Google PageRank Algorithm: An application of Numerical Linear Algebra

Module 3.5

An interesting simplification of the algorithm (seriously simplified…)

A search engine must …

- Access and index all web pages
- Determine importance of pages for a particular search
- Return pages in decreasing order of importance

Search Engine

Challenges

- Number of occurrences of search terms can be misleading
- Not all web pages are of equal significance

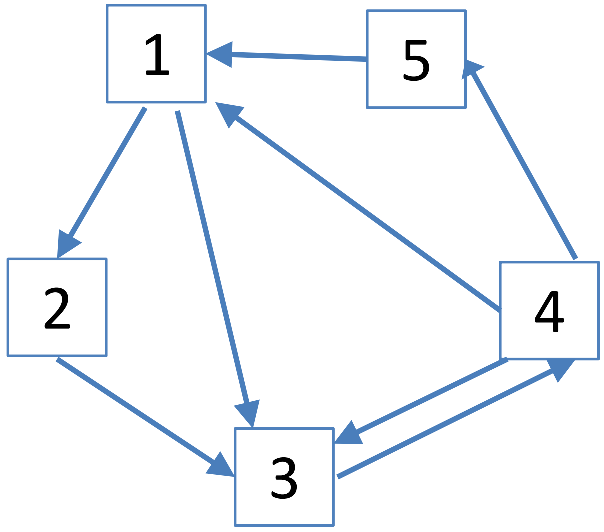
Search Engines

Consider the following:

- Importance of a web page is dependent on the number of pages that link to it.
- A web page transfers its importance to the pages it links to.

Example

- Consider web pages
- Links from page k to page j are shown as directed edges from Node k to Node j
- Which page is most important?



Measure of importance

- Let x_k be the importance of page k
- It "imparts" its importance equally to all pages it links to (each gets x_k/p importance, where k links to p pages)
- Represent this in a matrix

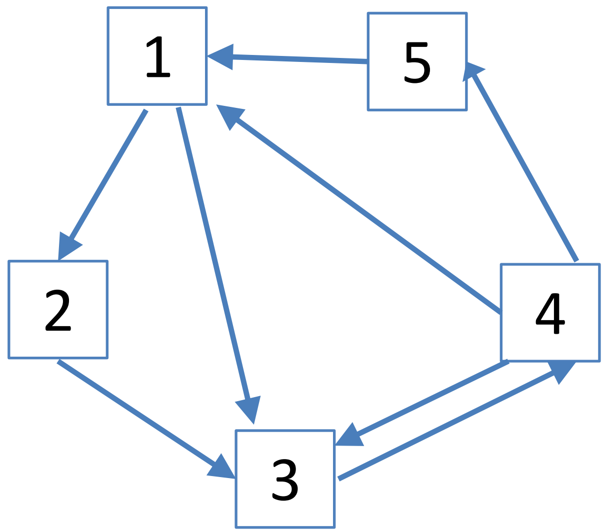
In matrix form, the importance of the web pages looks like …

$$\begin{bmatrix} 0 & 0 & 0 & 1/3 & 1 \\ 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 1/3 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1/3 & 0 \end{bmatrix} x = \begin{bmatrix} \frac{1}{3}x_4 + x_5 \\ \frac{1}{2}x_1 \\ \frac{1}{2}x_1 + x_2 + \frac{1}{3}x_4 \\ x_3 \\ \frac{1}{3}x_4 \end{bmatrix} = x$$

➔ find x such that $Ax = \lambda x$

➔ find the eigenvector for $\lambda=1$

Back to the web pages …



$Ax = x \Rightarrow x = [2,1,3,3,1]^T$

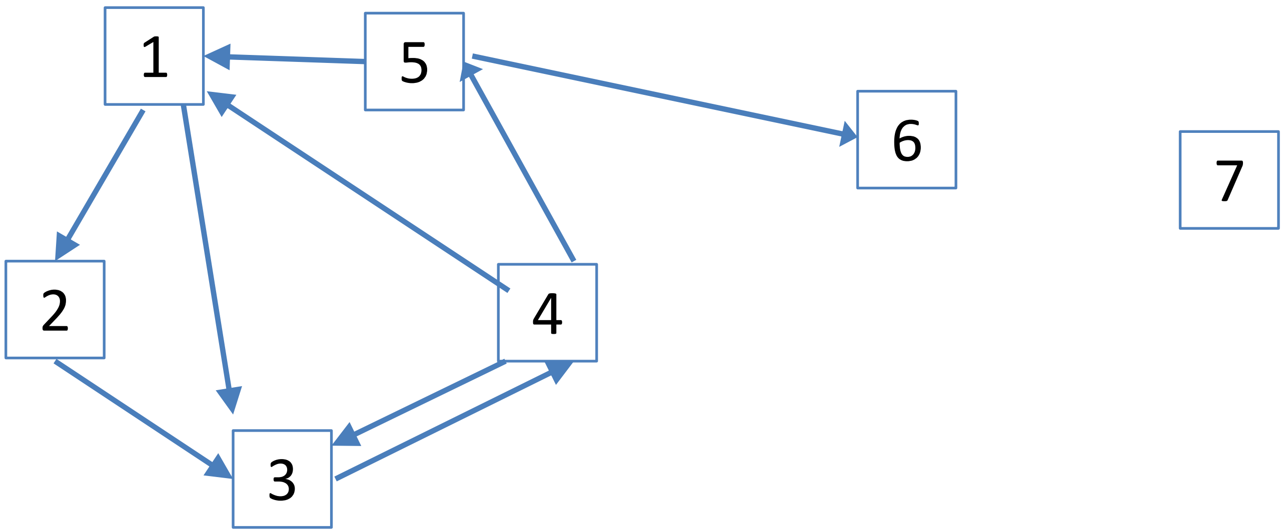
So, pages 3,4 have equal importance and would be returned first, then page 1, then pages 2 and 5.

Comments

- Were we just lucky that A has $\lambda=1$?
  – No, A is a stochastic matrix (columns sum to 1), which always have 1 as maximal absolute value eigenvalue

- Another view of the "importance" matrix A
  – A(j,k) is the probability that a random user chooses the link from page k to page j, viewing each as equally likely.

What if a page has no outgoing links?



- Using the previous approach, the columns for pages 6 and 7 are all zeroes.
- The resulting probabilty matrix is no longer stochastic, so it may not have an eigenvalue of 1.
- Instead, make all pages equally likely from these pages – including themselves!

New Probability Matrix

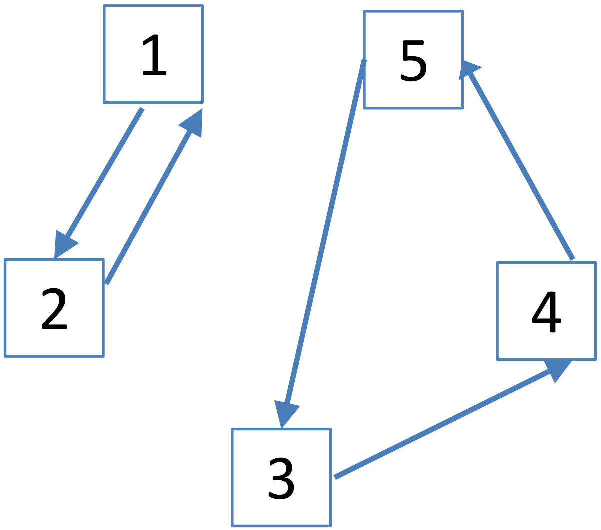
$$A = \begin{bmatrix} 0 & 0 & 0 & 1/3 & 0.5 & 1/7 & 1/7 \\ 0.5 & 0 & 0 & 0 & 0 & 1/7 & 1/7 \\ 0.5 & 1 & 0 & 1/3 & 0 & 1/7 & 1/7 \\ 0 & 0 & 1 & 0 & 0 & 1/7 & 1/7 \\ 0 & 0 & 0 & 1/3 & 0 & 1/7 & 1/7 \\ 0 & 0 & 0 & 0 & 0.5 & 1/7 & 1/7 \\ 0 & 0 & 0 & 0 & 0 & 1/7 & 1/7 \end{bmatrix}$$

This is a stochastic matrix → has eigenvalue 1

→ Eigenvector = $[1.7, 0.9, 2.9, 3, 1.1, 0.7, 0.1]^T$

1 is guaranteed to be an eigenvalue. But, what if two eigenvalues are 1?

Consider the pages:



$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

→Two eigenvalues are 1

→eigenvectors: $[1,1,0,0,0]^T$ & $[0,0,1,1,1]^T$

Breaking these "ties"

- Add a variable into the mix – a probability α of jumping to a random page and $(1-\alpha)$ of staying in the loop/sub-network you are in.
- Define $G = (1- \alpha)A + \alpha S$
- where G is the final link matrix,
- A is the probability matrix as before, and
- S is new matrix containing the probability of linking to any specific page randomly (set to 1/n for all entries)

→ Unique eigenvalues

Back to our non-unique example (α=0.15)

$$G = (1-\alpha)A + \alpha S$$

$$G = 0.85\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} + 0.15\begin{bmatrix} .2 & .2 & .2 & .2 & .2 \\ .2 & .2 & .2 & .2 & .2 \\ .2 & .2 & .2 & .2 & .2 \\ .2 & .2 & .2 & .2 & .2 \\ .2 & .2 & .2 & .2 & .2 \end{bmatrix}$$

$$G = \begin{bmatrix} .03 & .88 & .03 & .03 & .03 \\ .88 & .03 & .03 & .03 & .03 \\ .03 & .03 & .03 & .03 & .88 \\ .03 & .03 & .88 & 0.3 & .03 \\ .03 & .03 & .03 & .88 & .03 \end{bmatrix}$$ ➔ λ=1 with $\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$

Bringing this back to Numerical Linear Algebra…

- We need to calculate the eigenvector for G
- Matrix G has a row and column for every web page
- Billions of rows and columns
- How to do this?
  - No matrix decomposition (too big)
  - Only need to find eigenvector for $\lambda=1$
  - Stochastic matrix → Largest eigenvalue is 1
  - Not too many iterations

The Power Method

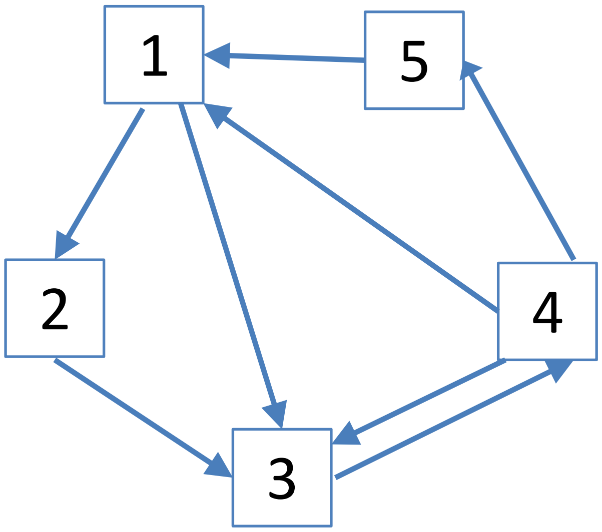
- Choose x_0 (random)
- Repeat until convergence:

$$x_{k+1} = \frac{Gx_k}{\|Gx_k\|}$$

- With α=0.15, usually takes < 100 iterations (small α takes longer)

Our original example:

$$A = \begin{bmatrix} 0 & 0 & 0 & 0.\overline{3} & 1 \\ 0.5 & 0 & 0 & 0 & 0 \\ 0.5 & 1 & 0 & 0.\overline{3} & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0.\overline{3} & 0 \end{bmatrix}$$



$$G = (1-\alpha)A + \alpha S$$

$$G = \begin{bmatrix} .03 & .03 & .03 & .3133 & .88 \\ .455 & .03 & .03 & .03 & .03 \\ .455 & .88 & .03 & .3133 & .03 \\ .03 & .03 & .88 & .03 & .03 \\ .03 & .03 & .03 & .3133 & .03 \end{bmatrix}$$

Using the power method

- Tried different starting x_0 (chosen randomly)
- After 20-23 iterations, $\|x_{k+1} - x_k\| \leq 10^{-5}$
- → converged to $[.42, .24, .61, .58,22]^T$
- Note this differs from our original vector, but relative rankings are quite similar
- Here, "best" search order is: Page 3,4,1,2,5
- Our "exact" order was: 3&4, 1, 2&5

Final comments

- Actual solution takes about 30 days → information updated about once a month
- Indexing of search terms on pages is separate – just discussing ranking of pages here
- This description is much simplified, but meets spirit of the algorithm