# Math 2270-1

# Notes of 10/29/19

# The Google Page Rank Algorithm

- Linear Algebra is as essential and central in application as is Calculus.
- In spite of that fact we have not looked much at specific examples of applications.
- However, today let's look at one major application that everybody is familiar with.
- We'll discuss the basic idea of the the original Google Page Rank Algorithm (GPRA)
- Of course, nowadays a Google search uses vastly more complex ideas and concepts.
- A great introduction (no kidding) to the GPRA is the US Patent No. 6,285,991, "Method for Node Ranking in a Linked Database", by Larry Page, dated September 4, 2001.
- This is the only patent I've ever looked at.
- Larry Page, of course, is one of the founders of Google, and he made billions of Dollars starting with the ideas we'll discuss today.
- A comprehensive book on the GPRA is: Amy Langville and Carl Meyer, "Google's PageRank and Beyond", Princeton University Press,

2006, ISBN-13: 978-0-691-12202-1, ISBN-10: 0-691-12202-4.

- Our discussion today will contain several simplifying ideas. I am indebted to Nick Korevaar who adapted the GPRA for classroom use and shared his notes with me.

- The GPRA focuses on the **Google Matrix** which has one row and one column for every web page considered by Google.

- See https://searchengineland.com/googles-search-indexes-hits-130-trillion-pages-documents-263378

- According to that web page Google indexes 130 trillion web pages and documents.

- That's about 18,000 pages and documents for every person on earth.

- That's mind boggling.

- Suppose the number of web pages searched is $n$. Clearly, some web pages are more important than others. We want to rank them.

- Note that this is independent of any particular search. Any actual search would look for highly ranked pages that are also relevant to the search.

- The basic idea of the GPRA is to consider how web pages are linked to others.

- Consider a surfer (Alice) moving from one web page to another. After looking at some particular page, with likelihood $\epsilon$ (where in the original patent $\epsilon = 0.15$) Alice will move next to a random page in the web, and with likelihood $1 - \epsilon$ Alice will follow with equal probability one of the links on the current page. If there aren't any links Alice will move to a random page with equal probability.
- The Google matrix $G$ has one row and one column for each web page.
- The entry $g_{ij}$ is the likelihood that Alice, when on page $j$, moves next to page $i$.
- Suppose that page $j$ has links to $n_j \geq 0$ other pages. Then

$$g_{ij} = \begin{cases} \begin{cases} \frac{(1-\epsilon)}{n_j} + \frac{\epsilon}{n} & \text{if page } j \text{ has a link to page } i \\ \frac{\epsilon}{n} & \text{otherwise} \end{cases} & \text{if } n_j > 0 \\ \frac{1}{n} & \text{if } n_j = 0 \end{cases}$$

- A page with no outlinks (links to other pages), is a dangling page (or node). For simplicity we'll ignore dangling nodes in our discussion. Thus for all pages we assume that

$$n_j > 0.$$

- Then we can write

$$G = (1 - \epsilon)A + \epsilon B$$

where

$$a_{ij} = \begin{cases} 0 & \text{if page } j \text{ has no link to page } i \\ \frac{1}{n_j} & \text{otherwise} \end{cases}$$

and $B$ has all entries equal to $\frac{1}{n}$.

- Note that $G$ is a huge matrix which has no zero entries at all.
- Usually, problems with large matrices become tractable only when the matrices are **sparse**, i.e., they have many zero entries.

The Google matrix is full, i.e., no entry at all is zero.

- However, most entries are the same, and equal to $\epsilon/n$.
- The columns of $G$ add to 1. The entries in each column are probabilities.
- Consider the simple example shown in Figure 1. Here $n = 4$. Page 1 links to page 2, page 2 to page 3, page 3 to page 4, page 4 back to page 1, and page 4 also links to page 3. There is also a link from page 1 to page 3.
- page 3 has three pages linking to it, so it ought to be important.
- page 4 is being linked to by an important page, so it ought to be important also. Pages 1 and 2 seem to be less important. However, page 1 is being linked to by an important page

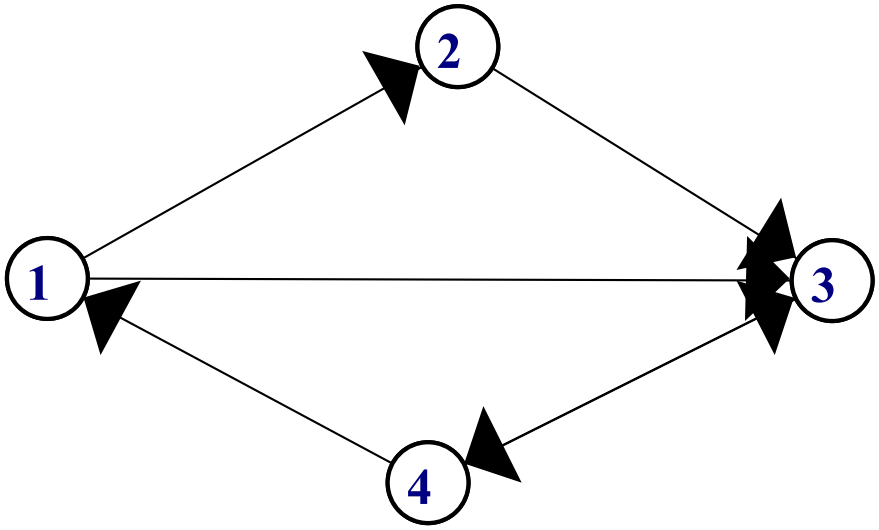


**Figure 1.** A simple internet.

and 2 is not, so we expect page 1 to be more important than page 2.

- Let's see what GPRA says.
- Letting $\epsilon = \frac{1}{6}$ (so as to get simple fractions) we get the matrices:

$$A = \begin{bmatrix} 0 & 0 & 0 & 1/2 \\ 1/2 & 0 & 0 & 0 \\ 1/2 & 1 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

and

$e = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$ $G^T e = e$

$$\begin{aligned} G &= \frac{5}{6}A + \frac{1}{6}B \\ &= \frac{5}{6}A + \frac{1}{24}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} \\ &= \frac{1}{24}\begin{bmatrix} 1 & 1 & 1 & 11 \\ 11 & 1 & 1 & 1 \\ 11 & 21 & 1 & 11 \\ 1 & 1 & 21 & 1 \end{bmatrix} \begin{bmatrix} 1/24 \\ 11/24 \\ 11/24 \\ 1/24 \end{bmatrix} \end{aligned}$$

- We now shift our point of view. Suppose the internet is surfed by a number $N$ of surfers that is much larger than the number of pages:

$$N \gg n.$$

Every second, all surfers go simultaneously to another page, each according to the probabilities given by $G$.

- Then at any point in time for each page there is a number of surfers visiting that page at the moment.
- Suppose now that $\mathbf{v}(t)$ is the vector of relative frequencies at a certain time $t$. Thus

$$v_j(t) = \frac{\#\text{ of visitors on page } j \text{ at time } t}{N}$$

Then the expected vector $\mathbf{v}(t+1)$ is

$$\boxed{\mathbf{v}(t+1) = G\mathbf{v}(t)} \qquad (1)$$

- Every page $i$ receives new visitors from other pages according to the probabilities in the $i$-th row of $G$. That's just what (1) says.

It turns out that there is a vector $\mathbf{v}$ of relative frequencies where the numbers of visitors remains constant on each page, i.e.,

$$\boxed{G\mathbf{v} = \mathbf{v}}$$

- In other words, 1 is an eigenvalue of $G$! The corresponding eigenvector $\mathbf{v}$ gives the ranking of the pages. Page $i$ is more important than page $j$ if $v_i > v_j$.

- Why is 1 an eigenvalue? We can easily see that 1 is an eigenvalue of $G^T$ since the columns of $G$, and the rows of $G^T$ add to 1. Thus if we multiply $G^T$ with the vector $\mathbf{e}$ of all 1s we get

$$G^T\mathbf{e} = \mathbf{e}.$$

- The eigenvalues of $G$ are the same as those of $G^T$. So 1 is an eigenvalue of $G$. Using Matlab, we can compute the corresponding eigenvector of $G$. It is given by

$$\mathbf{v} = \begin{bmatrix} 0.1834 \\ 0.1181 \\ 0.3583 \\ 0.3402 \end{bmatrix}$$

- Thus the pages, ordered by decreasing importance are

$$\boxed{3} > \boxed{4} > \boxed{1} > \boxed{2},$$

as we expected.

- The same ranking is suggested in Figure 2 where the amount of red color indicates the importance of the page.

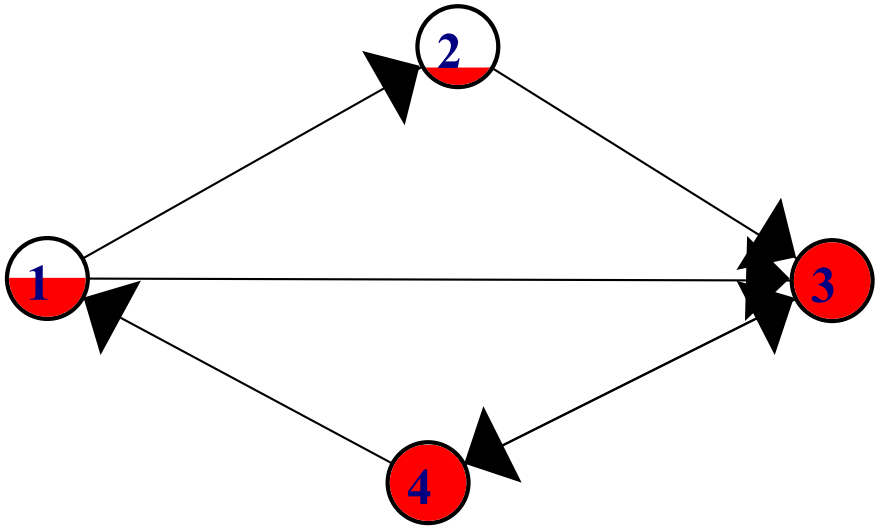


**Figure 2.** Pages Ranked.

# Perron Frobenius Theory

- The theoretical aspects of the Google Matrix and computations with it are well beyond the scope of our class, but I'll list some relevant facts. For details consult the book by Langville and Meyer.
- 1 is an eigenvalue of $G$.
- Its algebraic and geometric multiplicity are both 1.
- The components of the corresponding eigenvector are all of the same sign. Thus in particular we can find an eigenvector $\mathbf{v}$ of relative frequencies, with positive components that add to 1.
- The eigenvector $\mathbf{v}$ can be found as a nontrivial solution of the homogeneous linear system

$$(G - I)\mathbf{v} = \mathbf{0}$$

or as a solution of the eigenvalue problem

$$G\mathbf{v} = \mathbf{v}.$$

- All eigenvalues of $G$ other than 1 have an absolute value less than 1.
- The iteration

$$\mathbf{v}_0 \quad \text{given}, \qquad \mathbf{v}_{k+1} = G\mathbf{v}_k, \qquad k = 0, 1, 2, \ldots \tag{2}$$

converges to $\mathbf{v}$.

- It converges for all starting vectors $\mathbf{v}_0$ but of course it converges the faster the closer $\mathbf{v}_0$ is to $\mathbf{v}$. So in particular one might construct a starting vector based on results from analyzing an earlier version of the web.

- The computational challenges are formidable. Using the straightforward implementation of the iteration (2) one iteration costs $n^2$ operations. With $n$ being 1 trillion, and being able to perform a trillion operations per second, say, one iteration still takes a trillion seconds which equals about 32,000 years.

- What Google actually does to compute page ranks is a closely guarded trade secret.