Math 224
Tuesday, December 11, 2007
Google's PageRank Algorithm

1 Background.

During a typical use of a search engine, the following occurs:

1. The user enters a search word.
2. The search engine finds all web pages that contain the given word.
3. The search engine lists all the pages that contain the word in some order, ideally from most "interesting" to least "interesting."

When Google was launched in 1998, there were already a number of search engines. Google's main innovation was a superior way of performing step 3, i.e. a better way of ranking web pages. Google's method is called the PageRank algorithm, and was developed by Google founders Sergey Brin and Larry Page while they were graduate students at Stanford University. Brin and Page were 23 and 24 years old, respectively, when they developed the PageRank algorithm.

- Let N denote the total number of pages on the web (currently over 1 billion, according to Google).
- A ranking of all of the web pages can be considered as a vector $\mathbf{v}$ in $\mathbb{R}^N$, where the i-th component of $\mathbf{v}$, v_i, is a measure of the value (or interest or importance) of page i.
- In a given search, the first page Google will list is the page i among those containing the search word with the highest value of v_i.

In the PageRank algorithm, the ranking of pages (i.e. the calculation of the ranks v_i) is based solely on *how pages are linked* and not on the content of the pages or on how often the pages are visited. Of course, the web is constantly changing, so the rankings change, as well. Google calculates the ranks once per month. We'll start with a very simple ranking method, and gradually build up to the PageRank algorithm.

2 Method 1: The Very Simple Method.

We'll say that page j is *recommending* page i if page j has a link to page i. One idea is to rank pages purely by the number of recommendations each page receives. Let A be the $N \times N$ matrix whose i, j entry is

(2.1) $a_{i,j} =$

Then the number of recommendations that page i receives is:

$$a_{i1} + a_{i2} + a_{i3} + \cdots + a_{iN}.$$

We can think of this as a measure of the *value* v_i of page i:

(2.2) $v_i =$

Thus the ranking vector $\mathbf{v}$ is obtained by:

(2.3) $\mathbf{v} =$

3 Method 2: A First Improvement.

Being included on a list of the top 10 restaurants in New York should be viewed as "better than" being included on a list of all of the restaurants in New York. Thus, the weight of a recommendation made by page j should be inversely proportional to the total number of recommendations made by page j. Thus we replace Equation (2.2) with the following:

$$v_i = \tag{3.1}$$

Note that this equation can be undefined if j is a "dead-end", i.e. if $n_j = 0$, since then Equation (3.1) involves division by 0. Google gets around this problem by pretending that a page with no links out in fact has a link to *every* page, including itself. Thus we redefine the entries a_{ij} of the matrix A as follows:

$$a_{i,j} = \tag{3.2}$$

Then the ranking vector $\mathbf{v}$ is obtained by:

$$\mathbf{v} = \tag{3.3}$$

4 Method 3: A Further Improvement.

The weight of a recommendation should depend on the importance of the recommender. This *insensitivity to spammers* is one of the big advantages of PageRank over earlier search engines. To get a high overall PageRank, your page must be referenced by other high-ranking pages, something that is hard for any particular individual to influence. Since the importance of page i is measure by v_i, the value of page i, we by multiply each term in the right-hand side of Equation (3.1) by v_i. Thus we obtain:

(4.1) $v_i =$

Equation (4.1) can be expressed in matrix form as

(4.2) $\mathbf{v} =$

Equation (4.2) says that the ranking vector $\mathbf{v}$ must be an *eigenvector* of the matrix P with corresponding *eigenvalue* 1. So to find the ranking vector $\mathbf{v}$ using Method 3, one first forms the matrix P and then finds an eigenvector $\mathbf{v}$ of P corresponding to the eigenvalue 1. Thus, for the matrix P to give a ranking $\mathbf{v}$ of the web pages, 1 must be an eigenvalue of P. Fortunately, 1 must always be an eigenvalue of P since P is a *Markov* matrix:

1. Each entry of P is greater than or equal to 0.
2. The sum of the entries in each column of P is 1.

You have already proven that a Markov matrix always has an eigenvalue equal to 1, so we know that P does have an eigenvector $\mathbf{v}$ with eigenvalue 1.

You can consider the Markov matrix P as describing a Markov chain, or random walk, over the web in which you move from page to page as follows. When you're on a given page, you choose at random one of the links from that page, and then you follow that link. If you're at a dead end (no links), you choose a page at random from the entire web and move to it. If we imagine a large population of web surfers moving around the web in this way, then the i, j entry of P gives the expected fraction of those surfers on page j who will then move to page i. A solution $\mathbf{v}$ to the equation

$$\mathbf{v} = P\mathbf{v}$$

can be regarded as describing a steady state or equilibrium distribution of surfers on the various pages.

5 Method 4: The PageRank Algorithm.

Even though we know that the matrix P will always have an eigenvector $\mathbf{v}$ with eigenvalue 1, we don't know if there will be a unique such vector. In fact, if there are several clusters of web pages not connected to each other (which, of course, is the case in reality), the matrix P will have a set of two or more linearly independent eigenvectors with eigenvalue 1. In that case, the equation $\mathbf{v} = P\mathbf{v}$ will not determine a unique ranking.

Example. Suppose that there are only 4 pages on the web. Suppose that pages 1 and 2 link to each other, and that pages 3 and 4 link to each other, but there are no other links. The equation $\mathbf{v} = P\mathbf{v}$ implies that $v_1 = v_2$ and $v_3 = v_4$, but gives no information about the relative sizes of v_3 and v_4.

Google avoids these problems by replacing P by another Markov matrix Q, which is constructed as follows. First, fix some parameter r such that $0 < r < 1$. Google uses $r = 0.85$. Imagine that you have a biased coin such that the probability of heads is r. Now move around the web as follows.

- If you're on page j, flip the coin.
- If you get heads, then choose randomly one of the links from page j and follow that link.
- If page j is a dead-end, or if the coin comes up tails, then pick a page at random from the whole web and go to that page.

The matrix Q that describes this process is given by:

(5.1) $Q =$

Then the matrix Q is still a Markov matrix, but unlike P, the matrix Q has no zeros. Then we can apply the following theorem to Q.

Theorem. Let Q be a Markov matrix none of whose entries is 0. Then there is a **unique** vector $\mathbf{v}$ such that:

- The entries of $\mathbf{v}$ are all positive.
- The sum of the entries in $\mathbf{v}$ is 1.
- $\mathbf{v} = Q\mathbf{v}$.

Thus, up to scaling, Q has a unique eigenvector $\mathbf{v}$ with eigenvalue 1. This eigenvector $\mathbf{v}$ gives a ranking of the web pages: the page i with the largest v_i is ranked first, and so on. So to determine a ranking $\mathbf{v} \in \mathbb{R}^N$ of all of the pages on the web, we first calculate the Markov matrix Q, and then compute the eigenvector $\mathbf{v}$ corresponding to the eigenvalue 1. To find the eigenvector $\mathbf{v}$ corresponding to the eigenvalue 1, we must solve the linear system $(Q - I)\mathbf{v} = \mathbf{0}$. Since there are over 1 billion pages on the web, this is a computationally difficult system to solve. One area of current research in linear algebra is the construction of efficient algorithms for solving large linear systems such as this one.

References

[1] S. Kamvar, T. Haveliwala, and G. Golub, *Adaptive methods for the computation of PageRank*, Linear Algebra and its Applications, 2004, pp. 51–65.

[2] L. Page, S. Brin, R. Motwani, and T. Winograd, *The PageRank citation raking: bringing order to the web*, Technical Report, Stanford Digital Libraries Project, 1998.

[3] D. Higham and A. Taylor, *The sleekest link algorithm*, The Institute of Mathematics and Its Applications (IMA) Mathematics Today,2003, pp. 192–197.

[4] T. Haveliwala and S. Kamkar, *The second eigenvalue of the Google matrix*, 2003, Stanford University Technical Report.