Community Detection

**Problem:** Can we identify groups of densely connected nodes?

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Communities: Football Conferences

Nodes: Football Teams, Edges: Matches, Communities: Conferences

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Communities: Academic Citations

Nodes: Journals, Edges: Citations, Communities: Academic Disciplines

Source: Citation networks and Maps of science [Börner et al., 2012]

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Communities: Protein-Protein Interactions

Nodes: Proteins, Edges: Physical interactions, Communities: Functional Modules

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
We will work with **undirected** (unweighted) networks

(Adapted from: Mining of Massive Datasets, [http://www.mmds.org](http://www.mmds.org))
The notion of closeness centrality is meaningfully defined with respect to undirected and connected networks. The average shortest path distance, starting from node $i$, is denoted by $AvDist(i)$ and defined in terms of the pair wise shortest distance $Dist(i, j)$ between nodes $i$ and $j$ as follows:

$$AvDist(i) = \frac{1}{n-1} \sum_{j=1}^{n} Dist(i, j)$$ (19.8)

The closeness centrality is simply the inverse of the average distance of other nodes to node $i$.

$$CC(i) = \frac{1}{AvDist(i)}$$ (19.9)

Because the value of $AvDist(i)$ is at least 1, this measure ranges between 0 and 1. In the case of Fig. 19.1a, node 3 has the highest closeness centrality because it has the lowest average distance to other nodes.

A measure known as proximity prestige can be used to measure prestige in directed networks. To compute the proximity prestige of node $i$, the shortest path distance from all other nodes is computed. Unlike undirected networks, a confounding factor in the computation is that directed paths may not exist from other nodes to node $i$. For example, no path exists to node 7 in Fig. 19.1b. Therefore, the first step is to determine the set of nodes Influence($i$) that can reach node $i$ with a directed path. For example, in the case of the Twitter network, Influence($i$) corresponds to a recursively defined follower set of node $i$.

An example of an influence set of node 1 is illustrated in Fig. 19.1b. The value of $AvDist(i)$ can now be computed only with respect to the influence set Influence($i$).

$$AvDist(i) = \frac{1}{|Influence(i)|} \sum_{j \in Influence(i)} Dist(j, i)$$ (19.10)

Note that distances are computed from node $j$ to $i$, and vice versa, because we are computing a prestige measure, rather than a gregariousness measure.

Both the size of the influence set and average distance to the influence set play a role in defining the proximity prestige. While it is tempting to use the inverse of the average distance, as in the previous case, this would not be fair. Nodes that have less influence should be penalized. For example, in Fig. 19.1b, node 6 has the lowest possible distance value of 1 from node 7, which is also the only node it influences. While its low average distance to its influence set suggests high prestige, its small influence set suggests that it is not very influential.

- **Betweenness**: Number of shortest paths
- **Closeness**: Average distance to other nodes
- **Degree**: Number of connections to other nodes
Betweenness

Edge Strength (call volume)  Edge Betweenness

• **Betweenness**: Number of shortest paths passing through a node or edge

(Adapted from: Mining of Massive Datasets, [http://www.mmds.org](http://www.mmds.org))
Edge Betweenness

**Figure 10.7: Betweenness scores for the graph of Fig. 10.1.**

Clearly, edge (B,D) has the highest betweenness, so it should be removed first. That leaves us with exactly the communities we observed make the most sense, namely: {A, B, C} and {D, E, F, G}. However, we can continue to remove edges. Next to leave are (A, B) and (B, C) with a score of 5, followed by (D, E) and (D, G) with a score of 4.5. Then, (D, F), whose score is 4, would leave the graph. We see in Fig. 10.8 the graph that remains.

The “communities” of Fig. 10.8 look strange. One implication is that A and C are more closely knit to each other than to B. That is, in some sense B is a “traitor” to the community {A, B, C} because he has a friend D outside that community. Likewise, D can be seen as a “traitor” to the group {D, E, F, G}, which is why in Fig. 10.8, only E, F, and G remain connected.

• Count number of shortest paths passing through each edge (can be done with weighted edges)
• If there are multiple paths of equal length, then split counts
Girvan-Newman Algorithm
(hierarchical divisive clustering according to betweenness)

Repeat until k clusters found
1. Calculate betweenness
2. Remove edge(s) with highest betweenness

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Girvan-Newman Algorithm
(hierarchical divisive clustering according to betweenness)

Hierarchical network

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Girvan-Newman: Physics Citations

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Girvan-Newman

Two problems

1. How can we compute the betweenness for all edges?
2. How can we choose the number of components $k$?
Calculating Betweenness

*How can we count all shortest paths?*

- Loop over nodes in graph
  - Perform breadth-first search to find shortest paths to other nodes
  - Increment counts for edges traversed by shortest paths
- Divide final betweenness by 2 (since all paths counted twice)
Counting Shortest Paths

Count number of shortest paths from (E) to each node

Accumulate credit upwards, dividing across shortest paths

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Counting Paths: Larger Example

Original Graph

Breadth-first Ordering from A

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Counting Paths: Larger Example

Step 1. Count number of shortest paths from to each node

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Step 2. Propagate credit upwards, splitting according to number of paths to parents

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Counting Paths: Larger Example

Step 2. Propagate credit upwards, splitting according to number of paths to parents

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Counting Paths: Larger Example

Step 2. Propagate credit upwards, splitting according to number of paths to parents

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Counting Paths: Larger Example

Step 2. Propagate credit upwards, splitting according to number of paths to parents

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Counting Paths: Larger Example

Step 2. Propagate credit upwards, splitting according to number of paths to parents

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Determining the Number of Communities

Hierarchical decomposition

Choosing a cut-off

Analogous problem to deciding on number of clusters in hierarchical clustering

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Modularity

Idea: Compare fraction of edges within module to fraction that would be observed for random connections

\[
Q = \frac{1}{2m} \sum_{uv} \left[ A_{uv} - \frac{k_v k_w}{2m} \right] \delta(c_u, c_v)
\]

- \( m \): Number of edges in graph
- \( A_{uv} \): Adjacency matrix (1 if edge exists 0 otherwise)
- \( k_u \): Degree of node \( u \)
- \( c_u \): Cluster assignment for node \( u \)

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Use modularity to optimize connectivity within modules

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Spectral Clustering
Graph Partitioning

- What makes a good partition?
  - Maximize the within-group connections
  - Minimize the between-group connections

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Graph Cuts

Degree

\[ d_i = \sum_j A_{ij} \]

Volume

\[ \text{vol}(A) = \sum d_i \]

Cut

\[ \text{cut}(A, B) = \sum_{i \in A, j \in B} A_{ij} \]

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Minimal Cuts

\[ \arg \min_{A,B} \text{cut}(A,B) \]

*Problem:* minimal cut is not necessarily a good splitting criterion

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Normalized Cuts

\[ \text{Degree} \quad d_i = \sum_j A_{ij} \]

\[ \text{Volume} \quad \text{vol}(A) = \sum d_i \]

\[ \text{Cut} \quad \text{cut}(A, B) = \sum_{i \in A, j \in B} A_{ij} \]

\[ \text{Normalized Cut} \quad \text{ncut}(A, B) = \frac{\text{cut}(A, B)}{\text{vol}(A)} + \frac{\text{cut}(A, B)}{\text{vol}(B)} \]
Find Optimal Cut [Fiedler’73]

- Back to finding the optimal cut
- Express partition (A,B) as a vector
  \[ y_i = \begin{cases} 
  +1 & \text{if } i \in A \\
  -1 & \text{if } i \in B 
  \end{cases} \]
- We can minimize the cut of the partition by finding a non-trivial vector \( x \) that minimizes:

\[
 y^* = \arg\min_{y \in \{-1,1\}^n} \sum_{(i,j) \in E} (y_i - y_j)^2
\]

Can’t solve exactly. Let’s relax \( y \) and allow it to take any real value.

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Matrix Representations

• Adjacency matrix \((A)\):
  • \(n \times n\) matrix
  • \(A=[a_{ij}], a_{ij}=1\) if edge between node \(i\) and \(j\)

![Adjacency Matrix Diagram]

• Important properties:
  • Symmetric matrix
  • Eigenvectors are real and orthogonal

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Matrix Representations

• Degree matrix (D):
  • $n \times n$ diagonal matrix
  • $D=[d_{ii}]$, $d_{ii} =$ degree of node $i$

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Matrix Representations

- **Laplacian matrix (L):**
  - $n \times n$ symmetric matrix

  ![Graph with nodes and edges]

  ![Matrix representation]

- **What is trivial eigenpair?**
  - $x = (1, \ldots, 1)$ then $L \cdot x = 0$ and so $\lambda = \lambda_1 = 0$

- **Important properties:**
  - **Eigenvalues** are non-negative real numbers
  - **Eigenvectors** are real and orthogonal

(Adapted from: Mining of Massive Datasets, [http://www.mmds.org](http://www.mmds.org))
Second Eigenvalue

- Fact: For symmetric matrix $M$:

$$\lambda_2 = \min_x \frac{x^T M x}{x^T x}$$

- What is the meaning of $\min x^T L x$ on $G$?

- $x^T L x = \sum_{i,j=1}^{n} L_{ij} x_i x_j = \sum_{i,j=1}^{n} (D_{ij} - A_{ij}) x_i x_j$

- $= \sum_i D_{ii} x_i^2 - \sum_{(i,j) \in E} 2x_i x_j$

- $= \sum_{(i,j) \in E} (x_i^2 + x_j^2 - 2x_i x_j) = \sum_{(i,j) \in E} (x_i - x_j)^2$

Node $i$ has degree $d_i$. So, value $x_i^2$ needs to be summed up $d_i$ times. But each edge $(i,j)$ has two endpoints so we need $x_i^2 + x_j^2$

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Second Eigenvector of Laplacian

- What else do we know about \( x \)?
  - \( x \) is unit vector: \( \sum_i x_i^2 = 1 \)
  - \( x \) is orthogonal to 1st eigenvector \((1, \ldots, 1)\) thus:
    \( \sum_i x_i \cdot 1 = \sum_i x_i = 0 \)

- Remember:

\[
\lambda_2 = \min \sum_{(i, j) \in E} \frac{(x_i - x_j)^2}{\sum_i x_i^2}
\]

We want to assign values \( x_i \) to nodes \( i \) such that few edges cross 0.
(we want \( x_i \) and \( x_j \) to subtract each other)

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Rayleigh Theorem

$$\min_{y \in \mathbb{R}^n} f(y) = \sum_{(i,j) \in E} (y_i - y_j)^2 = y^T Ly$$

- $\lambda_2 = \min_y f(y)$: The minimum value of $f(y)$ is given by the 2^{nd} smallest eigenvalue $\lambda_2$ of the Laplacian matrix $L$

- $x = \arg\min_y f(y)$: The optimal solution for $y$ is given by the corresponding eigenvector $x$, referred as the Fiedler vector

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Spectral Clustering Algorithms

- Three basic stages:
  - 1) Pre-processing
    - Construct a matrix representation of the graph
    - More generally, construct similarity matrix
  - 2) Decomposition
    - Compute eigenvalues and eigenvectors of the matrix
    - Map each point to a lower-dimensional representation based on one or more eigenvectors
  - 3) Grouping
    - Assign points to two or more clusters, based on the new representation

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Spectral Partitioning Algorithm

1) Pre-processing:
   - Build Laplacian matrix $L$ of the graph

2) Decomposition:
   - Find eigenvalues $\lambda$ and eigenvectors $x$ of the matrix $L$
   - Map vertices to corresponding components of $\lambda_2$

How do we now find the clusters?

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Spectral Partitioning

3) **Grouping:**
- Sort components of reduced 1-dimensional vector
- Identify clusters by splitting the sorted vector in two

**How to choose a splitting point?**
- Naïve approaches:
  - Split at 0 or median value
- More expensive approaches:
  - Attempt to minimize normalized cut in 1-dimension (sweep over ordering of nodes induced by the eigenvector)

### Table:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
</tr>
<tr>
<td>4</td>
<td>-0.3</td>
</tr>
<tr>
<td>5</td>
<td>-0.3</td>
</tr>
<tr>
<td>6</td>
<td>-0.6</td>
</tr>
</tbody>
</table>

**Split at 0:**
- **Cluster A:** Positive points
- **Cluster B:** Negative points

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.3</td>
</tr>
<tr>
<td>2</td>
<td>0.6</td>
</tr>
<tr>
<td>3</td>
<td>0.3</td>
</tr>
</tbody>
</table>

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>-0.3</td>
</tr>
<tr>
<td>5</td>
<td>-0.3</td>
</tr>
<tr>
<td>6</td>
<td>-0.6</td>
</tr>
</tbody>
</table>

A | B
Example: Spectral Partitioning

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
Example: Spectral Partitioning

(Adapted from: Mining of Massive Datasets, http://www.mmds.org)
k-Way Spectral Clustering

• How do we partition a graph into \( k \) clusters?

• Two basic approaches:
  • Recursive bi-partitioning [Hagen et al., ’92]
    • Recursively apply bi-partitioning algorithm in a hierarchical divisive manner
    • Disadvantages: Inefficient, unstable
  • Cluster multiple eigenvectors [Shi-Malik, ’00]
    • Build a reduced space from multiple eigenvectors
    • Commonly used in recent papers
Spectral Clustering as General-purpose Method

Define “edge weight” $W$ using some similarity metric (e.g. a kernel function)

source: Ng, Jordan and Weiss, NIPS 2001