Chapter 8

PageRank

Many attribute the early success of Google to its superior search engine, based on an algorithm they called PageRank. It was assigned US Patent US6285999B1, with the sole inventor listed as Larry Page, one of the two founders of the firm. However, Page credited co-founder Sergei Brin as contributing in conversations about the idea, as well as some Stanford University faculty and some previous researchers who had done related work. The patent belongs to Stanford. The name, *PageRank*, is a pun on both the inventor's surname and the fact that it indexes Web *pages*.

The goal is to rank all the pages of thw World Wide Web in order of importance. The key is defining that latter term.

8.1 Markov Model

The method models Web "surfing" as following a *Markov chain*, a very widely-used approach to stochastic modeling in many fields. Its main assumption is rather strict, but in this case that is not very important, as we are merely putting together a rough measure of the importance of each Web page.

8.1.1 Structure

One observes some process at times 1,2,3,... The *state* at time m, a random variable, is denoted by X_m. Different applications have different state spaces; it could be queue length in a network buffer, for instance.

The main assumption is that the system follows the *Markov property*, which in rough terms can be described as:

> The probabilities of future states, given the present state and the past states, depends only on the present state; the past is irrelevant.

We sometimes say the system is "memoryless" — in looking to the future, knowing the present, the past is "forgotten."

A famous example is *random walk*. Say at time 0 we are at position 0 on the real number line. We flip a coin, then go 1 step left or right, depending on whether the outcome is head or tails. Then we do that again, repeatedly. X_m is our position at time m. Clearly this system is Markovian.

In formal terms:

$$P(X_{t+1} = s_{t+1} | X_t = s_t, X_{t-1} = s_{t-1}, \ldots, X_0 = s_0) = P(X_{t+1} = s_{t+1} | X_t = s_t) \tag{8.1}$$

8.1.2 Matrix Formulation

We define p_{ij} to be the probability of going from state i to state j in one time step; note that this is a *conditional* probability, i.e. $P(X_{n+1} = j \mid X_n = i)$. These quantities form a matrix, denoted by P.

So, this matrix gives the probabilites of going from state i to state j in one time step:

$$P(X_{n+1} = s \mid X_n = r) = P_{rs} = p_{rs} \tag{8.2}$$

And the Markov property can be used to show that P^2 gives the probabilities of the various 2-time step transitions, i.e.

$$P(X_{n+2} = s \mid X_n = r) = (P^2)_{rs} \tag{8.3}$$

Similarly, P^k gives the k-step probabilities,

$$P(X_{n+k} = s \mid X_n = r) = (P^k)_{rs} \tag{8.4}$$

8.1.3 Long-Run Behavior

Suppose the chain is *irreducible*, meaning that one can reach every state from every other, in a finite number of steps.[1] Suppose also that the state space is finite, and *aperiodic*. The latter term

[1] Computer scientists, true to form, have their own separate term for this, *strongly connected*.

means that the GCD of return times to any state is 1; the random walk is periodic, since that number is 2. Then

$$\pi_r = \lim_{m \to \infty} P(X_m = r) \tag{8.5}$$

exists, and is independent of the starting position X_0.

Note that π is a vector, with element r defined as above.

8.1.4 Finding π

(8.5) and (8.4), and our by-now well-honed skill at using partitioned matrices imply that

$$\lim_{m \to \infty} P^m = \begin{pmatrix} \pi' \\ \pi' \\ ... \\ \pi' \end{pmatrix} \tag{8.6}$$

Note that the fact the rows are identical here reflects the independence of (8.5) of the starting position X_0.

This gives us a way to find π: Simply raise P to a high power, and then each row is approximately π'. In fact, we could the get an even better approximation by averaging those rows. (Of course, this is for chains with finite state spaces.)

Alternatively, one can view the whole thing as an eigenvalue problem. The above material can be used to show that

$$\pi' = \pi P \tag{8.7}$$

so that

$$\pi' = P' \pi' \tag{8.8}$$

which makes π' an eigenvector of P' with eigenvalue 1. It's also possible to show that the other eigenvalues are between 0 and 1. So an alternate way to find π is to find the eigenvectors of P'. And in turn one way to do that is to use the famous Power Method for finding the eigenvector corresponding to the largest eigenvalue of a matrix. That method involves powers of P as well, so in the end it's similar to the above.

8.2 The PageRank Model

Here the state space consists of one state for every page on the Web, and "time" is hops from one page to another. X_3, say, is the third page you visit during a Web surfing session. What Page had to do was devise some clever formulation of the matrix P that would reasonably model Web surfing.

8.2.1 Details

The Markov property is questionable here, but again, we don't care much about that. It's just a mechanism for devising an importance measure.

Now, what is the matrix P here? For each Web page i, let n_i denote the number of outlinks from that page. The model is that after a visit to that page, the Web surfer will choose one of those links, with uniform probability among them, i.e. $1/n_i$ each. The π vector then becomes the vector of importance scores for the various pages.

Google also adds a *damping* factor to the model, which we will not go into here.

8.2.2 Computation of π

All of the above is fine, but there are literally billions of Web pages. Thus the matrix P has billions of rows and columns. Fortunately, though, it is a very sparse matrix, lots of 0s, so we can exploit that property to reduce computation.

Note too that there is a shortcut to find the matrix powers: We first square P, then square the result, yielding P^4, then square *that* result, yielding P^8 and so on.

There are many refinements of all this, of course.

8.3 Then What?

Clearly there is a lot more to what Google does to provide users with recommended Web page. What we saw here is only a first step.