Case Study: Matrix Computation in Web Search and Mining

T. Yang. CS140
Some slides are from C. Li (UCI) an R. Mooney (UTexas)
Mining Web Graph for Search Ranking

- **PageRank Algorithm by Google**
- **Intuition:**
  - The importance of each page is decided by what other pages “say” about this page
  - One naïve implementation: count the # of pages pointing to each page (i.e., # of inlinks)
- **Problem:**
  - We can easily fool this technique by generating many dummy pages that point to a page
Initial PageRank Idea

• Rank $r(p)$ for page $p$:

$$ r(p) = c \sum_{q: q \rightarrow p} \frac{r(q)}{N_q} $$

  – $N_q$ is the total number of out-links from page $q$.
  – A page, $q$, “gives” an equal fraction of its authority to all the pages it points to (e.g. $p$).
  – $c$ is a normalizing constant set so that the rank of all pages always sums to 1.

• Rank of a page represents its authority on the web
Initial PageRank Idea (cont.)

• Can view it as a process of PageRank “flowing” among pages.
Representing PageRank with Matrix Computation

• Assume three web sites: Netscape, Amazon, and Microsoft.
• Their weights are represented as a vector

For instance, in each iteration, half of the weight of AM goes to NE, and half goes to MS.
How to solve the matrix equations?

Iterative method

- Initially all rank values are 1
- Compute the new values \((n,m,a)\) using their old values from the equations
- Repeat many iterations

\[
\begin{bmatrix}
 n \\
 m \\
 a
\end{bmatrix}_{\text{new}} = \begin{bmatrix}
 1/2 & 0 & 1/2 \\
 0 & 0 & 1/2 \\
 1/2 & 1 & 0 \\
\end{bmatrix} \begin{bmatrix}
 n \\
 m \\
 a
\end{bmatrix}_{\text{old}}
\]
Iterative computation

\[
\begin{bmatrix}
  n \\
  m \\
  a \\
\end{bmatrix} = \begin{bmatrix}
  1 & 1 & \frac{5}{4} & \frac{9}{8} & \frac{5}{4} \\
  1 & \frac{1}{2} & \frac{3}{4} & \frac{1}{2} & \frac{11}{16} \\
  1 & \frac{3}{2} & 1 & \frac{11}{8} & \frac{17}{16} \\
\end{bmatrix} \rightarrow \begin{bmatrix}
  \frac{6}{5} \\
  \frac{3}{5} \\
  \frac{6}{5} \\
\end{bmatrix}
\]

Final result:
- Netscape and Amazon have the same importance, and twice the importance of Microsoft.
Another web graph with rank values

Converged results from iteration computation are marked
Problem 1 of algorithm: dead ends!

\[
\begin{bmatrix}
  n \\
  m \\
  a
\end{bmatrix}_{\text{new}} = \begin{pmatrix}
  1/2 & 0 & 1/2 \\
  0 & 0 & 1/2 \\
  1/2 & 0 & 0
\end{pmatrix}
\begin{bmatrix}
  n \\
  m \\
  a
\end{bmatrix}_{\text{old}}
\]

- MS does not point to anybody
- Result: weights of the Web “leak out”

\[
\begin{bmatrix}
  n \\
  m \\
  a
\end{bmatrix} = \begin{bmatrix}
  1 & 1 & 3/4 & 5/8 & 1/2 \\
  1 & 1/2 & 1/4 & 1/4 & 3/16 \\
  1 & 1/2 & 1/2 & 3/8 & 5/16
\end{bmatrix} \rightarrow \begin{bmatrix}
  0 \\
  0 \\
  0
\end{bmatrix}
\]
Problem 2 of algorithm: spider traps

\[
\begin{bmatrix}
    n \\
    m \\
    a_{\text{new}}
\end{bmatrix} = \begin{pmatrix}
    1/2 & 0 & 1/2 \\
    0 & 1 & 1/2 \\
    1/2 & 0 & 0
\end{pmatrix}
\begin{bmatrix}
    n \\
    m \\
    a_{\text{old}}
\end{bmatrix}
\]

- MS only points to itself
- Result: all weights go to MS!

\[
\begin{bmatrix}
    n \\
    m \\
    a
\end{bmatrix} = \begin{bmatrix}
    1 & 1 & 3/4 & 5/8 & 1/2 \\
    1 & 3/2 & 7/4 & 2 & 35/16 \\
    1 & 1/2 & 1/2 & 3/8 & 5/16
\end{bmatrix}
\begin{bmatrix}
    0 \\
    3 \\
    0
\end{bmatrix}
\]

\[\rightarrow_{\infty}\]

Ne

MS

Am
A revised solution with matrix notation: \( r = Ar + c \)

- **Matrix A**: web connectivity matrix
- **Portion of each page’s rank comes from a fixed weight** \( c \)
  - Example: 0.2.

\[
\begin{bmatrix}
  n \\
  m \\
  a
\end{bmatrix} = 0.8 \begin{pmatrix}
  1/2 & 0 & 1/2 \\
  0 & 1 & 1/2 \\
  1/2 & 0 & 0
\end{pmatrix}
\begin{bmatrix}
  n \\
  m \\
  a
\end{bmatrix} + \begin{bmatrix}
  0.2 \\
  0.2 \\
  0.2
\end{bmatrix}
\]

- **Rank result** \( r \) from iterative computation converges to

\[
\begin{bmatrix}
  n \\
  m \\
  a
\end{bmatrix} = \begin{bmatrix}
  7/11 \\
  21/11 \\
  5/11
\end{bmatrix}
\]