Network 3 – Community Detection

Giorgio Ricchiuti
www.grarchive.net
giorgio.ricchiuti@unifi.it

CompEc

Quick Recap

Last lecture:

- Centrality measures: degree, betweenness, closeness, eigenvector
- Katz centrality and PageRank
- Directed and weighted networks
- Python: computing centrality on the Karate Club network

Today:

- What is a community?
- Modularity Q: definition and intuition
- The Girvan–Newman algorithm
- The Louvain algorithm
- Comparing partitions: NMI
- Python: community detection on the Karate Club

What is a Community?

- Real-world networks are **not random**: nodes tend to cluster into groups.
- A **community** (or module) is a group of nodes that are **more densely connected to each other** than to the rest of the network.

Why does it matter?

- Social networks: detect groups, echo chambers
- Financial networks: identify clusters of correlated institutions
- Biology: functional modules in protein networks
- Economics: trading blocs in international trade networks

The challenge:

- There is **no unique definition** of community
- The number of communities is unknown a priori
- The problem is NP-hard in general

$\Rightarrow$ We need **heuristic algorithms** and a **quality function**

Modularity

The most widely used quality function is **modularity** Q (Newman & Girvan, 2004):

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j) \tag{1}$$

where:

- A_{ij}: adjacency matrix entry
- k_i: degree of node i; m: total number of edges
- $\frac{k_i k_j}{2m}$: expected number of edges between i and j in a **random graph** with the same degree sequence
- $\delta(c_i, c_j) = 1$ if i and j belong to the same community, 0 otherwise

Interpretation

Q measures how much the within-community edge density **exceeds** what we would expect by chance. $Q \in [-1, 1]$; values $Q > 0.3$ indicate meaningful community structure.

Modularity – Intuition

High modularity ($Q \approx 1$):

- Edges are concentrated *within* communities
- Few edges cross community boundaries
- Strong group structure

Low / negative modularity ($Q \approx 0$):

- Edge distribution is close to random
- No meaningful community structure

Limitations of modularity:

- **Resolution limit**: small communities inside large networks may be missed (Fortunato & Barthélemy, 2007)
- Many partitions can have similar Q values
- Maximising Q is NP-hard $\Rightarrow$ heuristics needed

The Girvan–Newman Algorithm

Key insight (Girvan & Newman, 2002): edges *between* communities tend to have **high edge betweenness** — many shortest paths pass through them.

Algorithm (divisive / top-down):

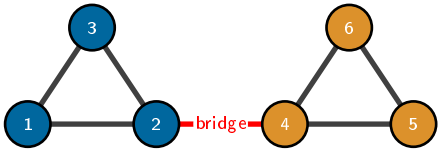
1. Compute **edge betweenness centrality** for all edges
2. Remove the edge with the **highest betweenness**
3. Recompute betweenness for all remaining edges
4. Repeat until no edges remain

This produces a **dendrogram**: a hierarchical tree of partitions.

Choosing the partition

We select the level of the dendrogram that **maximises modularity** Q.

Girvan–Newman – Example



- Nodes 1–3 and 4–6 form two dense groups
- Edge $(2, 4)$ is the **bridge**: all paths between the two groups pass through it
- It has the highest **edge betweenness** $\Rightarrow$ removed first
- After removal: two separate components = two communities

Complexity

$O(m^2 n)$ — slow on large networks.
Suitable for networks with $n \lesssim 10{,}000$.

The Louvain Algorithm

Louvain (Blondel et al., 2008): a fast, greedy modularity-maximisation algorithm.

Complexity: $O(n \log n)$ — scales to **millions of nodes**.

Phase 1 – Local optimisation:

1. Each node starts in its own community
2. For each node i, try moving it to a neighbour's community
3. Keep the move if it **increases** Q
4. Repeat until no single move improves Q

Phase 2 – Aggregation:

1. Build a new (smaller) network where nodes are the communities found in Phase 1
2. Apply Phase 1 to this aggregated network

Repeat both phases until Q no longer improves $\Rightarrow$ **hierarchical** community structure.

Louvain – Modularity Gain

When moving node i from community C to community D, the modularity gain is:

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

where:

- Σ_{in}: sum of edge weights *inside* community D
- Σ_{tot}: sum of degrees of all nodes in D
- $k_{i,in}$: sum of weights of edges from i to nodes in D
- k_i: degree of node i

Key property

ΔQ can be computed **locally** and **efficiently** — no need to recompute the full modularity at each step.

How Good is a Partition? NMI

When ground truth is known, we can measure how close a detected partition is to the real one.

Normalised Mutual Information (NMI):

$$\mathrm{NMI}(X, Y) = \frac{2\, I(X; Y)}{H(X) + H(Y)} \tag{3}$$

where $I(X; Y)$ is the mutual information and $H(\cdot)$ is entropy.

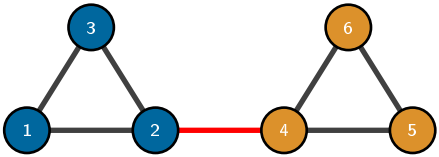
- $\mathrm{NMI} = 1$: perfect match with ground truth
- $\mathrm{NMI} = 0$: completely independent partitions
- Robust to the number of communities

Other metrics

- **ARI** (Adjusted Rand Index): accounts for chance
- **Accuracy**: fraction of correctly classified nodes (requires label matching)

A Worked Example – Computing Modularity by Hand

Consider a network with **6 nodes** and **7 edges**, split into two candidate communities $C_1 = \{1, 2, 3\}$ and $C_2 = \{4, 5, 6\}$:



Network parameters:

- $n = 6$ nodes, $m = 7$ edges
- Degrees:
  $k_1 = 2,\ k_2 = 3,\ k_3 = 2,\ k_4 = 3,\ k_5 = 2,\ k_6 = 2$
- Red edge: the only bridge between C_1 and C_2

Recall the modularity formula:

$$Q = \frac{1}{2m} \sum_{ij} \left[A_{ij} - \frac{k_i k_j}{2m} \right] \delta(c_i, c_j)$$

We only sum over pairs *within* the same community ($\delta = 1$).

A Worked Example – The Calculation

$m = 7$, so $2m = 14$. We compute $A_{ij} - \frac{k_i k_j}{14}$ for all within-community pairs:

Community $C_1 = \{1, 2, 3\}$:

i	j	A_{ij}	$\frac{k_i k_j}{14}$
1	2	1	$\frac{2 \cdot 3}{14} = 0.429$
1	3	1	$\frac{2 \cdot 2}{14} = 0.286$
2	3	1	$\frac{3 \cdot 2}{14} = 0.429$

Contribution C_1:

$$(1-0.429) + (1-0.286) + (1-0.429) = 1.856$$

Community $C_2 = \{4, 5, 6\}$:

i	j	A_{ij}	$\frac{k_i k_j}{14}$
4	5	1	$\frac{3 \cdot 2}{14} = 0.429$
4	6	1	$\frac{3 \cdot 2}{14} = 0.429$
5	6	1	$\frac{2 \cdot 2}{14} = 0.286$

Contribution C_2:

$$(1-0.429) + (1-0.429) + (1-0.286) = 1.856$$

Result

$$Q = \frac{1}{14}(1.856 + 1.856) = \frac{3.712}{14} \approx 0.265$$

$Q > 0$ confirms meaningful community structure. Removing the red bridge and re-running would increase Q further.

The Karate Club: Community Detection

Recall:

- 34 members, 78 friendships
- Conflict between instructor (**node 0**) and president (**node 33**)
- Club split into **two factions** ⇒ ground truth known

What we expect:

- Algorithms should recover (approximately) the two historical factions
- Nodes 0 and 33 should be in *different* communities
- NMI close to 1 ⇒ good recovery

Results (typical)

Algorithm	# Comm.	Q	NMI
Ground truth	2	—	1.000
Girvan–Newman	4	0.37	∼0.70
Louvain	4	0.42	∼0.72

Both algorithms recover the main split correctly. Louvain finds a slightly higher modularity partition with more sub-communities.

Algorithm Comparison

	Approach	Complexity	Scalability	Deterministic
Girvan–Newman	Divisive	$O(m^2 n)$	Low	Yes
Louvain	Greedy	$O(n \log n)$	High	No

Girvan–Newman:

+ Interpretable (edge betweenness logic)
+ Deterministic result
− Slow: not suitable for $n > 10{,}000$
− Must re-run to explore all levels

Louvain:

+ Very fast; scales to millions of nodes
+ Often finds higher Q
− Non-deterministic (random initialisation)
− Subject to resolution limit

Python: What We Will Do

1. Load the **Karate Club** network with `NetworkX`
2. Visualise the **ground-truth** partition (historical factions)
3. Run **Girvan–Newman**: plot modularity vs. number of communities
4. Run **Louvain** (package: `python-louvain`)
5. Visualise and compare the three partitions side by side
6. Compute **NMI** vs. ground truth for each algorithm

Key functions

- `nx.karate_club_graph()`
- `nx.community.girvan_newman()`
- `nx.community.modularity()`
- `community_louvain.best_partition()`
- `normalized_mutual_info_score()`
- `nx.draw_networkx()`

Python: Setup and Data

Packages needed

`networkx`, `matplotlib`, `scikit-learn`, `python-louvain`

Install: `pip install python-louvain`

Loading the network (Spyder)

```
import networkx as nx
import matplotlib.pyplot as plt
import community as community_louvain
from sklearn.metrics import normalized_mutual_info_score

G = nx.karate_club_graph()

# Ground-truth labels
ground_truth = {n: d['club'] for n, d in G.nodes(data=True)}
```

Python: Girvan–Newman

```
from networkx.algorithms.community import girvan_newman

gn_gen = girvan_newman(G)

best_Q, best_partition = -1, None
for partition in gn_gen:
    partition = list(partition)
    Q = nx.community.modularity(G, partition)
    if Q > best_Q:
        best_Q = Q
        best_partition = partition
    if len(partition) > 10:
        break

print(f"Best: {len(best_partition)} communities, Q = {best_Q:.4f}")
```

Python: Louvain and Comparison

Louvain

```
partition_lv = community_louvain.best_partition(G, random_state=42)
# partition_lv = {node: community_id, ...}
```

NMI vs ground truth

```
gt_labels = [0 if ground_truth[n]=='Mr. Hi' else 1 for n in G.nodes()]

# Girvan-Newman labels
gn_labels = [next(i for i,c in enumerate(best_partition) if n in c)
             for n in G.nodes()]

# Louvain labels
lv_labels = [partition_lv[n] for n in G.nodes()]

print(f"NMI Girvan-Newman: {normalized_mutual_info_score(gt_labels, gn_labels):.4f}")
print(f"NMI Louvain:       {normalized_mutual_info_score(gt_labels, lv_labels):.4f}")
```

Summary

Key concepts:

- A **community** = dense subgroup of nodes
- **Modularity** Q: compares observed density to a random benchmark
- **Girvan–Newman**: removes high-betweenness edges iteratively
- **Louvain**: greedily maximises ΔQ locally, then aggregates
- **NMI**: measures agreement between detected and true partition

Next lecture:

- We apply these tools to our **own class network**
- Data collection: who knows whom? who has worked with whom?
- Do communities in the class reflect study groups, friendships, or random chance?

References

Girvan & Newman (2002), PNAS
Newman & Girvan (2004), PRE
Blondel et al. (2008), J. Stat. Mech.
Newman (2010), *Networks*, Ch. 11