Mohit Kumar
SR No. 04-01-03-10-51-21-1-19825
MTech Artificial Intelligence

Assignment 3: Community Detection

---

python packages used

1. numpy
2. scipy
3. matplotlib
4. networkx

Implementation details

- The spectralDecomp_OneIter(edgelist) function takes as input edgelist (nx2 numpy array, every row is an edge connecting nodes in the first and second columns) and returns the fielder vector, adjacency matrix of the graph and graph_partition (nx2 numpy array first column consists of all nodes in the network and the second column lists their community id) after running one iteration of the spectral decomposition algorithm which partitions the graph into two communities.

Algorithm 1 Spectral Graph Partitioning Algorithm

Input: $G = (V, E)$ and adjacency matrix A
Output: class indicator vector $\mathbf{s}$

compute $\mathbf{L} = \mathbf{D} - \mathbf{A}$
compute second smallest eigenvector
for min cut: solve $\mathbf{Lx} = \lambda\mathbf{x}$
for normalized cut: solve $\mathbf{Lx} = \lambda\mathbf{Dx}$
$\mathbf{s} = \mathbf{sign}(\mathbf{x_2})$

- The spectralDecomposition(edgelist) function takes an input edgelist array and iteratively calls the spectralDecomp_OneIter function to iteratively partition the graph into communities and returns the partition array indicating the nodes and their communities. It maintains a list of edges within a community and partitions the community into two using the spectralDecomp_OneIter function until the ratio cut between the communities is greater than 0.75. If the number of edges within a community is greater than 10 then the edgelist of the community is marked to be partitioned further.
- The createSortedAdjMat(partition, edgelist) function takes graph partition and edgelist arrays as input and returns the adjacency matrix of the graph after sorting the nodes in accordance with the communities to which they belong.
- The code contains some helper functions taken from the source `https://github.com/multinetlab-amsterdam/network_TDA_tutorial/blob/main/1-network_analysis.ipynb` for visualization of communities within the network elegantly using the networkx library.

- The louvain_one_iter(edgelist) function takes an edgelist array and returns the graph_partition array after partitioning the graph by running one iteration of the Louvain algorithm. It makes use of a number of helper functions for running one iteration of the Louvain algorithm. Some of which are described below, rest functionality can be deduced from the comments within the helper functions:
  - The first_phase(node2com, edge_wts) runs the first phase of the Louvain algorithm and assigns nodes to communities so that the modularity of the network is maximised, initally assuming each node to be a part of its own community. It reassigns nodes to communities of node's neighbors until no increase in modularity can be obtained. It returns the nodes with their reassigned communities.
  - The second_phase(node2com, edge_wts) function merges the nodes in the communities to form a super node and updates the edge weights of the network correspondingly.
  - The working of the rest of the helper functions for the Louvain algorithm can be understood from the function comments in the code.
  -

Algorithm 1 Louvain Algorithm Phase 1

Input: $G = (V, E)$
Output: Graph Partitions

Initialize: Assign every node to its own community
Step 1: for $i \in V$
for $j \in C$
compute ΔQ_{ij} if i moves to j
Step 2: $i^*, j^* = arg\max_{ij} \Delta Q_{ij}$
Step 3: Move node i^* to community j^*

Repeat Step 2 and 3 until no further improvement in modularity is possible.

Results

1. https://snap.stanford.edu/data/ego-Facebook.html dataset
   (a) Spectral Decomposition Technique
   - Running Time : 45.41 secs
   - No of communities detected: 7

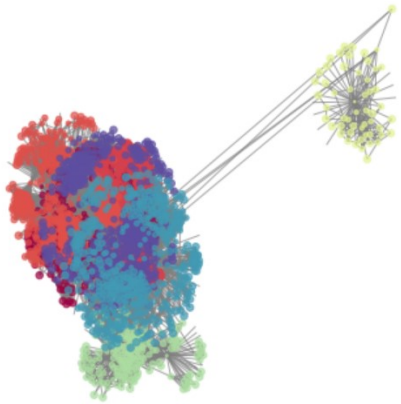

Figure 1: Visualization of communities in the network

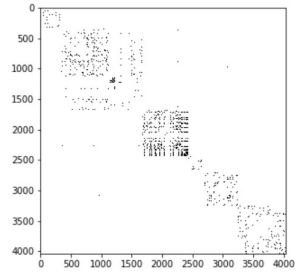

Figure 2: Sorted Adjacency Matrix

(b) Louvain Algorithm(One Iteration)

- Running Time: 88.96 secs
- No of communities detected: 109

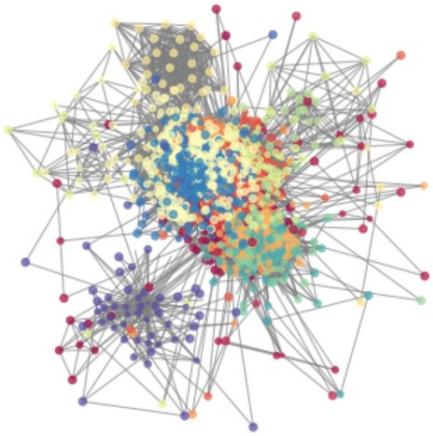

Figure 3: Visualization of communities in the network

2. https://snap.stanford.edu/data/soc-sign-bitcoin-otc.html dataset

   (a) Spectral Decomposition Technique

   - Running Time: 53.84 secs
   - No of communities detected:1

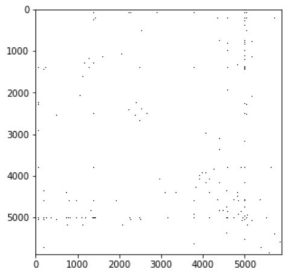

Figure 4: Sorted Adjacency Matrix

   (b) Louvain Algorithm

   - Running Time: 72.72 secs
   - No of communities detected:1208

- We see that one iteration of Louvain algorithm takes close to 2X the time taken by the Spectral Decomposition Technique.
- Since we ran the Louvain algorithm for only one iteration it detects order of magnitude more communities in comparison to Spectral Decomposition.
- In general, the Louvain Algorithm which greedily maximizes the modularity of the network will give rise to better communities (which are hierarchical), but since we did not run the Louvain algorithm to completion and just ran one iteration of it, in our case it seems like Spectral Decomposition gave rise to better communities.