Community Detection Algorithms

Leonid E. Zhukov

School of Applied Mathematics and Information Science
National Research University Higher School of Economics

14.03.2013
Set $S$ is a $k$-core if for $\forall v \in S$, $k(v) \geq k$ and is maximal subgraph.

The core number of a vertex is the highest order of a core that contains this vertex.
Finding k-cores

**Algorithm**: Core decomposition

**Input**: graph G(V,E)

**Output**: core[v] - core number for each vertex

- compute deg[v];
- sort V : deg[v_{i+1}] ≥ deg[v_i];

```
for each v ∈ V do
    core[v] = degree[v];
    for each u ∈ NN(v) do
        if deg[u] > deg[v] then
            deg[u] = deg[u] − 1;
            sort V
    end
end
```
**Community detection**

**Definition**

*Network community* is a group of vertices such that vertices inside the group are connected with many more edges than between groups.

- Consider only sparse graphs $m \ll n^2$
- Each community should be connected
- Divisive algorithms (vs agglomerative)
- Finding communities through graph partitioning
- Graph partition - division of graph vertices into groups such that each vertex belongs only to one group
- Graph cut is a set edges whose end points are in different partitions
- NP-hard, solved by approximation algorithms and heuristics
- Recursive 2-way partition, multiway partitioning
Edge betweenness

- number of shortest paths $\sigma_{st}(e)$ going through edge $e$

$$C_B(e) = \sum_{s \neq t} \frac{\sigma_{st}(e)}{\sigma_{st}}$$
Algorithm: Edge Betweenness

Input: graph $G(V,E)$
Output: Dendrogram

repeat
  For all $e \in E$ compute edge betweenness $C_B(e)$;
  remove edge $e_i$ with largest $C_B(e_i)$;
until edges left;

If bi-partition, then stop when graph splits in two components (check for connectedness)
Edge betweenness

Dendrogram
Modularity:

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

- \( c_i \)- class, \( \delta(c_i, c_j) \)- kronecker delta
- Single class, \( \delta(c_i, c_j) = 1 \)

\[ Q = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) = 0 \]
Modularity

- $e_{rs}$ - fraction of edges between vertices of types $r$ and $s$ (edges between groups):

$$e_{rs} = \frac{1}{2m} \sum_{i \in R, j \in S} A_{ij} = \frac{1}{2m} \sum_{ij} A_{ij} \delta(c_i, r) \delta(c_j, s)$$

- $a_r$ - fraction of edges attached to vertices of type $r$ (edges to all vertices):

$$a_r = \frac{1}{2m} \sum_{i \in R, j} A_{i,j} = \frac{1}{2m} \sum_{i} k_i \delta(c_i, r)$$

- Modularity

$$Q = \frac{1}{2m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) \delta(c_i, c_j) = \sum_r (e_{rr} - a_r^2)$$
Edge betweenness

Dendrogram and modularity score
Spectral Modularity Maximization

- Direct modularity maximization (bi-partitioning), [Newman, 2006]
- For two classes $C_1, C_2$ indicator variable $s = \pm 1$

$$\delta(c_i, c_j) = \frac{1}{2} (s_i s_j + 1) = \begin{cases} 
1 : & i, j \in C_1 \text{ or } i, j \in C_2 \\
0 : & i \in C_1, j \in C_2 \text{ or } i \in C_2, j \in C_1
\end{cases}$$

- Modularity

$$Q = \frac{1}{4m} \sum_{ij} \left( A_{ij} - \frac{k_i k_j}{2m} \right) (s_i s_j + 1) = \frac{1}{4m} \sum_{i,j} B_{ij} s_i s_j$$

where

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

- Quadratic form:
  \[ Q(s) = \frac{1}{4m} s^T B s \]

- Integer optimization - NP, relaxation \( s \rightarrow x, \ x \in R \)

- Keep norm \( \|x\|^2 = \sum_i x_i^2 = x^T x = n \)

- Quadratic optimization
  \[ Q'(x) = \frac{1}{4m} x^T B x - \lambda (x^T x - n) \]

- Eigenvector problem
  \[ B x_i = \lambda'_i x_i \]

- Approximate modularity
  \[ Q'(x_i) = \frac{n}{4m} \lambda_i \]

- Maximization - maximal \( \lambda \)
Can’t choose $s = x_k$, can select optimal $s$

Decompose in the basis: $s = \sum_j a_j x_j$, where $a_j = x_j^T s$

Modularity

$$Q(s) = \frac{1}{4m} s^T B s = \frac{1}{4m} \sum_i (x_i^T s)^2 \lambda_i$$

$\max Q(s)$ reached when $\lambda_1 = \lambda_{\text{max}}$ and $\max x_1^T s = \sum_j x_1 j s_j$

Choose $s \parallel x_1$, $s = \text{sign}(x_1)$
Algorithm: Spectral modularity maximization: two-way partition

**Input**: adjacency matrix $A$

**Output**: class indicator vector $s$

compute $k = \text{deg}(A)$;

compute $B = A - \frac{1}{2m} kk^T$;

solve for maximal eigenvector $Bx = \lambda x$;

set $s = \text{sign}(x_1)$

Recursive bisection
• An O(m) Algorithms for Cores Decomposition of Networks. V. Batagelj, M. Zaversnik, 2002