Community Structure in Networks

CS224W: Analysis of Networks
Jure Leskovec, Stanford University
http://cs224w.stanford.edu
Nodes with different structural roles (connector node, bridge node, etc.)

Nodes belonging to the same cluster/community
Plan for Today:

- **Structural role** discovery in networks
- **Community detection** via Modularity optimization
Structural Roles in Networks
Roles are “functions” of nodes in a network:
- Roles of species in ecosystems
- Roles of individuals in companies

Roles are measured by structural behaviors:
- Centers of stars
- Members of cliques
- Peripheral nodes, etc.
Example of Roles

- centers of stars
- members of cliques
- peripheral nodes

Network Science
Co-authorship network
[Newman 2006]
Roles versus Groups in Networks

- **Role:** A collection of nodes which have similar positions in a network:
  - Roles are based on the similarity of ties among subsets of nodes
  - Different from **community** (or cohesive subgroup)
    - Group is formed based on adjacency, proximity or reachability
    - This is typically adopted in current data mining

Nodes with the same role need not be in direct, or even indirect interaction with each other
Roles and Communities

- **Roles:**
  - A group of nodes with similar structural properties

- **Communities:**
  - A group of nodes that are well-connected to each other

- Roles and communities are complementary

Consider the social network of a CS Dept:

- **Roles:** Faculty, Staff, Students
- **Communities:** AI Lab, Info Lab, Theory Lab
- **Structural equivalence**: Nodes $u$ and $v$ are structurally equivalent if they have the same relationships to all other nodes [Lorrain & White 1971]

- Structurally equivalent nodes are likely to be similar in other ways – *i.e.*, friendships in social networks
Nodes $u$ and $v$ are **structurally equivalent**:
- For all the other nodes $k$, node $u$ has tie to $k$ iff node $v$ has tie to $k$

**Example:**

![Graph diagram]

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>-</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

*E.g.*, nodes 3 and 4 are structurally equivalent.
Discovering Structural Roles in Networks
### Why Are Roles Important?

<table>
<thead>
<tr>
<th>Task</th>
<th>Example Application</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Role query</strong></td>
<td>Identify individuals with similar behavior to a known target</td>
</tr>
<tr>
<td><strong>Role outliers</strong></td>
<td>Identify individuals with unusual behavior</td>
</tr>
<tr>
<td><strong>Role dynamics</strong></td>
<td>Identify unusual changes in behavior</td>
</tr>
<tr>
<td><strong>Identity resolution</strong></td>
<td>Identify/de-anonymize, individuals in a new network</td>
</tr>
<tr>
<td><strong>Role transfer</strong></td>
<td>Use knowledge of one network to make predictions in another</td>
</tr>
<tr>
<td><strong>Network comparison</strong></td>
<td>Compute similarity of networks, determine compatibility for knowledge transfer</td>
</tr>
</tbody>
</table>
**RolX:** Automatic discovery of nodes’ structural roles in networks

[Henderson, et al. 2011b]

- Unsupervised learning approach
- No prior knowledge required
- Assigns a mixed-membership of roles to each node
- Scales linearly in #(edges)
RolX: Approach Overview

Input

Node × Node Adjacency Matrix

Recursive Feature Extraction

Node × Feature Matrix

Role Extraction

Node × Role Matrix

Role × Feature Matrix

Output

Example: degree, mean weight, # of edges in ego-network, mean clustering coefficient of neighbors, etc.
Recursive Feature Extraction

- **Recursive feature extraction** [Henderson, et al. 2011a] turns network connectivity into structural features

- **Neighborhood features**: What is a node’s connectivity pattern?
- **Recursive features**: To what kinds of nodes is a node connected?
Idea: Aggregate features of a node and use them to generate new recursive features

Base set of a node’s neighborhood features:

Local features: All measures of the node degree:
- If network is directed, include in- and out-degree, total degree
- If network is weighted, include weighted feature versions

Egonetwork features: Computed on the node’s egonet:
- Egonet includes the node, its neighbors, and any edges in the induced subgraph on these nodes
- #(within-egonet edges), #(edges entering/leaving egonet)
Recursive Feature Extraction

- Start with the base set of node features
- Use the set of current node features to generate additional features:
  - Two types of aggregate functions: means and sums
    - E.g., mean value of “unweighted degree” feature among all neighbors of a node
    - Compute means and sums over all current features, including other recursive features
  - Repeat
- The number of possible recursive features grows exponentially with each recursive iteration:
  - Reduce the number of features using a pruning technique:
    - Look for pairs of features that are highly correlated
    - Eliminate one of the features whenever two features are correlated above a user-defined threshold
Role Extraction

Input

Recursively extract features

Output

Features

1) Can compare nodes based on their structural similarity
2) Can cluster nodes to identify different structural roles

e.g., RolX uses a clustering technique called non-negative matrix factorization
Task: Cluster nodes based on their structural similarity

Two networks:
- Network science co-authorship network:
  - Nodes: Network scientists; Edges: The number of co-authored papers
- Political books co-purchasing network:
  - Nodes: Political books on Amazon; Edges: Frequent co-purchasing of books by the same buyers

Setup: For each network:
- Use RolX to assign each node a distribution over the set of discovered, structural roles
- Determine similarity between nodes by comparing their role distributions
Making sense of roles:

- **Blue circle**: Tightly knit, nodes that participate in tightly-coupled groups
- **Red diamond**: Bridge nodes, that connect groups of nodes
- **Gray rectangle**: Main-stream, most of nodes, neither a clique, nor a chain
- **Green triangle**: Pathy, nodes that belong to elongated clusters
Community Structure in Networks
Roles and Communities: Example

Roles

Communities

Henderson, *et al.*, KDD 2012

Networks & Communities

- We often think of networks “looking” like this:

- What led to such a conceptual picture?
How does information flow through the network?
- What structurally distinct roles do nodes play?
- What roles do different links ("short" vs. "long") play?

How do people find out about new jobs?
- Mark Granovetter, part of his PhD in 1960s
- People find the information through personal contacts

But: Contacts were often acquaintances rather than close friends
- This is surprising: One would expect your friends to help you out more than casual acquaintances

Why is it that acquaintances are most helpful?
Two perspectives on friendships:

- **Structural**: Friendships span different parts of the network
- **Interpersonal**: Friendship between two people is either strong or weak

**Structural role: Triadic Closure**

If two people in a network have a friend in common, then there is an increased likelihood they will become friends themselves.

Which edge is more likely, a-b or a-c?
Granovetter makes a connection between social and structural role of an edge

First point: Structure
- Structurally embedded edges are also socially strong
- Long-range edges spanning different parts of the network are socially weak

Second point: Information
- Long-range edges allow you to gather information from different parts of the network and get a job
- Structurally embedded edges are heavily redundant in terms of information access
Triadic Closure

- **Triadic closure == High clustering coefficient**

**Reasons for triadic closure:**

- If $B$ and $C$ have a friend $A$ in common, then:
  - $B$ is *more likely to meet* $C$
    - (since they both spend time with $A$)
  - $B$ and $C$ *trust* each other
    - (since they have a friend in common)
  - $A$ has *incentive* to bring $B$ and $C$ together
    - (since it is hard for $A$ to maintain two disjoint relationships)

- **Empirical study by Bearman and Moody:**
  - Teenage girls with low clustering coefficient are more likely to contemplate suicide
For many years Granovetter’s theory was not tested
But, today we have large who-talks-to-whom graphs:
- Email, Messenger, Cell phones, Facebook

Onnela et al. 2007:
- Cell-phone network of 20% of country’s population
- Edge strength: # phone calls
**Neighborhood Overlap**

- **Edge overlap:**

\[
O_{ij} = \frac{|N(i) \cap N(j)|}{|N(i) \cup N(j)|}
\]

- \(N(i)\) ... a set of neighbors of node \(i\)

- **Overlap = 0** when an edge is a local bridge
Cell-phone network

Observation:
- Highly used links have high overlap!

Legend:
- True: The data
- Permuted strengths: Keep the network structure but randomly reassign edge strengths
Real edge strengths in mobile call graph

- Strong ties are more embedded (have higher overlap)
- Same network, same set of edge strengths but now strengths are randomly shuffled
Removing links by strength (#calls)

- Low to high
- High to low

Low disconnects the network sooner

Fraction of removed links vs. Size of largest component

Conceptual picture of network structure
Link Removal by Overlap

- Removing links based on **overlap**
  - Low to high
  - High to low

- Conceptual picture of network structure

- Low disconnects the network sooner
Granovetter’s theory leads to the following conceptual picture of networks.
Network Communities
Granovetter’s theory suggest that networks are composed of tightly connected sets of nodes.

- **Network communities:**
  - Sets of nodes with lots of internal connections and few external ones (to the rest of the network).
How to automatically find such densely connected groups of nodes?

Ideally such automatically detected clusters would then correspond to real groups.

For example:
Zachary’s Karate club network:

- Observe social ties and rivalries in a university karate club
- During his observation, conflicts led the group to split
- Split could be explained by a minimum cut in the network
Find micro-markets by partitioning the “query-to-advertiser” graph in web search:

Nodes: advertisers and queries/keywords; Edges: Advertiser advertising on a keyword.
Can we identify node groups? (communities, modules, clusters)

Nodes: Teams
Edges: Games played
NCAA Football Network

NCAA conferences

- Mid American
- Big East
- Atlantic Coast
- SEC
- Conference USA
- Big 12
- Western Athletic
- Pacific 10
- Mountain West
- Big 10
- Sun Belt
- Independents

Nodes: Teams
Edges: Games played
Can we identify social communities?

Nodes: Users
Edges: Friendships

Facebook Ego-network
Facebook Ego-network

High school

Company

Stanford (Squash)

Stanford (Basketball)

Social communities

Nodes: Users

Edges: Friendships
Protein-Protein Interactions

Can we identify functional modules?

Nodes: Proteins
Edges: Interactions
Protein-Protein Interactions

Functional modules

Nodes: Proteins
Edges: Interactions
Network Communities

- **Communities**: sets of tightly connected nodes

- **Define**: Modularity $Q$
  - A measure of how well a network is partitioned into communities
  - Given a partitioning of the network into groups $s \in S$:
    $$ Q \propto \sum_{s \in S} \left[ (\text{# edges within group } s) - (\text{expected # edges within group } s) \right] $$

**Need a null model!**
Given real $G$ on $n$ nodes and $m$ edges, construct rewired network $G'$

- Same degree distribution but random connections
- Consider $G'$ as a multigraph
- The expected number of edges between nodes $i$ and $j$ of degrees $k_i$ and $k_j$ equals to: $k_i \cdot \frac{k_j}{2m} = \frac{k_i k_j}{2m}$
- The expected number of edges in (multigraph) $G'$:
  $$\frac{1}{2} \sum_{i \in N} \sum_{j \in N} \frac{k_i k_j}{2m} = \frac{1}{2} \cdot \frac{1}{2m} \sum_{i \in N} k_i \left( \sum_{j \in N} k_j \right) =$$
  $$= \frac{1}{4m} 2m \cdot 2m = m$$

Note: $\sum_{u \in N} k_u = 2m$
Modularity

- Modularity of partitioning $S$ of graph $G$:
  - $Q \propto \sum_{s \in S} \left( \# \text{ edges within group } s \right) - \left( \text{expected } \# \text{ edges within group } s \right)$
  - $Q(G, S) = \frac{1}{2m} \sum_{s \in S} \sum_{i \in s} \sum_{j \in s} \left( A_{ij} - \frac{k_i k_j}{2m} \right)$

  Normalizing const.: $-1 < Q < 1$

- Modularity values take range $[-1, 1]$:
  - It is positive if the number of edges within groups exceeds the expected number
  - $Q$ greater than $0.3-0.7$ means significant community structure
Equivalently, modularity can be written as:

\[
Q(G, \mathcal{S}) = \frac{1}{2m} \sum_{s \in \mathcal{S}} \sum_{i \in s} \sum_{j \in s} \left( A_{ij} - \frac{k_i k_j}{2m} \right)
\]

Idea: We can identify communities by maximizing modularity.
Louvain Modularity
Louvain Algorithm

- **Greedy algorithm** for community detection
  - $O(n \log n)$ run time
- Supports weighted graphs
- Provides hierarchical partitions
- Widely utilized to **study large networks** because:
  - Fast
  - Rapid convergence properties
  - High modularity output (i.e., “better communities”)

“Fast unfolding of communities in large networks” Blondel et al. (2008)
Louvain Algorithm: At High Level

- Louvain algorithm **greedily maximizes** modularity
- **Each pass is made of 2 phases:**
  - **Phase 1:** Modularity is **optimized** by allowing only local changes of communities
  - **Phase 2:** The identified communities are **aggregated** in order to build a new network of communities
- **Goto Phase 1**

The passes are repeated **iteratively** until no increase of modularity is possible!
Louvain: 1st phase (partitioning)

- Put each node in a graph into a distinct community (one node per community)

- For each node $i$, the algorithm performs two calculations:
  - Compute the modularity gain ($\Delta Q$) when putting node $i$ into the community of some neighbor $j$
  - Move $i$ to a community of node $j$ that yields the largest gain $\Delta Q$

- The loop runs until no movement yields a gain
What is $\Delta Q$ if we move node $i$ to community $C$?

$$\Delta Q(i \rightarrow C) = \left[ \frac{\sum_{in} + k_{i,in}}{2m} - \left( \frac{\sum_{tot} + k_i}{2m} \right)^2 \right] - \left[ \frac{\sum_{in}}{2m} - \left( \frac{\sum_{tot}}{2m} \right)^2 - \left( \frac{k_i}{2m} \right)^2 \right]$$

- where:
  - $\sum_{in}$... sum of link weights between nodes in $C$
  - $\sum_{tot}$... sum of all link weights of nodes in $C$
  - $k_{i,in}$... sum of link weights between node $i$ and $C$
  - $k_i$... sum of all link weights (i.e., degree) of node $i$

- Also need to derive $\Delta Q(D \rightarrow i)$ of taking node $i$ out of community $D$.
- And then: $\Delta Q = \Delta Q(i \rightarrow C) + \Delta Q(D \rightarrow i)$
The partitions obtained in the first phase are contracted into **super-nodes**, and the network is created accordingly.

- Super-nodes are connected if there is at least one edge between nodes of the corresponding partitions.
- The weight of the edge between the two super-nodes is the sum of the weights from all edges between their corresponding partitions.

The loop runs until the community configuration does not change anymore.
Louvain Algorithm
Belgian Mobile phone network

- 2M nodes
- **Red nodes:** French speakers
- **Green nodes:** Dutch speakers

Figure 3. Graphical representation of the network of communities extracted from a Belgian mobile phone network. About 2M customers are represented on this network. The size of a node is proportional to the number of individuals corresponding to the community and its colour on a red-green scale represents the main language spoken (red for French and green for Dutch). Only the communities composed of more than 100 customers have been plotted. Notice the intermediate community of mixed colours between the two main language clusters. A zoom at higher resolution reveals that it is made of several sub-communities with less apparent language separation.

These communities may possibly be merged in the later passes, after blocks of nodes have been aggregated. However, our algorithm provides a decomposition of the network into communities for different levels of organization. For instance, when applied on the clique network proposed in [23], the cliques are indeed merged in the final partition but they are distinct after the first pass (see Figure 2). This result suggests that the intermediate solutions found by our algorithm may also be meaningful and that the uncovered hierarchical structure may allow the end-user to observe its structure with the desired resolution.