Centrality & Clustering

CMSC 498J: Social Media Computing

Department of Computer Science
University of Maryland
Spring 2015

Hadi Amiri
hadi@umd.edu
Lecture Topics

• Centrality
  ▫ Degree Centrality
  ▫ Closeness Centrality
  ▫ Betweenness Centrality

• Clustering
  ▫ Edge Betweenness
  ▫ Computing Edge Betweenness
Centrality

• What characterizes an important node in a network?
  ▫ Most influential people in social nets,
  ▫ Key infrastructure nodes in the Internet
  ▫ Main spreaders of disease
  ▫ Etc.

• Structural view:
  ▫ Importance of a node is related to its position in the network.
Centrality Measures

• Different centrality measures capture different structural characteristics of nodes!
• There is often a high correlation between these measures!
• Sometimes the most central node might depend on which measure is used!

C : Centrality
  ▫ C (i) : Centrality for node i
  ▫ C(A) : Centrality for a group of nodes A ∈ N
Degree Centrality

• A node is central if it has ties to many other nodes
  ▫ Look at the node degree
Degree Centrality - Cnt.

- A node is central if it has ties to many other nodes
  - Look at the node degree

\[ C(n_1) = \sum_{j=1}^{n} A_{1j} = \sum_{i=1}^{n} A_{i1} = 3 \]

\[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Adjacency Matrix (A)
Degree Centrality - Cnt.

• Standardized Degree Centrality
  ▫ Divide by the maximum possible degree centrality value!

• N-1

\[
C(n_1) = \left( \sum_{j=1}^{n} A_{1j} \right)/(n-1) = \left( \sum_{i=1}^{n} A_{i1} \right)/(n-1) = 3/4
\]

\[
\begin{align*}
n_1 & & n_2 & & n_3 & & n_4 & & n_5 \\
n_1 & & 0 & & 1 & & 1 & & 1 & & 0 & & 3/4 \\
n_2 & & 1 & & 0 & & 0 & & 0 & & 0 & & 1/4 \\
n_3 & & 1 & & 0 & & 0 & & 0 & & 0 & & 1/4 \\
n_4 & & 1 & & 0 & & 0 & & 0 & & 1 & & 1/2 \\
n_5 & & 0 & & 0 & & 0 & & 1 & & 0 & & 1/4 \\
\end{align*}
\]
Closeness Centrality

• A node is central if it is “close” to other nodes
  ▫ Look at distance btw nodes
  ▫ Closeness: 1 / Distance between nodes

\[ C(n_i) = \frac{1}{\left( \sum_{j=1}^{n} D_{ij} \right) \left( \sum_{i=1}^{n} D_{ii} \right)} = 1/4 \]

Distance Matrix (D)

\[
\begin{array}{ccccc}
  & n_1 & n_2 & n_3 & n_4 & n_5 \\
 n_1 & 0 & 1 & 1 & 1 & 1 \\
 n_2 & 1 & 0 & 2 & 2 & 2 \\
 n_3 & 1 & 2 & 0 & 2 & 2 \\
 n_4 & 1 & 2 & 2 & 0 & 2 \\
 n_5 & 1 & 2 & 2 & 2 & 0 \\
\end{array}
\]
Closeness Centrality - Cnt.

• Standardized Closeness Centrality
  ▫ Divide by the maximum possible closeness centrality value!
    • \( 1/(N-1) \)

\[
C(n_1) = (N - 1) / \left( \sum_{j=1}^{n} D_{1j} \right) = (N - 1) / \left( \sum_{i=1}^{n} D_{ii} \right) = \frac{4}{4}
\]

Distance Matrix (D)

<table>
<thead>
<tr>
<th></th>
<th>n1</th>
<th>n2</th>
<th>n3</th>
<th>n4</th>
<th>n5</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>n2</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>n3</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>n4</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>n5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

Values: 4/4, 4/7
Closeness Centrality- Cnt.

• How to compute Closeness Centrality in networks with disconnected components?

Only consider the giant component or do graph sampling?

Only consider nodes that are reachable in paths of length 1, 2, ... This is called k-Step Reach!
Betweenness Centrality

• A node is central if other nodes have to go through it to get to each other
  ▫ Look at shortest paths between nodes

1. 
2. For each pair of nodes, compute the shortest paths between them.
3. For each pair of nodes, determine the fraction of shortest paths that pass through a target node.
4. Sum the fractions over all pairs of nodes
5. }
Betweenness Centrality- Cnt.

\[ s_{jk} \]  
Number of shortest paths btw nodes \( n_j \) and \( n_k \) 

\[ s_{jk}(n_i) \]  
Number of shortest paths btw nodes \( n_j \) and \( n_k \) that include node \( n_i \) 

\[ \frac{s_{jk}(n_i)}{s_{jk}} \]  
Proportion of shortest paths btw nodes \( n_j \) and \( n_k \) that include node \( n_i \) 

\[ C(n_i) = \sum_{j,k!=i} \frac{s_{jk}(n_i)}{s_{jk}} \]  
Proportion of shortest paths btw all nodes that include node \( n_i \) 

<table>
<thead>
<tr>
<th>Shortest paths</th>
<th>( n_1-n_2-n_4 )</th>
<th>( n_1-n_3-n_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_{14} )</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>( s_{14}(n_2) )</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>( s_{14}(n_2)/s_{14} )</td>
<td>1/2</td>
<td></td>
</tr>
<tr>
<td>( C(n_2) )</td>
<td>1/2</td>
<td></td>
</tr>
</tbody>
</table>

Shortest paths btw \( n_1-n_3 \) and \( n_3-n_4 \) don’t include \( n_2 \) ! Their corresponding proportions are 0.
Betweenness Centrality - Cnt.

\[ s_{jk} \] Number of shortest paths btw nodes \( n_j \) and \( n_k \)

\[ s_{jk}(n_i) \] Number of shortest paths btw nodes \( n_j \) and \( n_k \) that include node \( n_i \)

\[ \frac{s_{jk}(n_i)}{s_{jk}} \] Proportion of shortest paths btw nodes \( n_j \) and \( n_k \) that include node \( n_i \)

\[ C(n_i) = \sum_{j} \sum_{k \neq j} \frac{s_{jk}(n_i)}{s_{jk}} \] Proportion of shortest paths btw all nodes that include node \( n_i \)

---

<table>
<thead>
<tr>
<th>Pair</th>
<th>Shortest path</th>
<th>Betweenness</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1 n2</td>
<td>n1-n2</td>
<td>n1</td>
</tr>
<tr>
<td>n1 n3</td>
<td>n1-n3</td>
<td>n2</td>
</tr>
<tr>
<td>n1 n4</td>
<td>n1-n4</td>
<td>n3</td>
</tr>
<tr>
<td>n1 n5</td>
<td>n1-n4-n5</td>
<td>n3</td>
</tr>
<tr>
<td>n2 n3</td>
<td>n2-n1-n3</td>
<td>n4</td>
</tr>
<tr>
<td>n2 n4</td>
<td>n2-n1-n4</td>
<td>n4</td>
</tr>
<tr>
<td>n2 n5</td>
<td>n2-n1-n4-n5</td>
<td>n5</td>
</tr>
<tr>
<td>n3 n4</td>
<td>n3-n1-n4</td>
<td>n5</td>
</tr>
<tr>
<td>n3 n5</td>
<td>n3-n1-n4-n5</td>
<td></td>
</tr>
</tbody>
</table>
Betweenness Centrality- Cnt.

- **Standardized Betweenness Centrality**
  - Divide by the maximum possible betweenness centrality value!
    - \((N-1)(N-2)/2\) : the number of other pairs of nodes (exclude the node itself)

\[
C(n_i) = \frac{\sum_{j,k \neq i} \frac{s_{jk}(n_i)}{s_{jk}}}{(N - 1)(N - 2)/2}
\]
Betweenness Centrality - Cnt.

- **Standardized Betweenness Centrality**
  - Divide by the maximum possible betweenness centrality value!
  - \((N-1)(N-2)/2\) : the number of other pairs of nodes (exclude the node itself)

```
<table>
<thead>
<tr>
<th>Node</th>
<th>Betweenness</th>
<th>Stnd. Betweenness</th>
</tr>
</thead>
<tbody>
<tr>
<td>n1</td>
<td>5</td>
<td>5/6 = 0.83</td>
</tr>
<tr>
<td>n2</td>
<td>0</td>
<td>0/6 = 0.00</td>
</tr>
<tr>
<td>n3</td>
<td>0</td>
<td>0/6 = 0.00</td>
</tr>
<tr>
<td>n4</td>
<td>3</td>
<td>3/6 = 0.50</td>
</tr>
<tr>
<td>n5</td>
<td>0</td>
<td>0/6 = 0.00</td>
</tr>
</tbody>
</table>
```
Clustering

• We aim to develop techniques to identify densely connected regions
  ▫ breaking a network into a set of densely connected nodes
  ▫ with sparse connections between groups
• Graph Partitioning
Clustering- Cnt.

- Members of the same group are heavily connected, while
- Members of different groups are less connected!
Clustering- Approaches

• Divisive methods
  ▫ breaking first at the 7-8 edge, and then the nodes into nodes 7 and 8
Clustering- Approaches- Cnt.

- Agglomerative methods
  - merge the 4 triangles and then pairs of triangles (via nodes 7 and 8)
Divisive Approach

• Bridges connect tightly-knit groups in networks!
  ▫ To find clusters, remove bridges and local bridges!
  ▫ **Issue 1**: when there are several bridges, which one to remove?
Divisive Approach - Cnt.

- Bridges connect tightly-knit groups in networks!
  - To find cluster, remove bridges and local bridges!
  - **Issue 2**: What if there is no bridge?

A network can display tightly-knit regions even when there are no bridges or local bridges along which to separate it.
Divisive Approach - Cnt.

- Bridges form part of the shortest path between pairs of nodes in different parts of the network! We can utilize this characteristic of bridges!
  - Find the edges that carry most of “traffic” in a network and successively remove edges of high traffic!
Edge Betweenness

• **Edge Betweenness:**
  • Let’s assume 1 unit of “flow” will pass over all shortest path btw any pair of nodes A and B.
  • Betweenness of an edge is the total amount of flow it carries!
  • If there are $k$ shortest path btw A and B, then $1/k$ units of flow will go along each shortest path!

• **Girvan-Newman Algorithm:**
  ▫ Repeat until no edges are left:
    • Calculate betweenness of edges
    • Remove edges with highest betweenness
Communities are the resultant connected components!
Computing Edge Betweenness

• A clever way to compute betweennesses efficiently
  ▫ Use breadth-first Search

1. For each node A{
2. Run BFS on A
3. Count the number of shortest paths from A to any other node
4. Determine the amount of traffic from A to other nodes
5. }
6. Compute betweenness for each edge by summing all the traffic passing over the edge
Computing Edge Betweenness

• A clever way to compute betweennesses efficiently
  ▫ Use breadth-first Search
  ▫ Consider the graph from the perspective of one node at a time!

BFS divides a graph into layers starting at a given node, with all the nodes in layer $d$ having distance $d$ from A.
Computing Edge Betweenness - Cnt.

- A clever way to compute betweennesses efficiently
  - Count the number of shortest paths from A to all other nodes of the network

Number of shortest paths to each node is the sum of the number of shortest paths to all nodes directly above it!
Computing Edge Betweenness- Cnt.

• A clever way to compute betweennesses efficiently
  ▫ Determine the amount of traffic from A to others
    • If there are $k$ shortest path btw A and B, then $1/k$ units of flow will go through each shortest path.
    • Working up from the lowest layers and computing the amount of flow that pass through each edge!
Computing Edge Betweenness - Cnt.

- A clever way to compute betweennesses efficiently
  - Determine the amount of traffic from A to others

3 paths from I and 3 from J → the 1 unit of flow will be equally divided

Amount of flow from A to E: 1 unit destined for E + 1 unit passing through to H = 2
1 path btw A & E → all flow go through it

Amount of flow from A to H: 1 unit destined for H + 1 unit passing through to J = 2
2 paths btw A & H → flow equally divided

Amount of flow from A to J: 1 unit destined for J + ½ unit passing through to K = 3/2
3 paths btw A & J → 2 shortest paths from H and one shortest path from G → the amount of flow from H should be twice than the amount of flow from G!

Amount of flow from A to K: 1 unit
Computing Edge Betweenness- Cnt.

• A clever way to compute betweennesses efficiently
  ▫ Use breadth-first Search

1. For each node A{
2. Run BFS on A
3. Count the number of shortest paths from A to any other node
4. Determine the amount of traffic from A to other nodes
5. }
6. Compute betweenness for each edge by summing all the traffic passing over the edge and divide by 2

Note that we count the flow between each pair of nodes A and B twice (once when running BFS from A and once when running BFS from B)! So, we need to divide resultant values by 2!
Questions?
Reading

• Ch.03 Strong and Weak Ties [NCM]