PageRank: Ranking of nodes in graphs

Gonzalo Mateos
Dept. of Electrical and Computer Engineering
University of Rochester
gmateosb@ece.rochester.edu
http://www.ece.rochester.edu/~gmateosb/

October 19, 2015
PageRank: Random walk

Ranking of nodes in graphs: Random walk

Ranking of nodes in graphs: Probability propagation
Graphs

- **Graph** ⇒ A set of $V$ of vertices or nodes $j = 1, \ldots, J$
  ⇒ Connected by a set of edges $E$ defined as ordered pairs $(i, j)$

- In figure ⇒ Nodes are $V = \{1, 2, 3, 4, 5, 6\}$
  ⇒ Edges $E = \{(1, 2), (1, 5), (2, 3), (2, 5), (3, 4), \ldots (3, 6), (4, 5), (4, 6), (5, 4)\}$

- **Ex. 1**: Websites and hyperlinks ⇒ World Wide Web (WWW)

- **Ex. 2**: People and friendship ⇒ Social network
How well connected nodes are?

Q: Which node is the most connected? A: Define most connected
   ⇒ Can define “most connected” in different ways

Two important connectivity indicators
   1) How many links point to a node (outgoing links irrelevant)
   2) How important are the links that point to a node

Node rankings to measure website relevance, social influence
Connectivity ranking

- **Key insight:** There is information in the structure of the network
- Knowledge is distributed through the network
  \[\Rightarrow \text{The network (not the nodes) knows the rankings}\]

- Idea exploited by Google's PageRank© to rank webpages
  ... by social scientists to study trust & reputation in social networks
  ... by ISI to rank scientific papers, transactions & magazines ...

- No one points to 1
- Only 1 points to 2
- Only 2 points to 3, but 2 more important than 1
- 4 as high as 5 with less links
- Links to 5 have lower rank
- Same for 6
Preliminary definitions

- Graph $G = (V, E)$ \Rightarrow vertices $V = \{1, 2, \ldots, J\}$ and edges $E$

- Outgoing neighborhood of $i$ is the set of nodes $j$ to which $i$ points

  $$n(i) := \{j : (i, j) \in E\}$$

- Incoming neighborhood, $n^{-1}(i)$ is the set of nodes that point to $i$:

  $$n^{-1}(i) := \{j : (j, i) \in E\}$$

- Strongly connected $G \Rightarrow$ directed path joining any pair of nodes
Definition of rank

- Agent A chooses node $i$, e.g., web page, at random for initial visit
- Next visit randomly chosen between links in the neighborhood $n(i)$
  - All neighbors chosen with equal probability
- If reach a dead end because node $i$ has no neighbors
  - Chose next visit at random equiprobably among all nodes
- Redefine graph $\mathcal{G} = (V, E)$ adding edges from dead ends to all nodes
  - Restrict attention to connected (modified) graphs

- Rank of node $i$ is the average number of visits of agent A to $i$
Formally, let $A_n$ be the node visited at time $n$

Define transition probability $P_{ij}$ from node $i$ into node $j$

$$P_{ij} := P(A_{n+1} = j \mid A_n = i)$$

Next visit equiprobable among $i$’s $N_i := |n(i)|$ neighbors

$$P_{ij} = \frac{1}{|n(i)|} = \frac{1}{N_i}, \text{ for all } j \in n(i)$$

Still have a graph

But also a MC

Red (not blue) circles
Formal definition of rank

► **Def:** Rank $r_i$ of $i$-th node is the time average of number of visits

$$ r_i := \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{I} \{ A_m = i \} $$

⇒ Define vector of ranks $\mathbf{r} := [r_1, r_2, \ldots, r_J]^T$

► Rank $r_i$ can be approximated by average $r_{ni}$ at time $n$

$$ r_{ni} := \frac{1}{n} \sum_{m=1}^{n} \mathbb{I} \{ A_m = i \} $$

⇒ Since $\lim_{n \to \infty} r_{ni} = r_i$, it holds $r_{ni} \approx r_i$ for $n$ sufficiently large

⇒ Define vector of approximate ranks $\mathbf{r}_n := [r_{n1}, r_{n2}, \ldots, r_{nJ}]^T$

► If modified graph is connected, rank independent of initial visit
Output: Vector $r(i)$ with ranking of node $i$
Input: Scalar $n$ indicating maximum number of iterations
Input: Vector $N(i)$ containing number of neighbors of $i$
Input: Matrix $N(i,j)$ containing indices $j$ of neighbors of $i$

$m = 1$; $r=\text{zeros}(J,1)$; % Initialize time and ranks
$A_0 = \text{random}\left(\text{‘unid’},J\right)$; % Draw first visit uniformly at random

while $m < n$ do
    jump = random(‘unid’,$N_{A_{m-1}}$); % Neighbor uniformly at random
    $A_m = N(A_{m-1},\text{jump})$; % Jump to selected neighbor
    $r(A_m) = r(A_m) + 1$; % Update ranking for $A_m$
    $m = m + 1$;
end

$r = r/n$; % Normalize by number of iterations $n$
Social graph example

- Asked probability students about homework collaboration
- Created (crude) graph of the social network of students in the class
  ⇒ Used ranking algorithm to understand connectedness
- **Ex:** I want to know how well students are coping with the class
  ⇒ Best to ask people with higher connectivity ranking
- 2009 data from “UPenn’s ECE440”
Introduction to Random Processes

Ranking of nodes in graphs
Convergence metrics

- Recall \( \mathbf{r} \) is vector of ranks and \( \mathbf{r}_n \) of rank iterates
- By definition \( \lim_{n \to \infty} \mathbf{r}_n = \mathbf{r} \). How fast \( \mathbf{r}_n \) converges to \( \mathbf{r} \) (\( \mathbf{r} \) given)?
- Can measure by \( \ell_2 \) distance between \( \mathbf{r} \) and \( \mathbf{r}_n \)

\[
\zeta_n := \| \mathbf{r} - \mathbf{r}_n \|_2 = \left( \sum_{i=1}^{J} (r_{ni} - r_i)^2 \right)^{1/2}
\]

- If interest is only on highest ranked nodes, e.g., a web search
  ⇒ Denote \( r^{(i)} \) as the index of the \( i \)-th highest ranked node
  ⇒ Let \( r_n^{(i)} \) be the index of the \( i \)-th highest ranked node at time \( n \)
- First element wrongly ranked at time \( n \)

\[
\xi_n := \arg \min_i \{ r^{(i)} \neq r_n^{(i)} \}
\]
Evaluation of convergence metrics

- **Distance close to** $10^{-2}$ in $\approx 5 \times 10^3$ iterations
- **Bad**: Two highest ranks in $\approx 4 \times 10^3$ iterations
- **Awful**: Six best ranks in $\approx 8 \times 10^3$ iterations
- **(Very) slow convergence**
When does this algorithm converge?

- Cannot confidently claim convergence until $10^5$ iterations
  - Beyond particular case, slow convergence inherent to algorithm

- Example has 40 nodes, want to use in network with $10^9$ nodes!
  - Leverage properties of MCs to obtain a faster algorithm
Ranking of nodes in graphs: Random walk

Ranking of nodes in graphs: Probability propagation
Limit probabilities

- Recall definition of rank \( r_i := \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{I} \{A_m = i\} \)

- Rank is time average of number of state visits in a MC
  \( \Rightarrow \) Can be as well obtained from limiting probabilities

- Recall transition probabilities \( P_{ij} = \frac{1}{N_i} \), for all \( j \in n(i) \)

- Stationary distribution \( \pi = [\pi_1, \pi_1, \ldots, \pi_J]^T \) solution of
  \[
  \pi_i = \sum_{j \in n^{-1}(i)} P_{ji} \pi_j = \sum_{j \in n^{-1}(i)} \frac{\pi_j}{N_j} \quad \text{for all } i
  \]
  \( \Rightarrow \) Plus normalization equation \( \sum_{i=1}^{J} \pi_i = 1 \)

- As per ergodicity of MC (strongly connected \( G \)) \( \Rightarrow r = \pi \)
As always, can define matrix $P$ with elements $P_{ij}$

$$\pi_i = \sum_{j \in n^{-1}(i)} P_{ji} \pi_j = \sum_{j=1}^{J} P_{ji} \pi_j \quad \text{for all } i$$

Right hand side is just definition of a matrix product leading to

$$\pi = P^T \pi, \quad \pi^T 1 = 1$$

Also added normalization equation

Idea: solve system of linear equations or eigenvalue problem on $P^T$

Requires matrix $P$ available at a central location

Computationally costly (sparse matrix $P$ with $10^{18}$ entries)
What are limit probabilities?

- Let \( p_i(n) \) denote probability of agent \( A \) visiting node \( i \) at time \( n \)
  \[
  p_i(n) := P(A_n = i)
  \]
- Probabilities at time \( n + 1 \) and \( n \) can be related
  \[
  P(A_{n+1} = i) = \sum_{j \in n^{-1}(i)} P(A_{n+1} = i \mid A_n = j) P(A_n = j)
  \]
- Which is, of course, probability propagation in a MC
  \[
  p_i(n + 1) = \sum_{j \in n^{-1}(i)} P_{ji} p_j(n)
  \]
- By definition limit probabilities are (let \( \mathbf{p}(n) = [p_1(n), \ldots, p_J(n)]^T \))
  \[
  \lim_{n \to \infty} \mathbf{p}(n) = \pi = \mathbf{r}
  \]
  \( \Rightarrow \) Compute ranks from limit of propagated probabilities
Can also write probability propagation in matrix form

\[ p_i(n + 1) = \sum_{j \in n^{-1}(i)} P_{ji}p_j(n) = \sum_{j=1}^{J} P_{ji}p_j(n) \quad \text{for all } i \]

Right hand side is just definition of a matrix product leading to

\[ p(n + 1) = P^T p(n) \]

Idea: can approximate rank by large \( n \) probability distribution

\[ r = \lim_{n \to \infty} p(n) \approx p(n) \quad \text{for } n \text{ sufficiently large} \]
Algorithm is just a recursive matrix product, a power iteration

Output: Vector $r(i)$ with ranking of node $i$
Input: Scalar $n$ indicating maximum number of iterations
Input: Matrix $P$ containing transition probabilities

$m = 1$; % Initialize time
$r = (1/J) \text{ones}(J,1)$; % Initial distribution uniform across all nodes
while $m < n$ do
  $r = P^T r$; % Probability propagation
  $m = m + 1$;
end
Q: Why does the random walk converge so slow?

A: Need to register a large number of agent visits to every state

Ex: 40 nodes, say 100 visits to each \( \Rightarrow 4 \times 10^3 \) iters.

Smart idea: Unleash a large number of agents \( K \)

\[
    r_i = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \frac{1}{K} \sum_{k=1}^{K} \mathbb{I} \{A_{km} = i\}
\]

Visits are now spread over time and space

\( \Rightarrow \) Converges “\( K \) times faster”

\( \Rightarrow \) But haven’t changed computational cost
Interpretation of prob. propagation (continued)

- **Q:** What happens if we unleash infinite number of agents $K$?

  \[ r_i = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \lim_{K \to \infty} \frac{1}{K} \sum_{k=1}^{K} \mathbb{I}\{A_{km} = i\} \]

- Using law of large numbers and expected value of indicator function

  \[ r_i = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{E}\left[ \mathbb{I}\{A_m = i\} \right] = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} P(A_m = i) \]

- Graph walk is an ergodic MC, then \( \lim_{m \to \infty} P(A_m = i) \) exists, and

  \[ r_i = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} p_i(m) = \lim_{n \to \infty} p_i(n) \]

  \[ \Rightarrow \text{Probability propagation} \approx \text{Unleashing infinitely many agents} \]
Distance to rank

- Initialize with uniform probability distribution \( \Rightarrow p(0) = (1/J)1 \)
- \( \Rightarrow \) Plot distance between \( p(n) \) and \( r \)

Distance is \( 10^{-2} \) in \( \approx 30 \) iters., \( 10^{-4} \) in \( \approx 140 \) iters.

\( \Rightarrow \) Convergence two orders of magnitude faster than random walk
Number of nodes correctly ranked

- Rank of highest ranked node that is wrongly ranked by time $n$

- Not bad: All nodes correctly ranked in 120 iterations
- Good: Ten best ranks in 70 iterations
- Great: Four best ranks in 20 iterations
Distributed algorithm to compute ranks

- Nodes want to compute their rank $r_i$
  - Can communicate with neighbors only (incoming + outgoing)
  - Access to neighborhood information only

- Recall probability update
  $$p_i(n + 1) = \sum_{j \in n^{-1}(i)} P_{ji}p_j(n) = \sum_{j \in n^{-1}(i)} \frac{1}{N_j} p_j(n)$$
  - Uses local information only

- Distributed algorithm. Nodes keep local rank estimates $r_i(n)$
  - Receive rank (probability) estimates $r_j(n)$ from neighbors $j \in n^{-1}(i)$
  - Update local rank estimate $r_i(n + 1) = \sum_{j \in n^{-1}(i)} r_j(n)/N_j$
  - Communicate rank estimate $r_i(n + 1)$ to outgoing neighbors $j \in n(i)$

- Only need to know the number of neighbors of my neighbors
Distributed implementation of random walk

- Can communicate with neighbors only (incoming + outgoing)
  - But cannot access neighborhood information
  - Pass agent (‘hot potato’) around

- Local rank estimates $r_i(n)$ and counter with number of visits $V_i$

- Algorithm run by node $i$ at time $n$

```
if Agent received from neighbor then
  $V_i = V_i + 1$
  Choose random neighbor
  Send agent to chosen neighbor
end

$n = n + 1; \ r_i(n) = V_i / n$;
```

- Speed up convergence by generating many agents to pass around
Comparison of different algorithms

- **Random walk (RW) implementation**
  - Most secure. No information shared with other nodes
  - Implementation can be distributed
  - Convergence exceedingly slow

- **System of linear equations**
  - Least security. Graph in central server
  - Distributed implementation not clear
  - Convergence not an issue
  - But computationally costly to obtain approximate solutions

- **Probability propagation**
  - Somewhat secure. Information shared with neighbors only
  - Implementation can be distributed
  - Convergence rate acceptable (orders of magnitude faster than RW)
Glossary

- Graph, nodes and edges
- Connectivity indicators
- Node ranking
- Google's PageRank
- Node's neighborhood
- Strong connectivity
- Random walk on a graph
- Long-run fraction of state visits

- Ranking algorithm
- Convergence metrics
- Computational cost
- Probability propagation
- Power method
- Distributed algorithm
- Security