Analysis of Large Graphs: Link Analysis, PageRank

Slide source: Mining of Massive Datasets
Jure Leskovec, Anand Rajaraman, Jeff Ullman
Stanford University
http://www.mmds.org
Top-10 Algorithm Selected at ICDM’06

- #1: C4.5 Decision Tree - Classification (61 votes)
- #2: K-Means - Clustering (60 votes)
- #3: SVM – Classification (58 votes)
- #4: Apriori - Frequent Itemsets (52 votes)
- #5: EM – Clustering (48 votes)
- #6: PageRank – Link mining (46 votes)
- #7: AdaBoost – Boosting (45 votes)
- #7: kNN – Classification (45 votes)
- #7: Naive Bayes – Classification (45 votes)
- #10: CART – Classification (34 votes)
Link Analysis on WWW

- Ranking algorithms
  - PageRank
  - HITS
Graph Data: Social Networks

Facebook social graph
4-degrees of separation [Backstrom-Boldi-Rosa-Ugander-Vigna, 2011]
Graph Data: Information Nets

Citation networks and Maps of science
[Börner et al., 2012]
Web as a Directed Graph
How to organize the Web?

First try: Human curated Web directories
- Yahoo, DMOZ, LookSmart

Second try: Web Search
- Information Retrieval investigates:
  Find relevant docs in a small and trusted set
  - Newspaper articles, Patents, etc.
  - But: Web is huge, full of untrusted documents, random things, web spam, etc.
Ranking Nodes on the Graph

- All web pages are not equally “important”
  www.joe-schmoe.com vs. www.stanford.edu

- There is large diversity in the web-graph node connectivity.
  Let’s rank the pages by the link structure!
Links as Votes

- **Idea: Links as votes**
  - Page is more important if it has more links
    - In-coming links? Out-going links?
  - **Think of in-links as votes:**
    - [www.stanford.edu](http://www.stanford.edu) has 23,400 in-links
    - [www.joe-schmoe.com](http://www.joe-schmoe.com) has 1 in-link
  - **Are all in-links are equal?**
    - Links from important pages count more
    - Recursive question!
Example: PageRank Scores

A 3.3

B 38.4

C 34.3

D 3.9

E 8.1

F 3.9
Each link’s vote is proportional to the importance of its source page

If page $j$ with importance $r_j$ has $n$ out-links, each link gets $r_j/n$ votes

Page $j$’s own importance is the sum of the votes on its in-links

$$r_j = r_i/3 + r_k/4$$
PageRank: The “Flow” Model

- A “vote” from an important page is worth more
- A page is important if it is pointed to by other important pages
- Define a “rank” \( r_j \) for page \( j \)

\[
\sum_{i \rightarrow j} \frac{r_i}{d_i} = r_j
\]

\( d_i \ldots \) out-degree of node \( i \)

The web in 1839

“Flow” equations:

- \( r_y = \frac{r_y}{2} + \frac{r_a}{2} \)
- \( r_a = \frac{r_y}{2} + r_m \)
- \( r_m = \frac{r_a}{2} \)
Solving the Flow Equations

- **3 equations, 3 unknowns, no constants**
  - No unique solution
  - All solutions equivalent modulo the scale factor
- **Additional constraint forces uniqueness:**
  - \( r_y + r_a + r_m = 1 \)
  - **Solution:** \( r_y = \frac{2}{5}, \ r_a = \frac{2}{5}, \ r_m = \frac{1}{5} \)
- Gaussian elimination method works for small examples, but we need a better method for large web-size graphs
- **We need a new formulation!**

Flow equations:
- \( r_y = \frac{r_y}{2} + \frac{r_a}{2} \)
- \( r_a = \frac{r_y}{2} + r_m \)
- \( r_m = \frac{r_a}{2} \)
PageRank: Matrix Formulation

- **Stochastic adjacency matrix** $M$
  - Let page $i$ has $d_i$ out-links
  - If $i \to j$, then $M_{ji} = \frac{1}{d_i}$ else $M_{ji} = 0$
  - $M$ is a **column stochastic matrix**
    - Columns sum to 1
- **Rank vector** $r$: vector with an entry per page
  - $r_i$ is the importance score of page $i$
  - $\sum_i r_i = 1$
- The flow equations can be written
  \[
  r = M \cdot r
  \]
  \[
  r_j = \sum_{i \to j} \frac{r_i}{d_i}
  \]
Remember the flow equation: 

\[ r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i} \]

Flow equation in the matrix form:

\[ M \cdot r = r \]

Suppose page \( i \) links to 3 pages, including \( j \).
The flow equations can be written

\[ \mathbf{r} = \mathbf{M} \cdot \mathbf{r} \]

So the rank vector \( \mathbf{r} \) is an eigenvector of the stochastic web matrix \( \mathbf{M} \)

- In fact, its first or principal eigenvector, with corresponding eigenvalue \( 1 \)
  - Largest eigenvalue of \( \mathbf{M} \) is \( 1 \) since \( \mathbf{M} \) is column stochastic (with non-negative entries)
    - We know \( \mathbf{r} \) is unit length and each column of \( \mathbf{M} \) sums to one, so \( \mathbf{M} \mathbf{r} \leq 1 \)

We can now efficiently solve for \( \mathbf{r} \)!

The method is called Power iteration

**NOTE:** \( \mathbf{x} \) is an eigenvector with the corresponding eigenvalue \( \lambda \) if:

\[ \mathbf{A} \mathbf{x} = \lambda \mathbf{x} \]
Example: Flow Equations & M

\[ r = M \cdot r \]

\[
\begin{align*}
  r_y &= r_y / 2 + r_a / 2 \\
  r_a &= r_y / 2 + r_m \\
  r_m &= r_a / 2 \\
\end{align*}
\]

\[
\begin{array}{ccc}
  y & a & m \\
  \frac{1}{2} & \frac{1}{2} & 0 \\
  \frac{1}{2} & 0 & 1 \\
  0 & \frac{1}{2} & 0 \\
\end{array}
\]

Given a web graph with $n$ nodes, where the nodes are pages and edges are hyperlinks

**Power iteration:** a simple iterative scheme

- Suppose there are $N$ web pages
- Initialize: $r^{(0)} = [1/N,\ldots,1/N]^T$
- Iterate: $r^{(t+1)} = M \cdot r^{(t)}$
- Stop when $|r^{(t+1)} - r^{(t)}|_1 < \varepsilon$

$d_i$ .... out-degree of node $i$

$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i}$

$|x|_1 = \sum_{1 \leq i \leq N} |x_i|$ is the $L_1$ norm

Can use any other vector norm, e.g., Euclidean

---

PageRank: How to solve?

Example:

\[
\begin{pmatrix}
ry \\
r_a \\
r_m
\end{pmatrix} = \begin{pmatrix}
1/3 & 1/3 & 5/12 & 9/24 & 6/15 \\
1/3 & 3/6 & 1/3 & 11/24 & \ldots & 6/15 \\
1/3 & 1/6 & 3/12 & 1/6 & 3/15
\end{pmatrix}
\]

Iteration 0, 1, 2, …

\[
\begin{array}{ccccccc}
y & & a & & m & & \\
\hline
y & \frac{1}{2} & \frac{1}{2} & 0 & \\
a & \frac{1}{2} & 0 & 1 & \\
m & 0 & \frac{1}{2} & 0 & \\
\end{array}
\]
Imagine a random web surfer:
- At any time $t$, surfer is on some page $i$
- At time $t + 1$, the surfer follows an out-link from $i$ uniformly at random
- Ends up on some page $j$ linked from $i$
- Process repeats indefinitely

Let:
- $p(t)$ ... vector whose $i^{th}$ coordinate is the prob. that the surfer is at page $i$ at time $t$
- So, $p(t)$ is a probability distribution over pages
Where is the surfer at time $t+1$?

- Follows a link uniformly at random
  
  $$p(t + 1) = M \cdot p(t)$$

- Suppose the random walk reaches a state
  
  $$p(t + 1) = M \cdot p(t) = p(t)$$

  then $p(t)$ is **stationary distribution** of a random walk

- **Our original rank vector** $r$ satisfies
  
  $$r = M \cdot r$$

  So, $r$ is a stationary distribution for the random walk
PageRank: Questions

\[ r_{j}^{(t+1)} = \sum_{i \to j} \frac{r_{i}^{(t)}}{d_{i}} \]

or equivalently \( r = Mr \)

- Does this converge?
- Does it converge to what we want?
Does this converge?

Example:

\[ r_a = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix} \]

\[ r_b = \begin{pmatrix} \end{pmatrix} \]

Iteration 0, 1, 2, …
Does it converge to what we want?

Example:

\[
\begin{align*}
\mathbf{r}_a &= \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix} \\
\mathbf{r}_b &= \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \end{pmatrix}
\end{align*}
\]

Iteration 0, 1, 2, …

\[
\mathbf{r}_j^{(t+1)} = \sum_{i \rightarrow j} \frac{\mathbf{r}_i^{(t)}}{d_i}
\]
A central result from the theory of random walks (a.k.a. Markov processes):

For graphs that satisfy certain conditions (strong connected, no dead ends) the stationary distribution is unique and eventually will be reached no matter what the initial probability distribution at time $t = 0$. 
The web

- Tendrils Out
- Tendrils In
- In Component
- Out Component
- Strongly Connected Component
- Tubes
- Disconnected Components
PageRank: Problems

2 problems:

1. Some pages are **dead ends** (have no out-links)
   - Random walk has “nowhere” to go to
   - Such pages cause importance to “leak out”

2. **Spider traps**: (all out-links are within the group)
   - Random walked gets “stuck” in a trap
   - And eventually spider traps absorb all importance
Problem: Spider Traps

Example:

\[
\begin{pmatrix}
  r_y \\
r_a \\
r_m
\end{pmatrix} =
\begin{pmatrix}
  1/3 & 2/6 & 3/12 & 5/24 & 0 \\
  1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\
  1/3 & 3/6 & 7/12 & 16/24 & 1
\end{pmatrix}
\]

Iteration 0, 1, 2, …

All the PageRank score gets “trapped” in node m.
The Google solution for spider traps: At each time step, the random surfer has two options

- With prob. $\beta$, follow a link at random
- With prob. $1-\beta$, jump to some random page
- Common values for $\beta$ are in the range 0.8 to 0.9

Surfer will teleport out of spider trap within a few time steps
**Problem: Dead Ends**

Example:

\[
\begin{bmatrix}
 r_y \\
 r_a \\
 r_m 
\end{bmatrix} =
\begin{bmatrix}
 1/3 & 2/6 & 3/12 & 5/24 & 0 \\
 1/3 & 1/6 & 2/12 & 3/24 & \ldots & 0 \\
 1/3 & 1/6 & 1/12 & 2/24 & 0
\end{bmatrix}
\]

Iteration 0, 1, 2, …

Here the PageRank “leaks” out since the matrix is not stochastic.
**Solution: Always Teleport!**

- **Teleports:** Follow random teleport links with probability 1.0 from dead-ends
  - Adjust matrix accordingly

```
<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1/2</td>
<td>1/2</td>
<td>0</td>
</tr>
<tr>
<td>a</td>
<td>1/2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>1/2</td>
<td>0</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>y</th>
<th>a</th>
<th>m</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td>1/2</td>
<td>1/2</td>
<td>1/3</td>
</tr>
<tr>
<td>a</td>
<td>1/2</td>
<td>0</td>
<td>1/3</td>
</tr>
<tr>
<td>m</td>
<td>0</td>
<td>1/2</td>
<td>1/3</td>
</tr>
</tbody>
</table>
```
Why are dead-ends and spider traps a problem and why do teleports solve the problem?

- **Spider-traps** are not a problem, but with traps PageRank scores are **not** what we want
  - **Solution:** Never get stuck in a spider trap by teleporting out of it in a finite number of steps

- **Dead-ends** are a problem
  - The matrix is not column stochastic so our initial assumptions are not met
  - **Solution:** Make matrix column stochastic by always teleporting when there is nowhere else to go
Solution: Random Teleports

- **Google’s solution that does it all:**
  At each step, random surfer has two options:
  - With probability $\beta$, follow a link at random
  - With probability $1 - \beta$, jump to some random page

- **PageRank equation** [Brin-Page, 98]

$$r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N}$$

$d_i$ ... out-degree of node $i$

This formulation assumes that $M$ has no dead ends. We can either preprocess matrix $M$ to remove all dead ends or explicitly follow random teleport links with probability 1.0 from dead-ends.
The Google Matrix

- **PageRank equation** [Brin-Page, ‘98]
  \[ r_j = \sum_{i \rightarrow j} \beta \frac{r_i}{d_i} + (1 - \beta) \frac{1}{N} \]

- **The Google Matrix** \( A \):
  \[ A = \beta M + (1 - \beta) \left[ \frac{1}{N} \right]_{N \times N} \]

- **We have a recursive problem:**
- **And the Power method still works!**
- **What is \( \beta \)?**
  - In practice \( \beta = 0.8, 0.9 \) (make 5 steps on avg., jump)
Random Teleports ($\beta = 0.8$)

\[
\begin{align*}
M &= \begin{pmatrix}
1/2 & 1/2 & 0 \\
1/2 & 0 & 0 \\
0 & 1/2 & 1 \\
\end{pmatrix} + 0.2 \\
\begin{pmatrix}
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/3 \\
\end{pmatrix} \\
A &= \begin{pmatrix}
y & 7/15 & 7/15 & 1/15 \\
a & 7/15 & 1/15 & 1/15 \\
m & 1/15 & 7/15 & 13/15 \\
\end{pmatrix} \\
\begin{pmatrix}
y & 1/3 & 0.33 & 0.24 & 0.26 & 7/33 \\
a & 1/3 & 0.20 & 0.20 & 0.18 & \ldots & 5/33 \\
m & 1/3 & 0.46 & 0.52 & 0.56 & 21/33 \\
\end{pmatrix}
\end{align*}
\]
PageRank: The Complete Algorithm

- **Input:** Graph $G$ and parameter $\beta$
  - Directed graph $G$ (can have spider traps and dead ends)
  - Parameter $\beta$
- **Output:** PageRank vector $r^{new}$

- **Set:** $r_j^{old} = \frac{1}{N}$

- **repeat until convergence:** $\sum_j |r_j^{new} - r_j^{old}| > \varepsilon$

  - $\forall j$: $r_j^{new} = \sum_{i \rightarrow j} \beta \frac{r_i^{old}}{d_i}$
  - $r_j^{new} = 0$ if in-degree of $j$ is 0

  - **Now re-insert the leaked PageRank:**
    - $\forall j$: $r_j^{new} = r_j^{new} + \frac{1-S}{N}$ \text{ where: } S = \sum_j r_j^{new}$

  - $r^{old} = r^{new}$

If the graph has no dead-ends then the amount of leaked PageRank is $1-\beta$. But since we have dead-ends the amount of leaked PageRank may be larger. We have to explicitly account for it by computing $S$.  

### Sparse Matrix Encoding

- **Encode sparse matrix using only nonzero entries**
  - Space proportional roughly to number of links
  - Say $10N$, or $4 \times 10 \times 1$ billion = 40GB
  - Still won’t fit in memory, but will fit on disk

<table>
<thead>
<tr>
<th>source node</th>
<th>degree</th>
<th>destination nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
<td>1, 5, 7</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>17, 64, 113, 117, 245</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>13, 23</td>
</tr>
</tbody>
</table>
Power Iteration

- What if $v$ cannot fit into memory?
- Strip method
- Partitioning method

![Diagram showing matrix $M$ and vector $v$](image)
Some Problems with Page Rank

- Measures generic popularity of a page
  - Biased against topic-specific authorities
  - **Solution:** Topic-Specific PageRank

- Uses a single measure of importance
  - Other models of importance
  - **Solution:** Hubs-and-Authorities

- Susceptible to Link spam
  - Artificial link topographies created in order to boost page rank
  - **Solution:** TrustRank