Centrality Measures and Link Analysis

Gonzalo Mateos
Dept. of ECE and Goergen Institute for Data Science
University of Rochester
gmateosb@ece.rochester.edu
http://www.ece.rochester.edu/~gmateosb/

February 18, 2016
Centrality measures

Case study: Stability of centrality measures in weighted graphs

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
In network analysis many questions relate to vertex importance.

Example

- **Q1:** Which actors in a social network hold the ‘reins of power’?
- **Q2:** How authoritative is a WWW page considered by peers?
- **Q3:** The ‘knock-out’ of which genes is likely to be lethal?
- **Q4:** How critical to the daily commute is a subway station?

- Measures of vertex centrality quantify such notions of importance.
  - Degrees are simplest centrality measures. Let’s study others.
Closeness centrality

- **Rationale:** ‘central’ means a vertex is ‘close’ to many other vertices

- **Def:** Distance $d(u, v)$ between vertices $u$ and $v$ is the length of the shortest $u - v$ path. Oftentimes referred to as geodesic distance

- **Closeness centrality** of vertex $v$ is given by

$$c_{Cl}(v) = \frac{1}{\sum_{u \in V} d(u, v)}$$

- Interpret $v^* = \arg \max_v c_{Cl}(v)$ as the most approachable node in $G$
Normalization, computation and limitations

- To compare with other centrality measures, often normalize to \([0, 1]\)

\[
c_{CI}(v) = \frac{N_v - 1}{\sum_{u \in V} d(u, v)}
\]

- **Computation:** need all pairwise shortest path distances in \(G\)
  \(\Rightarrow\) Dijkstra’s algorithm in \(O(N_v^2 \log N_v + N_v N_e)\) time

- **Limitation 1:** sensitivity, values tend to span a small dynamic range
  \(\Rightarrow\) Hard to discriminate between central and less central nodes

- **Limitation 2:** assumes connectivity, if not \(c_{CI}(v) = 0\) for all \(v \in V\)
  \(\Rightarrow\) Compute centrality indices in different components
Betweenness centrality

- **Rationale:** ‘central’ node is (in the path) ‘between’ many vertex pairs

- **Betweenness centrality** of vertex $v$ is given by

$$c_{Be}(v) = \sum_{s \neq t \neq v \in V} \frac{\sigma(s, t \mid v)}{\sigma(s, t)}$$

- $\sigma(s, t)$ is the total number of $s - t$ shortest paths
- $\sigma(s, t \mid v)$ is the number of $s - t$ shortest paths through $v \in V$

- Interpret $v^* = \arg \max_v c_{Be}(v)$ as the **controller of information flow**
Notice that a $s - t$ shortest path goes through $v$ if and only if
\[ d(s, t) = d(s, v) + d(v, t) \]

Betweenness centralities can be naively computed for all $v \in V$ by:

1. Use Dijkstra to tabulate $d(s, t)$ and $\sigma(s, t)$ for all $s, t$
2. Use the tables to identify $\sigma(s, t|v)$ for all $v$
3. Sum the fractions to obtain $c_{Be}(v)$ for all $v$ ($O(N_v^3)$ time)

Cubic complexity can be prohibitive for large networks

$O(N_vN_e)$-time algorithm for unweighted graphs in:

Eigenvector centrality

- **Rationale**: ‘central’ vertex if ‘in-neighbors’ are themselves important
  \[ \Rightarrow \text{Compare with ‘importance-agnostic’ degree centrality} \]

- **Eigenvector centrality** of vertex \( v \) is implicitly defined as
  \[
  c_{Ei}(v) = \alpha \sum_{(u,v) \in E} c_{Ei}(u)
  \]

- 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
Recall the adjacency matrix $A$ and

$$c_{Ei}(v) = \alpha \sum_{(u,v) \in E} c_{Ei}(u)$$

Vector $c_{Ei} = [c_{Ei}(1), \ldots, c_{Ei}(N_v)]^\top$ solves the eigenvalue problem

$$Ac_{Ei} = \alpha^{-1} c_{Ei}$$

⇒ Typically $\alpha^{-1}$ chosen as largest eigenvalue of $A$ [Bonacich'87]

If $G$ is undirected and connected, by Perron’s Theorem then

⇒ The largest eigenvalue of $A$ is positive and simple
⇒ All the entries in the dominant eigenvector $c_{Ei}$ are positive

Can compute $c_{Ei}$ and $\alpha^{-1}$ via $O(N_v^2)$ complexity power iterations

$$c_{Ei}(k+1) = \frac{Ac_{Ei}(k)}{\|Ac_{Ei}(k)\|}, \quad k = 0, 1, \ldots$$
Example: Comparing centrality measures

- Q: Which vertices are more central? A: It depends on the context

- Each measure identifies a different vertex as most central
  ⇒ None is ‘wrong’, they target different notions of importance
Example: Comparing centrality measures

- Q: Which vertices are more central? A: It depends on the context

- Small green vertices are arguably more peripheral
  ⇒ Less clear how the yellow, dark blue and red vertices compare
Case study

Centrality measures

Case study: Stability of centrality measures in weighted graphs

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
Centrality measures robustness

- Robustness to noise in network data is of practical importance
- Approaches have been mostly **empirical**
  - Find average response in random graphs when perturbed
  - Not generalizable and does not provide explanations
- Characterize behavior in **noisy real graphs**
  - *Degree and closeness are more reliable than betweenness*
- **Q:** What is really going on?
  - Framework to study formally the stability of centrality measures
Definitions for weighted digraphs

- **Weighted** and **directed** graphs $G(V, E, W)$
  - Set $V$ of $N_v$ vertices
  - Set $E \subseteq V \times V$ of edges
  - Map $W : E \to \mathbb{R}_{++}$ of weights in each edge

- Path $P(u, v)$ is an ordered sequence of nodes from $u$ to $v$

- When weights represent dissimilarities
  - **Path length** is the sum of the dissimilarities encountered

- **Shortest path length** $s_G(u, v)$ from $u$ to $v$

$$s_G(u, v) := \min_{P(u,v)} \sum_{i=0}^{\ell-1} W(u_i, u_{i+1})$$
Stability of centrality measures

- Space of graphs $G_{(V,E)}$ with $(V,E)$ as vertex and edge set

- Define the metric $d_{(V,E)}(G,H) : G_{(V,E)} \times G_{(V,E)} \to \mathbb{R}_+$

  $$d_{(V,E)}(G,H) := \sum_{e \in E} |W_G(e) - W_H(e)|$$

- **Def:** A centrality measure $c(\cdot)$ is **stable** if for any vertex $v \in V$ in any two graphs $G, H \in G_{(V,E)}$, then

  $$|c^G(v) - c^H(v)| \leq K_G d_{(V,E)}(G,H)$$

  - $K_G$ is a constant depending on $G$ only
  - Stability is related to **Lipschitz continuity** in $G_{(V,E)}$
  - Independent of the definition of $d_{(V,E)}$ (equivalence of norms)

- Node importance should be robust to small perturbations in the graph
Degree centrality

- Sum of the weights of incoming arcs

\[ c_{De}(v) := \sum_{u|(u,v) \in E} W(u, v) \]

- Applied to graphs where the weights in \( W \) represent similarities
- High \( c_{De}(v) \) \( \Rightarrow \) \( v \) similar to its large number of neighbors

**Proposition 1**

For any vertex \( v \in V \) in any two graphs \( G, H \in \mathcal{G}(V, E) \), we have that

\[ |c_{De}^G(v) - c_{De}^H(v)| \leq d_{(V,E)}(G, H) \]

i.e., degree centrality \( c_{De} \) is a stable measure

- Can show closeness and eigenvector centralities are also stable
Betweenness centrality

- Look at the shortest paths for every two nodes distinct from $v$
  - Sum the proportion that contains node $v$

$$c_{Be}(v) := \sum_{s \neq v \neq t \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)}$$

- $\sigma(s, t)$ is the total number of $s - t$ shortest paths
- $\sigma(s, t|v)$ is the number of those paths going through $v$

**Proposition 2**

The betweenness centrality measure $c_{Be}$ is not stable
Instability of betweenness centrality

- Compare the value of $c_{Be}(v)$ in graphs $G$ and $H$

  $c^G_{Be}(v) = 9$

  $c^H_{Be}(v) = 0$

  $\Rightarrow$ Centrality value $c^H_{Be}(v) = 0$ remains unchanged for any $\epsilon > 0$

- For small values of $\epsilon$, graphs $G$ and $H$ become arbitrarily similar

  $9 = |c^G_{Be}(v) - c^H_{Be}(v)| \leq K_G d_{(V,E)}(G, H) \to 0$

  $\Rightarrow$ Inequality is not true for any constant $K_G$
Stable betweenness centrality

- Define $G^v = (V^v, E^v, W^v)$, $V^v = V \setminus \{v\}$, $E^v = E|_{V^v \times V^v}$, $W^v = W|_{E^v \times E^v}$
  
  $\Rightarrow$ $G^v$ obtained by deleting from $G$ node $v$ and edges connected to $v$

- Stable betweenness centrality $c_{SBe}(v)$

  $$c_{SBe}(v) := \sum_{s \neq v \neq t \in V} s_{G^v}(s, t) - s_G(s, t)$$

  $\Rightarrow$ Captures impact of deleting $v$ on the shortest paths

- If $v$ is (not) in the $s - t$ shortest path, $s_{G^v}(s, t) - s_G(s, t) > (\leq) 0$

  $\Rightarrow$ Same notion as (traditional) betweenness centrality $c_{Be}$

**Proposition 3**

For any vertex $v \in V$ in any two graphs $G, H \in \mathcal{G}(V, E)$, then

$$|c_{SBe}^G(v) - c_{SBe}^H(v)| \leq 2N_v^2 \cdot d_{(V, E)}(G, H)$$

i.e., stable betweenness centrality $c_{SBe}$ is a stable measure
Centrality ranking variation in random graphs

- $G_{n,p}$ graphs with $p = 10/n$ and weights $U(0.5, 1.5)$
  - Vary $n$ from 10 to 200
  - Perturb multiplying weights with random numbers $U(0.99, 1.01)$

- Compare centrality rankings in the original and perturbed graphs

- Betweenness centrality presents larger maximum and average changes
Centrality ranking variation in random graphs

- Compute probability of observing a ranking change ≥ 5
  ⇒ Plot the histogram giving rise to the empirical probabilities

- For $c_{Be}$ some node varies its ranking by 5 positions with high probability

- Long tail in histogram is evidence of instability
  ⇒ Minor perturbation generates change of 19 positions
Centrality ranking variation in an airport graph

- Real-world graph based on the air traffic between popular U.S. airports
  - Nodes are $N_v = 25$ popular airports
  - Edge weights are the number of yearly passengers between them

- Betweenness centrality still presents the largest variations
Centrality measures

Case study: Stability of centrality measures in weighted graphs

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
The problem of ranking websites

- Search engines rank pages by looking at the Web itself
  - Enough information intrinsic to the Web and its structure

- Information retrieval is a historically difficult problem
  - Keywords vs complex information needs (synonymy, polysemy)

- Beyond explosion in scale, unique issues arised with the Web
  - Diversity of authoring styles, people issuing queries
  - Dynamic and constantly changing content
  - Paradigm: from scarcity to abundance

- Finding and indexing documents that are relevant is ‘easy’

- Q: Which few of these should the engine recommend?
  - Key is understanding Web structure, i.e., link analysis
Voting by in-links

Ex: Suppose we issue the query ‘newspapers’

► First, use text-only information retrieval to identify relevant pages

► Idea: Links suggest implicit endorsements of other relevant pages
  ► Count in-links to assess the authority of a page on ‘newspapers’
A list-finding technique

- Query also returns pages that compile lists of relevant resources
  - These hubs voted for many highly endorsed (authoritative) pages

- Idea: Good lists have a better sense of where the good results are
  - Page’s hub value is the sum of votes received by its linked pages
Repeated improvement

- Reasonable to weight more the votes of pages scoring well as lists
  ⇒ Recompute votes summing linking page values as lists

- Q: Why stop here? Use also improved votes to refine the list scores
  ⇒ Principle of repeated improvement
Hubs and authorities

- Relevant pages fall in two categories: hubs and authorities

- **Authorities** are pages with useful, relevant content
  - Newspaper home pages
  - Course home pages
  - Auto manufacturer home pages

- **Hubs** are ‘expert’ lists pointing to multiple authorities
  - List of newspapers
  - Course bulletin
  - List of US auto manufacturers

- **Rules:** Authorities and hubs have a mutual reinforcement relationship
  - ⇒ A good **hub** links to multiple good **authorities**
  - ⇒ A good **authority** is linked from multiple good **hubs**
Hubs and authorities ranking algorithm

- Hyperlink-Induced Topic Search (HITS) algorithm [Kleinberg’98]

- Each page $v \in V$ has a hub score $h_v$ and authority score $a_v$

  ⇒ Network-wide vectors $h = [h_1, \ldots, h_{N_v}]^\top$, $a = [a_1, \ldots, a_{N_v}]^\top$

  **Authority update rule:**

  $$a_v(k) = \sum_{(u,v) \in E} h_u(k-1), \text{ for all } v \in V \Leftrightarrow a(k) = A^\top h(k-1)$$

  **Hub update rule:**

  $$h_v(k) = \sum_{(v,u) \in E} a_u(k), \text{ for all } v \in V \Leftrightarrow h(k) = Aa(k)$$

- Initialize $h(0) = 1/\sqrt{N_v}$, normalize $a(k)$ and $h(k)$ each iteration
Define the hub and authority rankings as
\[ a := \lim_{k \to \infty} a(k), \quad h := \lim_{k \to \infty} h(k) \]

From the HITS update rules one finds for \( k = 0, 1, \ldots \)
\[ a(k + 1) = \frac{A^\top Aa(k)}{\|A^\top Aa(k)\|}, \quad h(k + 1) = \frac{AA^\top h(k)}{\|AA^\top h(k)\|} \]

Power iterations converge to dominant eigenvectors of \( A^\top A \) and \( AA^\top \)
\[ A^\top Aa = \alpha_a^{-1}a, \quad AA^\top h = \alpha_h^{-1}h \]

⇒ Hub and authority ranks are eigenvector centrality measures
Ex: link analysis of citations among US Supreme Court opinions

- Rise and fall of authority of key Fifth Amendment cases [Fowler-Jeon’08]
PageRank

- **Node rankings** to measure website relevance, social influence

- **Key idea:** in-links as votes, but ‘not all links are created equal’
  - How many links point to a node (outgoing links irrelevant)
  - How important are the links that point to a node

- **PageRank** key to Google's original ranking algorithm [Page-Brin’98]

- **Intuition 1:** fluid that percolates through the network
  - Eventually accumulates at most relevant Web pages

- **Intuition 2:** random web surfer *(more soon)*
  - In the long-run, relevant Web pages visited more often

- PageRank and HITS success was quite different after 1998
Basic PageRank update rule

- Each page $v \in V$ has PageRank $r_v$, let $r = [r_1, \ldots, r_N]^\top$
  - Define $P := (D^{out})^{-1}A$, where $D^{out}$ is the out-degree matrix

**PageRank update rule:**

$$r_v(k) = \sum_{(u,v) \in E} \frac{r_u(k-1)}{d_{out}^u}, \text{ for all } v \in V \iff r(k) = P^T r(k-1)$$

- Split current PageRank evenly among outgoing links and pass it on
  - New PageRank is the total fluid collected in the incoming links
  - Initialize $r(0) = 1/N_v$. Flow conserved, no normalization needed

- **Problem:** ‘Spider traps’
  - Accumulate all PageRank
  - Only when not strongly connected
Apply the basic PageRank rule and scale the result by $s \in (0, 1)$

Split the leftover $(1 - s)$ evenly among all nodes (evaporation-rain)

**Scaled PageRank update rule:**

$$ r_v(k) = s \times \frac{r_u(k - 1)}{d_u^{out}} + \frac{1 - s}{N_v}, \text{ for all } v \in V $$

- Can view as basic update $r(k) = \tilde{P}^T r(k - 1)$ with

$$ \tilde{P} := sP + (1 - s)\frac{11^\top}{N_v} $$

- Scaling factor $s$ typically chosen between 0.8 and 0.9
- Power iteration converges to the dominant eigenvector of $\tilde{P}$
A primer on Markov chains

Centrality measures

Case study: Stability of centrality measures in weighted graphs

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
Markov chains

- Consider discrete-time index \( n = 0, 1, 2, \ldots \)
- Time-dependent random state \( X_n \) takes values on a countable set
  - In general denote states as \( i = 0, 1, 2, \ldots \), i.e., here the state space is \( \mathbb{N} \)
  - If \( X_n = i \) we say “the process is in state \( i \) at time \( n \)”
- Random process is \( X_\mathbb{N} \), its history up to \( n \) is \( X_n = [X_n, X_{n-1}, \ldots, X_0]^T \)
- Def: process \( X_\mathbb{N} \) is a Markov chain (MC) if for all \( n \geq 1, i, j, x \in \mathbb{N}^n \)
  \[
P(X_{n+1} = j \mid X_n = i, X_{n-1} = x) = P(X_{n+1} = j \mid X_n = i) = P_{ij}
\]
- Future depends only on current state \( X_n \) (memoryless, Markov property)
  ⇒ Future conditionally independent of the past, given the present
Group the $P_{ij}$ in a transition probability “matrix” $\mathbf{P}$

$$
\mathbf{P} = \begin{pmatrix}
  P_{00} & P_{01} & P_{02} & \cdots & P_{0j} & \cdots \\
  P_{10} & P_{11} & P_{12} & \cdots & P_{1j} & \cdots \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
  P_{i0} & P_{i1} & P_{i2} & \cdots & P_{ij} & \cdots \\
  \vdots & \vdots & \vdots & \ddots & \vdots & \ddots
\end{pmatrix}
$$

$\Rightarrow$ Not really a matrix if number of states is infinite

Row-wise sums should be equal to one, i.e., $\sum_{j=0}^{\infty} P_{ij} = 1$ for all $i$
A graph representation or state transition diagram is also used.

Useful when number of states is infinite, skip arrows if $P_{ij} = 0$.

Again, sum of per-state outgoing arrow weights should be one.
Example: Bipolar mood

- I can be happy ($X_n = 0$) or sad ($X_n = 1$)
  - My mood tomorrow is only affected by my mood today

- Model as Markov chain with transition probabilities

\[
P = \begin{pmatrix}
0.8 & 0.2 \\
0.3 & 0.7
\end{pmatrix}
\]

- Inertia \(\Rightarrow\) happy or sad today, likely to stay happy or sad tomorrow
- But when sad, a little less likely so \((P_{00} > P_{11})\)
Example: Random (drunkard’s) walk

- Step to the right w.p. $p$, to the left w.p. $1 - p$
  $\Rightarrow$ Not that drunk to stay on the same place

- States are 0, $\pm 1$, $\pm 2$, $\ldots$ (state space is $\mathbb{Z}$), infinite number of states
- Transition probabilities are
  $$P_{i,i+1} = p, \quad P_{i,i-1} = 1 - p$$
- $P_{ij} = 0$ for all other transitions
Q: What can be said about multiple transitions?

Probabilities of $X_{m+n}$ given $X_m$ \iff $n$-step transition probabilities

$$P_{ij}^n = P(X_{m+n} = j \mid X_m = i)$$

\iff Define the matrix $P^{(n)}$ with elements $P_{ij}^n$

Theorem

*The matrix of $n$-step transition probabilities $P^{(n)}$ is given by the $n$-th power of the transition probability matrix $P$, i.e.,*

$$P^{(n)} = P^n$$

*Henceforth we write $P^n$*
Unconditional probabilities

- All probabilities so far are conditional, i.e., \( P^n_{ij} = P(X_n = j \mid X_0 = i) \)
  ⇒ May want unconditional probabilities \( p_j(n) = P(X_n = j) \)
- Requires specification of initial conditions \( p_i(0) = P(X_0 = i) \)
- Using law of total probability and definitions of \( P^n_{ij} \) and \( p_j(n) \)
  \[
p_j(n) = P(X_n = j) = \sum_{i=0}^{\infty} P(X_n = j \mid X_0 = i) P(X_0 = i) \\
  = \sum_{i=0}^{\infty} P^n_{ij} p_i(0)
  \]
- In matrix form (define vector \( \mathbf{p}(n) = [p_1(n), p_2(n), \ldots]^T \))
  \[
  \mathbf{p}(n) = (\mathbf{P}^n)^T \mathbf{p}(0)
  \]
Limiting distributions

- MCs have one-step memory. Eventually they forget initial state
- Q: What can we say about probabilities for large $n$?

$$\pi_j := \lim_{n \to \infty} P\left( X_n = j \mid X_0 = i \right) = \lim_{n \to \infty} P^n_{ij}$$

⇒ Assumed that limit is independent of initial state $X_0 = i$

- We’ve seen that this problem is related to the matrix power $P^n$

$$P = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}, \quad P^2 = \begin{pmatrix} 0.7 & 0.3 \\ 0.45 & 0.55 \end{pmatrix}, \quad P^7 = \begin{pmatrix} 0.6031 & 0.3969 \\ 0.5953 & 0.4047 \end{pmatrix}, \quad P^{30} = \begin{pmatrix} 0.6000 & 0.4000 \\ 0.6000 & 0.4000 \end{pmatrix}$$

- Matrix product converges ⇒ probs. independent of time (large $n$)
- All rows are equal ⇒ probs. independent of initial condition
Limit distribution of ergodic Markov chains

**Theorem**

*For an ergodic (i.e. irreducible, aperiodic, and positive recurrent) MC, \( \lim_{n \to \infty} P^n_{ij} \) exists and is independent of the initial state \( i \), i.e.,*

\[
\pi_j = \lim_{n \to \infty} P^n_{ij}
\]

*Furthermore, steady-state probabilities \( \pi_j \geq 0 \) are the unique nonnegative solution of the system of linear equations*

\[
\pi_j = \sum_{i=0}^{\infty} \pi_i P_{ij}, \quad \sum_{j=0}^{\infty} \pi_j = 1
\]

*Limit probs. independent of initial condition exist for ergodic MC*  
⇒ Simple algebraic equations can be solved to find \( \pi_j \)
Markov chains meet eigenvalue problems

- Define vector steady-state distribution \( \pi := [\pi_0, \pi_1, \ldots, \pi_J]^T \)

- Limit distribution is unique solution of
  \[
  \pi = P^T \pi, \quad \pi^T 1 = 1
  \]

- Eigenvector \( \pi \) associated with eigenvalue 1 of \( P^T \)
  - Eigenvectors are defined up to a scaling factor
  - Normalize to sum 1

- All other eigenvalues of \( P^T \) have modulus smaller than 1

- Computing \( \pi \) as eigenvector is computationally efficient
Ergodicity

- **Def:** Fraction of time $T_i^{(n)}$ spent in $i$-th state by time $n$ is

$$T_i^{(n)} := \frac{1}{n} \sum_{m=1}^{n} \mathbb{I} \{X_m = i\}$$

- Compute expected value of $T_i^{(n)}$

$$\mathbb{E} \left[ T_i^{(n)} \right] = \frac{1}{n} \sum_{m=1}^{n} \mathbb{E} \left[ \mathbb{I} \{X_m = i\} \right] = \frac{1}{n} \sum_{m=1}^{n} P (X_m = i)$$

- As $n \to \infty$, probabilities $P (X_m = i) \to \pi_i$ (ergodic MC). Then

$$\lim_{n \to \infty} \mathbb{E} \left[ T_i^{(n)} \right] = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} P (X_m = i) = \pi_i$$

- For ergodic MCs same is true without expected value $\Rightarrow$ Ergodicity

$$\lim_{n \to \infty} T_i^{(n)} = \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{I} \{X_m = i\} = \pi_i, \quad \text{a.s.}$$
Consider an ergodic Markov chain with transition probability matrix

\[ P := \begin{pmatrix}
0 & 0.3 & 0.7 \\
0.1 & 0.5 & 0.4 \\
0.1 & 0.2 & 0.7
\end{pmatrix} \]

Visits to states, \( nT_i^{(n)} \)

Ergodic averages, \( T_i^{(n)} \)

Ergodic averages slowly converge to \( \pi = [0.09, 0.29, 0.61]^T \)
PageRank: Random walk formulation

Centrality measures

Case study: Stability of centrality measures in weighted graphs

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
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 of** $i$ **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 $n(i)$
  $\Rightarrow$ All neighbors chosen with equal probability
- If reach a dead end because node $i$ has no neighbors
  $\Rightarrow$ Chose next visit at random equiprobably among all nodes
- Redefine graph $G = (V, E)$ adding edges from dead ends to all nodes
  $\Rightarrow$ 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

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

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
Ranking algorithm

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; \quad r = \text{zeros}(J,1); \quad \% \text{ Initialize time and ranks}
A_0 = \text{random(}'unid',J); \quad \% \text{ Draw first visit uniformly at random}
\textbf{while} \ m < n \ \textbf{do}
\begin{align*}
\quad \quad \quad \text{jump} &= \text{random(}'unid',d_{A_{m-1}}^{out}); \quad \% \text{ Neighbor uniformly at random} \\
\quad \quad \quad A_m &= N(A_{m-1}, \text{jump}); \quad \% \text{ Jump to selected neighbor} \\
\quad \quad \quad r(A_m) &= r(A_m) + 1; \quad \% \text{ Update ranking for } A_m \\
\quad \quad \quad m &= m + 1;
\end{align*}
\textbf{end}
\quad \quad \quad r = r/n; \quad \% \text{ 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”
Ranked class graph
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 $r^{(i)}_n$ be the index of the $i$-th highest ranked node at time $n$
- First element wrongly ranked at time $n$

$$
\xi_n := \arg \min_i \{ r^{(i)} \neq r^{(i)}_n \}
$$
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
Centrality measures

Case study: Stability of centrality measures in weighted graphs

Centrality, link analysis and web search

A primer on Markov chains

PageRank as a random walk

PageRank algorithm leveraging Markov chain structure
Recall definition of rank \( r_i := \lim_{n \to \infty} \frac{1}{n} \sum_{m=1}^{n} \mathbb{1}\{A_m = i\} \)

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

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

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

- 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 $K$

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

Visits are now spread over time and space

$\Rightarrow$ Converges “$K$ times faster”

$\Rightarrow$ But haven’t changed computational cost
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

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

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

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

⇒ Probability propagation $\approx$ Unleashing infinitely many agents
Distance to rank

- Initialize with uniform probability distribution \( p(0) = (1/J)1 \)

\( \Rightarrow \) 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

- Centrality measure
- Closeness centrality
- Dijkstra’s algorithm
- Betweenness centrality
- Information controller
- Eigenvector centrality
- Perron’s Theorem
- Power method
- Information retrieval
- Link analysis
- Repeated improvement

- Hubs and authorities
- HITS algorithm
- PageRank
- Spider traps
- Scaled PageRank updates
- Ergodic Markov chain
- Limiting probabilities
- Random walk on a graph
- Long-run fraction of state visits
- Probability propagation
- Distributed algorithm