The Billion Dollar Eigenvector

The mathematics behind Google's pagerank algorithm

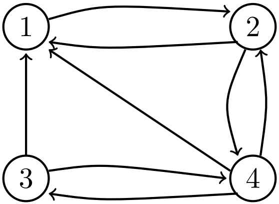
Dan Christensen

The Web

Google came to prominence, and became a multi-billion dollar corporation, because they were able to provide the most relevant search results.

How do they do it? We'll describe a simplified version of their PageRank algorithm.

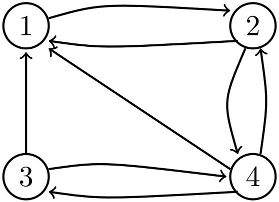
To make things concrete let's consider a simplified web with only four pages that are linked as follows:



PageRank

The idea behind PageRank is that we should give each page a score which is based on the number of links to that page.

So, in our example network



page 1 should rank highly because it has a lot of incoming links. So the scores might be:

$$x_1 = 3, \quad x_2 = 2, \quad x_3 = 1, \quad x_4 = 2 \quad ?$$

Some votes matter more

There are two extra tricks that make this work well.

First, links from a page that has a high PageRank score should count for more.

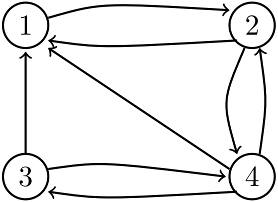
Some votes matter more

There are two extra tricks that make this work well.

First, links from a page that has a high PageRank score should count for more.

This would suggest formulas such as

$$x_1 = x_2 + x_3 + x_4$$
$$x_2 = x_1 + x_4$$
$$x_3 = x_4$$
$$x_4 = x_2 + x_3$$

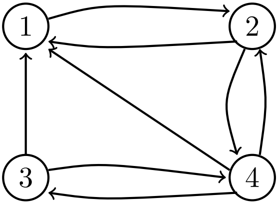


But, there are various problems with this.
For example, there is **no non-zero solution** to this system!

Sharing the vote

The second trick is that when a page links to several other pages, the score it gives to them should be shared, giving:

$$\begin{aligned}
x_1 &= \quad \tfrac{1}{2}x_2 + \tfrac{1}{2}x_3 + \tfrac{1}{3}x_4 \\
x_2 &= x_1 \qquad\qquad\quad + \tfrac{1}{3}x_4 \\
x_3 &= \qquad\qquad\qquad\quad \tfrac{1}{3}x_4 \\
x_4 &= \quad \tfrac{1}{2}x_2 + \tfrac{1}{2}x_3
\end{aligned}$$



This approach works well! For our web, it gives:

$$x_1 = 4, \quad x_2 = 5, \quad x_3 = 1, \quad x_4 = 3,$$

with page 2 ranked the highest.

Matrix form

The equations we got

$$\begin{aligned} x_1 &= \quad \tfrac{1}{2}x_2 + \tfrac{1}{2}x_3 + \tfrac{1}{3}x_4 \\ x_2 &= x_1 \qquad\qquad\quad + \tfrac{1}{3}x_4 \\ x_3 &= \qquad\qquad\qquad\quad \tfrac{1}{3}x_4 \\ x_4 &= \quad \tfrac{1}{2}x_2 + \tfrac{1}{2}x_3 \end{aligned}$$

can be written in matrix form:

$$\mathbf{x} = A\mathbf{x}$$

where

$$\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \quad \text{and} \quad A = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{3} \\ 1 & 0 & 0 & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{3} \\ 0 & \frac{1}{2} & \frac{1}{2} & 0 \end{pmatrix}$$

Eigenvectors

We know that solving the system

$$\mathbf{x} = A\mathbf{x}$$

is called finding an eigenvector of A with eigenvalue 1. Since A is a stochastic matrix, such an $\mathbf{x}$ always exists.

Eigenvectors

We know that solving the system

$$\mathbf{x} = A\mathbf{x}$$

is called finding an eigenvector of A with eigenvalue 1. Since A is a stochastic matrix, such an $\mathbf{x}$ always exists.

What's more surprising is that there is an efficient way to compute it, even when A is huge. (It might be 10 billion by 10 billion!)

For more details, see the excellent article by Kurt Bryan and Tanya Leise at

`http://www.rose-hulman.edu/~bryan/google.html`

Or just put "google eigenvector" into Google and you'll find it!

Eigenvectors

We know that solving the system

$$\mathbf{x} = A\mathbf{x}$$

is called finding an eigenvector of A with eigenvalue 1. Since A is a stochastic matrix, such an $\mathbf{x}$ always exists.

What's more surprising is that there is an efficient way to compute it, even when A is huge. (It might be 10 billion by 10 billion!)

For more details, see the excellent article by Kurt Bryan and Tanya Leise at

`http://www.rose-hulman.edu/~bryan/google.html`

Or just put "google eigenvector" into Google and you'll find it!

PS: When you are rich, don't forget who taught you Linear Algebra!