimal places.

Graph	# transitions	# multiplications
three.txt		5
tiny.txt		9
medium.txt		10
wikipedia.txt		26
p2p-Gnutella08-mod.txt		

Optional

Build a time machine, fly back to the early 1990s. Start a search engine company based on this idea.

Perspective

For more thorough introduction to the mathematics behind this model, see David Austin, *How Google Finds Your Needle in the Web's Haystack*, American Mathematical Society Feature Column, 2006.[5]

The original paper is Sergey Brin, Lawrence Page, *The anatomy of a large-scale hypertextual Web search engine*[6], which also mentions a bit about the data structure used for storing web page content. A different model for establishing web page relevance was established by Kleinberg around the same time as PageRank.[7]

[5] www.ams.org/samplings/feature-column/fcarc-pagerank, retrieved 20 Sep 2012.

[6] Computer Networks and ISDN Systems, 33: 107-17, 1998. infolab.stanford.edu/pub/papers/google.pdf

[7] Kleinberg, Jon (1999). *Authoritative sources in a hyperlinked environment*. Journal of the ACM 46 (5): 604–632. doi:10.1145/324133.324140.