# Community

Yujia Zhu

## Community

Edges in a graph can be devided into two kinds. One is structurally embedded edges (tightly-connected edges), the other is long-range edges. Structurally embedded edges mean socially strong relationship, while long-range edges mean weak relationship. Granovetter's theory suggests that networks are composed of tightly connected sets of nodes. **Network Communities** are sets of nodes with lots of internal connections and few external ones to the rest of the network.

Since community is a set of tightly connected nodes, we can define Modularity to measure how well a network is partitioned into communties. Given a partition of the network into groups disjoint $s \in S$, Modularity $Q$ can be defined as:

$$Q \propto \sum_{s \in S} [(\#\text{edges within group } s) - (\#\text{expected edges within group } s)]$$

To compute expected number of edges within a group, we need a null model. Given a real graph $G$ with n nodes and m edges, reconstruct graph $G'$ with the same degree distribution by randomly connecting nodes. The expected number of edges between nodes $i$ and $j$ with degrees $k_i$ and $k_j$ equals $k_i \frac{k_j}{2m} = \frac{k_i k_j}{2m}$ . We consider directed graphs here, so there in all have $2m$ possible edges. So the expected number of edges between nodes in $G'$ is:

$$E = \frac{1}{2} \sum_{i \in N} \sum_{j \in N} \frac{k_i k_j}{2m} = \frac{1}{4m} \left( \sum_{i \in N} k_i \right) \left( \sum_{j \in N} k_i \right) = \frac{1}{4m} (2m)(2m) = m$$

So, the modularity of partitioning $S$ of graph $G$ is:

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

$A_{ij} = 1$ if node $i$ and $j$ are coneected else 0. For weighted edges, $A_{ij}$ can be the weight of the edge. So, modularity can apply to both weighted and unweighted networks. $\frac{1}{2m}$ is a normalizing constant which makes $Q(G, S) \in [-1, 1]$. Q value larger than 0.3-0.7 means significant community. We can identify communities by maximizing modularity.

## Louvain Algorithm

Louvain Algorithm is a greedy algorithm for community detection. Louvain algorithm greedily maximizes modularity in each pass. Each pass consists of 2 phases:

1. Modularity is optimized by allowing only local changes to node-communities memberships
2. The identified communities are aggregated into super-nodes to build a new network

Louvain algorithm considers graphs as weighted (as we aggregate nodes into super-nodes, a unweighted graph will turn into a weighted graph). In the first phase, we will put each node in a graph into a distinct community and then compute the modularity delta when putting node $i$ into the community of some neighbour $j$ for each node $i$. We will move $i$ to the community of node $j$ which gains largest $\Delta Q$.

Now we will compute $\Delta Q$ when we move node $i$ from community $D$ to $C$. This means $i$ is in community $D$ before moving and community $C$ afterwards. Define $\Delta Q(D \to i \to C)$ as the delta modualrity of moving node $i$ from community $D$ to $C$. It can be decomposed to two parts: $\Delta Q(D \to i)$ (moving node $i$ out from $D$) and $\Delta Q(i \to C)$ (moving node $i$ into $C$).

$$\Delta Q(D \to i \to C) = \Delta Q(D \to i) + \Delta Q(i \to C)$$

$\Delta Q(D \to i)$ and $\Delta Q(i \to C)$ can be derived similarly, so we will only dervie $\Delta Q(i \to C)$ as an example. To derive $\Delta Q(i \to C)$, we first derive modualrity between $C$, i.e., $Q(C)$. Define $\Sigma_{in} = \sum_{i,j \in C} A_{ij}, \Sigma_{tot} = \sum_{i \in C} k_i$, so $\Sigma_{in}$ is the sum of link weights between nodes in $C$ and $\Sigma_{out}$ is sum of all link weights of nodes in $C$. Then, we have:

$$\begin{aligned} Q(C) &= \frac{1}{2m} \sum_{i,j \in C} \left[ A_{ij} - \frac{k_i k_j}{2m} \right] = \frac{\sum_{i,j \in C} A_{ij}}{2m} - \frac{(\sum_{i \in C} k_i)(\sum_{j \in C} k_j)}{4m^2} \\ &= \frac{\Sigma_{in}}{2m} - \left( \frac{\Sigma_{tot}}{2m} \right)^2 \end{aligned}$$

Further define $k_{i,in} = \sum_{j \in C} A_{ij} + \sum_{j \in C} A_{ji}$ as the sum of link weights between node $i$ and community $C$ and $k_i$ as sum of all link weights (i.e. degree) of node $i$. So, we have:

$$Q_{\text{before}} = Q(C) + Q(\{i\}) = \left[ \frac{\Sigma_{in}}{2m} - \left( \frac{\Sigma_{tot}}{2m} \right)^2 \right] + \left[ 0 - \left( \frac{k_i}{2m} \right)^2 \right]$$

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

So,

$$\Delta Q(i \to C) = Q_{\text{after}} - Q_{\text{before}} = \frac{k_{i,in}}{2m} - \frac{k_i \Sigma_{tot}}{2m^2}$$

We can summarize Louvain algorithm as:

1. The first phase: iterate until no node moves to a new community. For each node $i$ in community $C$, compute the best $C'$
$$C' = arg \max_{C'} \Delta Q(C \to i \to C')$$
if $\Delta Q(C \to i \to C') > 0$, then change the partitioning: $C := C - \{i\}, C' := C' + \{i\}$.
2. The second phase: the communities obtained in the first phase are contracted into super-nodes. Note that super-nodes are connected if there is at least one edge between the nodes of the corresponding communities and the weight of the edge between the two supernodes is the sum of the weights from all edges between their corresponding communities.
3. Iterative step 1 and step 2 until there is only one node. Note that the outcome provides hierarchical communities.

## Summary

Community is very important for graph. Louvain algorithm is a fast and powerful algorithm for community detection.