Community Detection 1

Modularity and Community Structure in Networks

Mia Yuan

Mar 11, 2024

DS8104

The detection of community

Community structure means the appearance of density connected groups of vertices, with only sparser coonnections between groups.
Past work on methods for discovering groups in networks:

1. Graph partitioning: number and size are known; doesn't consider the existence of good divisions.
2. Community structure detection

Find the communities

- In this case, we want to split the vertices into non-overlapping groups; these communities may be of any size.
- Begin with 2 communities:
  1. Minimum cut approach: look for divisions of the vertices into two groups so as to **minimize the number of edges running between the groups**.
  2. Problem with this approach: there's no constraints on size or number so simply counting edges is not a good way to quantify the intuitive concept of community structure.

Community

Modularity Optimization

Applications

Modularity

Modularity: **the number of edges falling within groups** minus the expected number in an equivalent network with **edges placed at random**.

$$Q = \frac{1}{2m}\sum_{ij}(A_{ij} - P_{ij})\delta_{ij}$$

$$Q = \frac{1}{4m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})(s_i s_j + 1) = \frac{1}{4m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})s_i s_j$$

1. Hypothesis: Randomly wired networks are not expected to have a community structure.
2. The configuration model can generate random graphs maintaining a fixed degree sequence.

Explaination

$$Q = \frac{1}{4m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})(s_i s_j + 1) = \frac{1}{4m}\sum_{ij}(A_{ij} - \frac{k_i k_j}{2m})s_i s_j$$

1. 1 being left out: $\sum_{ij} A_{ij} = \sum_{ij} \frac{k_i k_j}{2m} = 2m$;
2. s_i indicates which group i belongs to; when i and j belong to the different group, then $s_i s_j + 1 = 0$; this term is relative to a specific partition;
3. $m = \sum_i k_i/2$ is the total number of edges in the network; $2m$ is the number of stubs
4. $P_{ij} = \sum_{n=1}^{k_i} \frac{k_j}{2m-1} = \frac{k_i k_j}{2m-1} \approx \frac{k_i k_j}{2m}$

Modularity Optimization

Write it in matrix form:

$$Q = \frac{1}{4m}\mathbf{s}^T\mathbf{B}\mathbf{s}$$

where B is a modularity matrix, defined as:

$$B_{ij} = A_{ij} - \frac{k_i k_j}{2m}$$

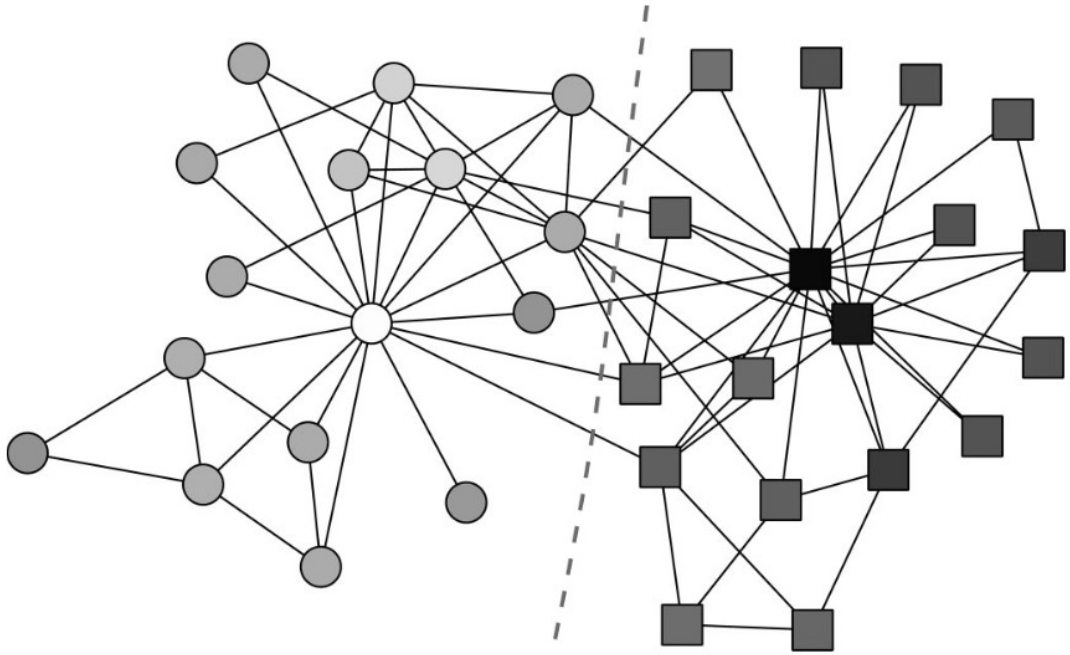
- $\mathbf{B}$ is a real symmetric matrix: $\mathbf{B} = \sum_{i=1}^{n} \beta_i \mathbf{u}_i \mathbf{u}_i^T$; $\mathbf{u}_i^T \mathbf{u}_j = 0, i \neq j$; $\mathbf{u}_i^T \mathbf{u}_i = 1$.
- $Q = \frac{1}{4m} \sum_i a_i \mathbf{u}_i^T (\sum_i \beta_i \mathbf{u}_i \mathbf{u}_i^T) \sum_i a_i \mathbf{u}_i = \frac{1}{4m} \sum_i (a_i \beta_i \mathbf{u}_i^T)(\sum_i a_i \mathbf{u}_i) = \frac{1}{4m} \sum_i (a_i^2 \beta_i) = \frac{1}{4m} \sum_i (\mathbf{u}_i^T \mathbf{s})^2 \beta_i$
- Graph Laplacian: the elements of each of its rows and columns sum to zero, so that it always has an eigenvector $(1, 1, 1, ..., 1)$ with eigenvalue zero.

Spectral Solution

- $Q = \frac{1}{4m}\sum_i (\mathbf{u}_i^T \mathbf{s})^2 \beta_i$:
  To optimize Q, $\mathbf{s}$'s direction should be as close as $\mathbf{u}_1$ to give largest weight to β_1. (Assume $\beta_1 \geq \beta_2 ... \geq \beta_n$)
- First find the largest eigenvalue of matrix B, then use the sign of the elements in its corresponding eigenvector to separate the vertices.
- As long as there is any positive eigenvalue this method will not put all vertices in the same group.
- It's possible there's no positive eigenvalue. In this case, the leading eigenvector is $(1, 1, 1, ..., 1)$ corresponding to all vertices in a single group together. This means no division of the network that results in positive modularity.

Besides the signs

- The magnitudes convey information too: Vertices corresponding to elements of large magnitude make large contributions to the modularity.
- Thus, the elements of the leading eigenvector measure how firmly each vertex belongs to its assigned community, those with large vector elements being strong central members of their communities.



More than two communities

- The following equation calculates the change of modularity when seperate 1 group into 2.

$$\Delta Q = \frac{1}{2m}[\frac{1}{2}\sum_{i,j\in g} B_{ij}(s_i s_j + 1) - \sum_{i,j\in g} B_{ij}]$$

$$\Delta Q = \frac{1}{4m}\mathbf{s}^T \mathbf{B}^{(g)}\mathbf{s},\ B_{ij}^{(g)} = B_{ij} - \delta_{ij}\sum_{k\in g} B_{ik}$$

- Apply spectral approach to this generalized modularity matrix to maximize ΔQ.
- The rows and columns of this matrix still sum to zero. So ΔQ is correctly zero if the group is undivided.
- Repeat the separation step on subgraphs after first divide it into 2 parts.

What if:

- If the leading eigenvalue is zero, which is the smallest value it can take, then the subgraph is indivisible. sufficient but not necessary.
- if there are only small positive eigenvalues and large negative ones, the terms for negative β_i may outweigh those for positive.
- In this case, calculate ΔQ for proposed split directly to ensure the contribution is positive.

Further Techniques

Fine-tuning:

1. Find the initial division through spectral method;
2. If move an vertice to the other group increase the modularity then do it;
3. Until there's no vertice can be moved; find the best intermediate state;
4. Start from the new stage until no further improvement in modularity.

Example 1

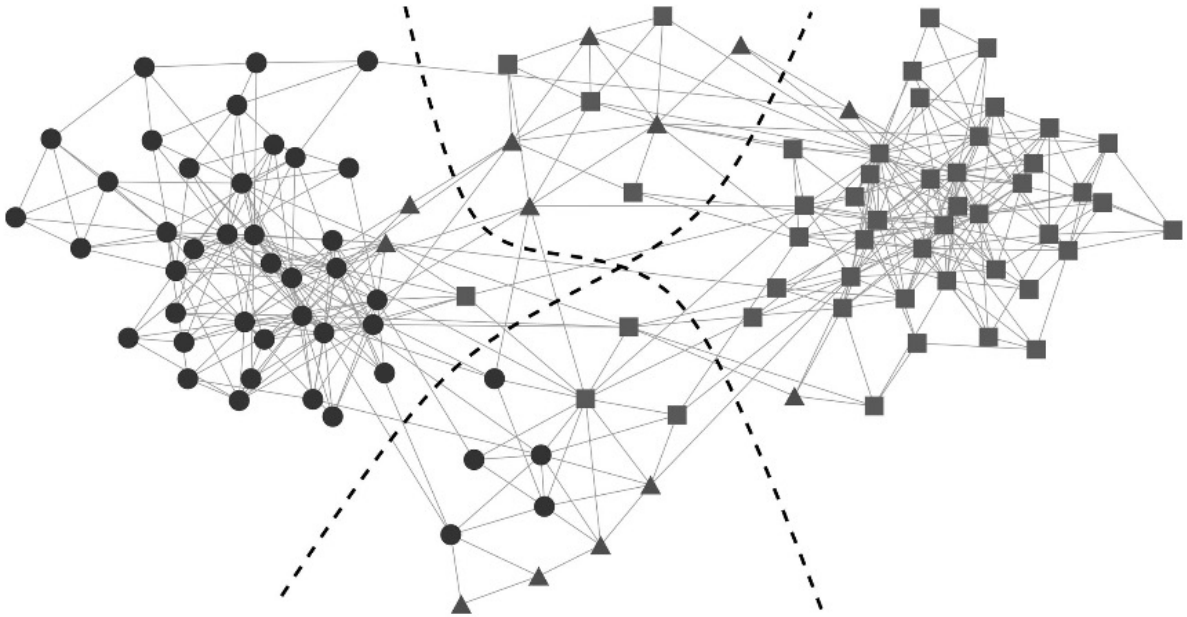
GN: betweenness-based; CNM: modularity optimization with greedy algortithm; DA: extremal optimization.

1. Outperforms GN and CNM in all settings;
2. Compared with DA: as the network goes larger, proposed method performs better

		Modularity Q			
Network	Size n	GN	CNM	DA	This article
Karate	34	0.401	0.381	0.419	0.419
Jazz musicians	198	0.405	0.439	0.445	0.442
Metabolic	453	0.403	0.402	0.434	0.435
E-mail	1,133	0.532	0.494	0.574	0.572
Key signing	10,680	0.816	0.733	0.846	0.855
Physicists	27,519	—	0.668	0.679	0.723

Example2

1. books about politics:



2. blogs about politics: divide 1225 blogs into two communities with modularity 0.426.

Computational complexity

1. Newman: $O(n^2 \log n)$
2. GN betweenness-based: $O(n^3)$
3. DA extremal optimization: $O(n^2 \log^2 n)$
4. CNM greedy algortithm:$O(n \log^2 n)$