13: Betweenness Centrality
Machine Learning and Real-world Data (MLRD)

Paula Buttery (based on slides by Simone Teufel)
Last session: some simple network statistics

- You measured the **degree** of each node and the **diameter** of the network.

- Next two sessions:
  - Today: finding **gatekeeper** nodes via **betweenness centrality**.
  - Next session: using betweenness centrality of edges to split graph into **cliques**.

- Reading for social networks (all sessions):
  - Easley and Kleinberg for background: Chapters 1, 2, 3 and first part of Chapter 20.
  - Brandes algorithm: two papers by Brandes (links in practical notes).
Centralities help us talk about interesting nodes

- **Degree**: the number of edges connected to a node (can be split into incoming and outgoing) (discovers direct influencers)
- **Closeness**: average of the distances from the node (discovers indirect influencers)
- **Betweenness**: relative number of shortest paths that rely on the node (discovers gatekeepers)
Gatekeepers nodes are associated with local bridges

- Last time we saw the concept of **local bridge**: an edge which increased the shortest paths if cut.

![Diagram](image)

- A–B is a local bridge here.

*Figure 3-4 from Easley and Kleinberg (2010)*
Nodes with high betweenness are on relatively many shortest paths

- The betweenness centrality of a node $V$ is defined in terms of the proportion of shortest paths that go through $V$ for each pair of nodes.

Here: the red nodes have high betweenness centrality. because we only care about shortest paths.

https://www.linkedin.com/pulse/wtf-do-youactually-know-who-influencers-walter-pike
Betweenness: example

Claudio Rocchini: https://commons.wikimedia.org/wiki/File:Graph_betweenness.svg

- Betweenness: red is minimum; dark blue is maximum.
Betweenness centrality, formally

- Directed graph $G = \langle V, E \rangle$
- $\sigma(s, t)$: number of shortest paths between nodes $s$ and $t$
- $\sigma(s, t|v)$: number of shortest paths between nodes $s$ and $t$ that pass through $v$.
- $C_B(v)$, the betweenness centrality of $v$:

$$C_B(v) = \sum_{s,t \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)}$$
Calculating betweenness verbosely

\[ V = \{A, B, C, D, E\} \]
\[ E = \{(A,B), (B,C), (B,D), (C,E), (D,E)\} \]

\[ C_B(v) = \sum_{s,t\in V} \frac{\sigma(s, t|v)}{\sigma(s, t)} \]

<table>
<thead>
<tr>
<th>path</th>
<th>route</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A,B)</td>
<td>A-B</td>
<td></td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A,C)</td>
<td>A-B-C</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A,D)</td>
<td>A-B-D</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(A,E)</td>
<td>A-B-C-E</td>
<td>0.5</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>A-B-D-E</td>
<td>0.5</td>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(B,C)</td>
<td>B-C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>(B,D)</td>
<td>B-D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>(B,E)</td>
<td>B-C-E</td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td></td>
<td>B-D-E</td>
<td></td>
<td></td>
<td></td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>(C,E)</td>
<td>C-E</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>(D,E)</td>
<td>D-E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>
Calculating Betweenness with Brandes

1) Find number of shortest paths:

- $\sigma(s, t)$ can be calculated recursively:

$$\sigma(s, t) = \sum_{u \in \text{Pred}(t)} \sigma(s, u)$$

- $\text{Pred}(t) = \{u: (u, t) \in E, d(s, t) = d(s, u) + 1\}$
predecessors of $t$ on shortest path from $s$
- $d(s, u)$: Distance between nodes $s$ and $u$

- Using a Breadth First search with each node as source $s$
once, gives total complexity of $O(V(V + E))$. 
2) Find dependency on specific nodes for specific shortest paths:

- There are a cubic number of pairwise dependencies \( \delta(s, t|v) \) where:
  \[
  \delta(s, t|v) = \frac{\sigma(s, t|v)}{\sigma(s, t)}
  \]

- Brandes algorithm intuition: the dependencies can be aggregated without calculating them all explicitly.
- Can calculate dependency of \( s \) on \( v \) based on dependencies one step further away.
Calculating Betweenness with Brandes

2) Find dependency on specific nodes for specific shortest paths:

- Define one-sided dependencies (how dependant are shortest paths from $s$ on $v$):

$$\delta(s|v) = \sum_{t \in V} \delta(s, t|v)$$

- Then Brandes (2001) shows:

$$\delta(s|v) = \sum_{(v, w) \in E} \frac{\sigma(s, v)}{\sigma(s, w)} \cdot (1 + \delta(s|w)) \quad w: d(s, w) = d(s, v) + 1$$

And: $C_B(v) = \sum_{s \in V} \delta(s|v)$
Calculating Betweenness with Brandes

Algorithm:

- For all vertices \( s \in V \):
- Calculate \( \delta(s \mid v) \) for all \( v \in V \) in two phases:
  1. Breadth-first search, calculating distances and shortest path counts from \( s \), (push all vertices onto stack as they’re visited).
  2. Visit all vertices in reverse order (pop off stack), aggregating dependencies according to equation.
Calculating Betweenness with Brandes: example

\[ \sigma(s, t) = \sum_{u \in \text{Pred}(t)} \sigma(s, u) \]

where

\[ \text{Pred}(t) = \{ u : (u, t) \in E, d(s, t) = d(s, u) + 1 \} \]

and \( d(s, t) \) is distance between \( s \) and \( t \)
Calculating Betweenness with Brandes: example

\[ \delta(s|v) = \sum_{(v,w) \in E} \frac{\sigma(s,v)}{\sigma(s,w)} \cdot (1+\delta(s|w)) \quad \text{w: } d(s,w) = d(s,v) + 1 \]
Calculating Betweenness with Brandes: example

\[
\delta(s|v) = \sum_{(v,w) \in E} \frac{\sigma(s,v)}{\sigma(s,w)} \cdot (1 + \delta(s|w)) \\
\text{where: } d(s,w) = d(s,v) + 1
\]
Calculating Betweenness with Brandes: example

\[ C_B(v) = \sum_{s \in V} \delta(s|v) \]

\[ C_B(A) = 0 \]
\[ C_B(B) = 3 \]
\[ C_B(C) = 1 \]
\[ C_B(D) = 1 \]
\[ C_B(E) = 0 \]
Calculating Betweenness with Brandes

- The algorithm is for directed graphs.
- But undirected graphs are easy: the algorithm works in exactly the same way, except that each pair is considered twice, once in each direction.
- Therefore: halve the scores at the end for undirected graphs.
- Brandes has lots of other variants, including edge betweenness centrality, which we’ll use in the next session.
Today

- **Task 11**: Implement the Brandes algorithm for efficiently determining the betweenness of each node.
Literature

- Detailed notes on the Brandes algorithm on course page / Moodle.
- Easley and Kleinberg (2010, page 79-82). But this is an informal description.