Lecture 9 - PageRank

Wednesday, February 8, 2023 3:31 PM

Joke: What football team used the power of graph theory to tour their way to the 1960 and 1961 AFL championships?

A: Houston Eulers

PageRank

- "A modern classic."

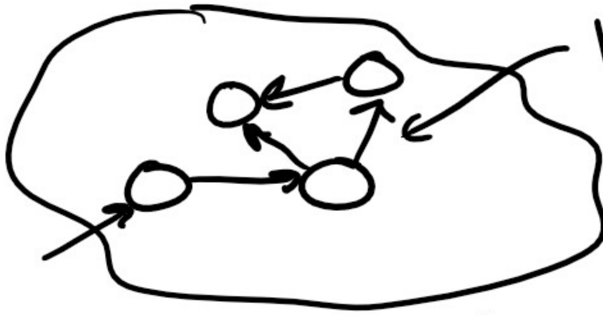
  — Gittens

→ a <u>centrality</u> algorithm

↳ importance

BITD: a lot of internet, but no good way to find stuff

Google: we'll incorporate "trust" into our search



hyperlinks

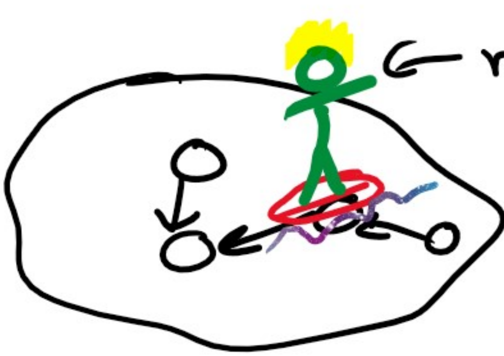
web pages

Internet = Big ol' graph

"Trust" → you're trusted if trustworthy site link to you

PageRank == trust

Random surfer model



← random surfer

↳ performing an infinite random walk

Internet

PageRank: probability that our random surfer is at vertex v at some point in time

Issue: what about $d^+(v) = 0$ ← sinks

$d^-(v) = 0$ ← source

↳ randomly jump from a sink to any vertex with equal probability

---

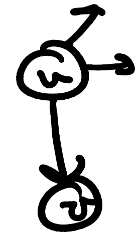
Graph Algorithmic Model

→ vertex centric computation over some iterations

PageRank:

initialize $p(v) = \frac{1}{|V(G)|}$ for all $v \in V(G)$

we iterate $p(v) = \sum\limits_{u \in N^-(v)} \frac{p(u)}{d^+(u)}$



To hand sinks | sources:

sinks: add edges to all $v \in V(G)$

In addition:

add some additional probability of a random jump for each step the surfer takes

(damping factor)

Linear Algebraic Model

consider our adj. matrix A

consider the diagonal degree matrix D

- D_{ii} = out degree of vertex i

$$= \sum_j A_{ij}$$

- $D_{ij} = 0 \quad \forall i \neq j$

define transition probability matrix M'

$M' = D^{-1}A \leftarrow$ defines transitional

$M' = D^{-1}A$ ← defines transitional probabilites following out edges

$M = (D^{-1}A)^T$ ← in edge transition transition probabilities

Q: how can we calculate PageRanks

$$p_0(v) = \frac{1}{|V(G)|}$$

↑ PR on iteration zero

$$p_{i+1} = M p_i$$

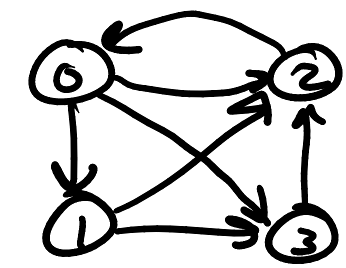
eventually $p_\infty = M p_\infty$

recall for some matrix A

$$\boxed{A\underset{\text{eigenvector}}{x} = \underset{\text{eigenvalues}}{\lambda} x}$$

→ PageRanks are just the eigenvector of M that eigenvector of M that corresponds to eigenvalue 1

Example calculation



$$A \quad \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{array}$$

$$D \begin{bmatrix} 3 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad D^{-1} \begin{bmatrix} 1/3 & & \cdots & 0 \\ \vdots & 1/2 & & \vdots \\ \vdots & & 1 & \vdots \\ 0 & \cdots & \cdots & 1 \end{bmatrix}$$

$$M^T = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 1/2 & 1/2 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad M = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 1/3 & 0 & 0 & 0 \\ 1/3 & 1/2 & 0 & 1 \\ 1/3 & 1/2 & 0 & 0 \end{bmatrix}$$

$$p_0 = \begin{bmatrix} 1/4 \\ 1/4 \\ 1/4 \\ 1/4 \end{bmatrix} \qquad p_1 = Mp_0 = \begin{bmatrix} 1/4 \\ 1/12 \\ 11/24 \\ 5/24 \end{bmatrix}$$

Competition networks

vertices = competitors

edges = competitions

orient: add direction to some undirected edge

For PageRank:

- we can orient edges to victors
- PR "flows" to winners

gives us a ranking of competitors

Modifications:

- Can weight edges by the margin of victory

Likely future special topic:

spectral clustering

spectral clustering

Graph Laplacian

$$L = D - A$$

It's spectrum contains information on how tightly coupled vertices are to each other

Application:

Graph Clustering
Graph Partitioning