Second Note on Google's PageRank

References: Experiments with MATLAB by Cleve Moler, the paper from class, and previous notes.

Imagine surfing the Web, hopping from page to page by randomly choosing an outgoing link from one page to get to the next. This can lead to dead ends at pages with no outgoing links, or cycles around interconnected pages. So, a certain fraction of the time, simply choose a random page from the Web. This random walk is known as a Markov chain or *Markov process*.

The limiting probability that an infinitely dedicated random surfer visits any particular page is its PageRank.

Setup

- As before, let $\mathbf{A}$ be the *link* matrix. That is, $a_{ij} = 1$ if there is a link *to* page i from page j and 0 otherwise. $\mathbf{A}$ can be huge, but is very sparse.

- Let r_i denote the i-th row of $\mathbf{A}$, which shows the *back*links to page i: $r_i = \sum_j a_{ij}$ is the *in-degree* of the i-th page.

  Let c_j denote the j-th column of $\mathbf{A}$, which shows the links on the j-th page: $c_j = \sum_i a_{ij}$ is the *out-degree* of the j-th page.

  Scale $\mathbf{A}$ as follows: divide each column j by page j's out-degree, c_j. That is, the i, jth entry in the updated $\mathbf{A}$ is now

  $$a_{ij} = a_{ij}/c_j$$

  Now $\mathbf{A}$ is the same matrix we had at the end of the previous note.

- Let p be the probability that the random walk follows a link. A typical value is $p = 0.85$. Then $1 - p$ is the probability that some arbitrary page is chosen and $(1 - p)/n$ is the probability that a particular random page is chosen.

- As before, let $\mathbf{S}$ denote the $n \times n$ matrix with all entries $1/n$ and we will replace $\mathbf{A}$ with

  $$\mathbf{M} = p\mathbf{A} + (1 - p)\mathbf{S}$$

  where $0 \leq p \leq 1$. Thus $\mathbf{M}$ is the weighted average of $\mathbf{A}$ and $\mathbf{S}$.

- All in all, to end up with $\mathbf{M}$, we've made the following updates to the original link matrix $\mathbf{A}$:

  $$m_{ij} = \begin{cases} p\ a_{ij}/c_j + (1 - p)/n & : c_j \neq 0 \\ 1/n & : c_j = 0 \end{cases}$$

Thus $\mathbf{M}$ comes from scaling the link matrix by its column sums. The j-th column is the probability of jumping from the j-th page to the other pages on the Web. If the j-th page is a dead-end (has no out-links) then we assign a uniform probability of $1/n$ to all the elements in its column. Also note that most of the elements of $\mathbf{M}$ are $1/n$.

- Note that for any column-stochastic matrix $\mathbf{A}$, $\mathbf{M}$ is also column-stochastic. $\mathbf{M}$ is called the *transition probability matrix* of the Markov chain.
- As before, $V_1(\mathbf{M})$, the eigenspace corresponding to the eigenvalue 1 for matrix $\mathbf{M}$, can be shown to be 1-dimensional when $0 \leq p \leq 1$.
- Therefore $\mathbf{M}$ can be used to compute the unambiguous importance scores. Now we'll see how to compute Google PageRank in `MATLAB`.

Finding PageRank in `MATLAB`

- As before, a unique real solution to $\mathbf{M}x = x$ exists and it can be scaled so that $\sum x_i = 1$ to obtain the ranking.
- For small n, an easy way to compute the solution to the singular, homogeneous linear system $(\mathbf{A} - \mathbf{I})x = \mathbf{0}$ is to start with some approximate solution and to simply repeat the assignment statement `x = A*x` until successive vectors agree to within a specified tolerance.[1]
- The next section contains the `MATLAB` statements that implement this approach.

[1] The best way to compute PageRank in `MATLAB` is to take advantage of the particular structure of the Markov matrix. Write the transition matrix as:

$$\mathbf{M} = p\mathbf{A}\mathbf{D} + \mathbf{e}\mathbf{z}^T$$

where $\mathbf{D}$ is the diagonal matrix formed from the reciprocals of out-degrees,

$$d_{jj} = \begin{cases} 1/c_j & : c_j \neq 0 \\ 0 & : c_j = 0 \end{cases}$$

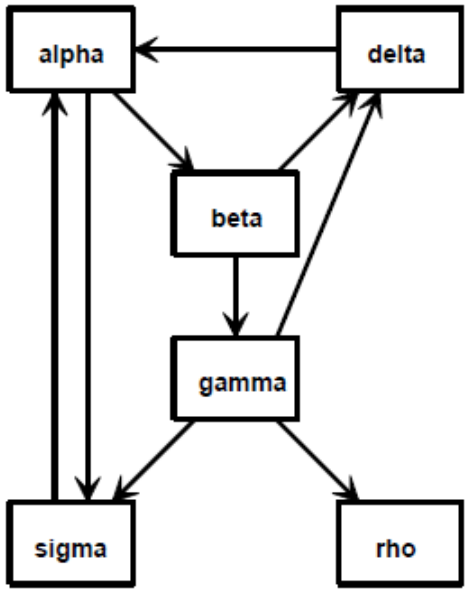
and $\mathbf{e}$ is the n-vector of all ones, and $\mathbf{z}$ is the vector with components

$$z_j = \begin{cases} (1-p)/n & : c_j \neq 0 \\ 1/n & : c_j = 0 \end{cases}$$

The rank-one matrix $\mathbf{e}\mathbf{z}^T$ accounts for the random choices of Web pages that do not follow links.

Another Tiny Web Example

- Consider the following example with $n = 6$ instead $n =$ a few hundred billion web pages, interconnected as shown:



- Note that while surfing this web the random surfer may end up on page `rho`, which contains no out-links. When at page `rho`, the surfer chooses any given page (including `rho`) with probability 1/6 and moves to it next.

- The following `MATLAB` statements (from 'Experiments with `MATLAB`' by Cleve Moler, slightly modified) can be used to create the sparse connectivity matrix and compute the ranking:

```
%% Sparse matrices
    n = 6
    i = [2  6  3  4  4  5  6  1  1]
    j = [1  1  2  2  3  3  3  4  6]
    A = sparse(i,j,1,n,n)
    %spy(A)
    %full(A)

%% Page Rank
    p = 0.85;
    delta = (1-p)/n;
    c = sum(A,1);
    k = find(c~=0);
    D = sparse(k,k,1./c(k),n,n);
    e = ones(n,1);j
    I = speye(n,n);
    x = (I - p*A*D)\e;
```

```
x = x/sum(x)
bar(x)
title('Ranking for the Tiny Web')
```

- The result: