... picking up from where the slides ran out...

What is page rank?

→ It's just a measure of "importance" of a page, based purely on graph structure. (It doesn't depend at all on the query, and so can be precomputed.)

> Q: What possible measures of importance can you think of?

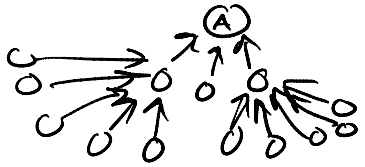
Strawman Answer: Degree

Q: Is the in-degree of a webpage a good measure of importance?

A: No!

→ Not all links are equal!

e.g. Ⓐ is likely the most important in this graph



→ It's very easy to manipulate.

Q: What can we do to improve on degree?

Idea: Weight the edges by the importance

Details: Let $r_j(t)$ be the rank of page j at iteration t.

Set $r_j(0) = 1/n$ ← # of nodes

& $r_j(t+1) = \sum_{j \in N(i)} r_i(t)/d_i^{out}$ ← out degree of i

↑ pages that link to j

In words, each node splits its rank evenly among its outgoing links, & each node's rank is the sum of what it gets from its neighbors.

We can write this nicely in matrix form:

$$r(t+1) = r(t)P = r(t)\left[P_{ij}\right]$$

where $P_{ij} = \begin{cases} \frac{1}{d_i} & \text{if } i \to j \\ 0 & \text{else} \end{cases}$

(this is just a variation of the adjacency matrix of the graph)

Note: $\sum_j r_j(t) = 1 \ \forall t$.

Pagerank Algorithm (take 1):

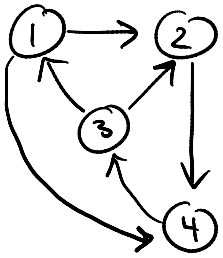
1) Assign each node an initial pagerank $1/n$.
2) Repeat $r(t+1) = r(t)P$ until it converges

☆ Think of a "random web surfer" starting at each node with probability $1/n$ and then following each link out of each node with equal probability. Then $r_j(t)$ is the

probability you're at node j at time t.

example:

Consider the graph



$$P = \begin{pmatrix} 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 \\ 1/2 & 1/2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

$$r(0) = (\tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4})$$

$$r(1) = r(0)P = (\tfrac{1}{8}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{3}{8})$$
$$r(2) = r(1)P = (\tfrac{1}{8}, \tfrac{3}{16}, \tfrac{3}{8}, \tfrac{5}{16})$$
$$r(3) = r(2)P = (\tfrac{3}{16}, \tfrac{1}{4}, \tfrac{5}{16}, \tfrac{1}{4})$$
$$r(4) = r(3)P = (\tfrac{5}{32}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{11}{32})$$
$$\vdots$$

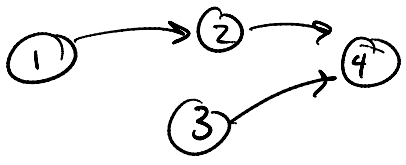
Two questions we want to answer about this:

1) Does this always converge?

2) Does it converge to what we want?

To get a feel, let's look at some more examples:

ex:



$$P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$r(0) = (\tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4}, \tfrac{1}{4})$$

$$r(1) = r(0)P$$

$$\begin{pmatrix} \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$r(1) = r(0)\,P$

$r(2) = (0, \frac{1}{4}, 0, \frac{1}{2})$

$r(3) = (0, 0, 0, 1)$

$r(4) = (0, 0, 0, 1)$

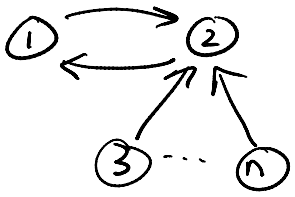
So, we converge...

But, is this what we want?

No!

we would answer node 4 for every query!

ex:



$$P = \begin{pmatrix} 0 & 1 & 0 & \cdots \\ 1 & 0 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ 0 & 1 & 0 & \cdots \\ \vdots & \vdots & \vdots & \\ \vdots & \vdots & \vdots & \end{pmatrix}$$

$r(0) = (\frac{1}{n}, \cdots, \frac{1}{n})$

$r(1) = r(0)P = (\frac{1}{n}, \frac{n-1}{n}, 0, \cdots 0)$

$r(2) = r(1)P = (\frac{n-1}{n}, \frac{1}{n}, 0, \cdots 0)$

$r(3) = (\frac{1}{n}, \frac{n-1}{n}, 0 \cdots)$

$r(4) = (\frac{n-1}{n}, \frac{1}{n}, 0, \cdots)$

So it oscilates and never converges and the difference between ranks goes from ≈ 0 to ≈ 1 between iterations!

$\Rightarrow$ So there are certainly some problems with our initial version of pagerank!

1) periodicity in the links caused trouble in the second example.
2) Having weakly connected nodes caused problems in both examples.

(if we had been disconnected of

Course more problems would have shown up!)

Thm. If the graph is finite, strongly connected and aperiodic, then $\lim_{t\to\infty} r(t) = \pi$ & π is the unique solution to $\pi = \pi P$

"Stationary" rank vector

def G is aperiodic iff ∀ vertices v the GCD of the path lengths from v to itself is 1

Q: Why is π called the stationary vector?

A: Because if we were to start with $r(0) = \pi$ then the ranks would never change.

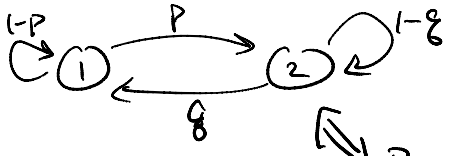
Q: Does anyone recognize what π is?

A1: left eigenvector of P w/ eigenvalue 1.

A2: Stationary distribution of the Markov chain defined by P. (Take CS147 or ACM216 to learn more about these)

Aside: 1 thing to notice is that this theorem gives us a second way to calculate pagerank: using $\pi = \pi P$ & $\sum_i \pi_i = 1$.

ex: Consider



$$\Rightarrow P = \begin{pmatrix} 1-p & p \\ q & 1-q \end{pmatrix}$$

So $\pi = \pi P$:

$$\pi_1 = \pi_1(1-p) + \pi_2 q \Rightarrow p\pi_1 = q\pi_2$$

$$\pi_2 = \pi_1 p \cdot \pi_2(1-q) \Rightarrow p\pi_1 = q\pi_2$$

$$\pi_1 + \pi_2 = 1$$

$$\pi_1 + \frac{p}{q}\pi_1 = 1$$

$$\pi_1 = \frac{1}{1+p/q} = \frac{q}{p+q}$$

$$\& \ \pi_2 = p/p+q$$

....back to pagerank...

★ **Big Question** Is the web graph finite, aperiodic, & strongly connected?

A: Of course not

So how do we adjust P so that it is aperiodic & strongly connected?

↓

Add an edge from each node to every other node.

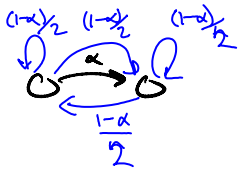
... make this edge have small weight compared to the real edges.

Specifically:

instead of P, use

$$G = \alpha P + (1-\alpha)\begin{bmatrix} \frac{1}{n} & \frac{1}{n} & \cdots \\ \frac{1}{n} & \ddots & \\ \vdots & & \end{bmatrix}$$

So

Becomes 

Q: what is an interpretation for this

change?

A: Random web surfer follows links for a while, but w/prob $(1-\alpha)$ starts fresh...

so this can be viewed as more "realistic" too!

Great! We now have an algorithm that we know will provide convergence to something reasonable.

& we have two ways to calculate the ranks

(1) Iterate $r(t) = r(t-1)\,G$

(2) Solve $\pi = \pi G$ & $\sum \pi_i = 1$

(set of n equations with n unknowns)

Q: Which does google use?

A: (1) ...

Because of network structure (small world & heavy tailed degree) convergence of the iterations happens quite quickly. $\simeq 100$ iterations whereas (2) requires inverting a very large matrix...

Pagerank Algorithm (take 2):

1) Assign each node an initial pagerank $1/n$.
2) Repeat $r(t) = r(t-1)\,G$ for ≈ 100 steps

Some other issues:

1) Q: What should α be?

   A: $\alpha \approx .85$ turns out to be good for both computation time & results quality.

   (Turns out to get w/in ε of π you need $\frac{\log \varepsilon}{\log \alpha}$ iterations.

   $\alpha \to 1$ lots of iterations
   $\alpha \to 0$ few iterations, but ignore web structure.)

2) We made the matrix very dense by adding lots of links
   → This seems to make computation much harder!

   ✡ This can often be avoided, ie. we can still compute with sparse matricies.
   (Possible HW question!)

Now that we understand page rank...
What are it's problems?

(1) Takes lots of computation!

Right now it takes days to calculate & it needs to be updated weekly or faster!.

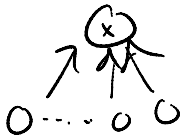
So → it cannot be personalized much to
- type of user, or
- topic

... which both would lead to big improvements in quality.

(2) Google Bombs

Page rank can be gamed by exploiting link structure!

"Link farms" → to help page x, make lots of other pages that will link to it.



even though these pages aren't linked to x guarantees them some pagerank which they can transfer to x.

Q: How can google fight this?

A: Change G so that when the "random surfer" restarts it always jumps to "safe" pages. (e.g. google, cnn, etc.)

(e.g. google, cmu, etc.)

i.e $G = \alpha P + (1-\alpha) \begin{bmatrix} \frac{1}{n} & 0 & \cdots & \frac{1}{n} & 0 & \cdots \\ \vdots & \ddots & & \vdots & & \end{bmatrix}$

... But you can't do this completely otherwise new pages that aren't linked to can't get positive page rank...

Last thought:

In real life G changes, but only by a small amount.

Q: Can we take advantage of this?

A: It seem so... this should make the iterations converge faster if we start from the old pageranks

→ This seems natural, but unfortunately it doesn't usually work

---

Now that we're finished w/ page rank, lets go back to the big picture

... go back to ppt