 ISBN: 978-93-86005-02-1

Community Detection and Probability Estimation of Community Connectedness

Ranabir Devgupta* and Arka Prava Bandyopadhyay

CSE Department, RCC Institute of Information Technology, Kolkata - 700015, West Bengal, India

Abstract

We present a methodology to analyze large social data sets based on a new community detection algorithm and based on the communities we find a method to detect the probability of a node to be part of a community. Our main aim is to find the communities based on locally computed score and later fit the scores in a distribution to find the probability of its connectedness in a community. Our work is mostly based on FOCS algorithm. Therefore in this article we refer to the FOCS algorithm often, describe it and then point out the changes brought by us.

Keywords: Community Connectedness, Probability Estimation, Social Network Analysis

1. Introduction

The topic of community detection in social networks has attracted a lot of attention in recent years. Individuals can't stay alone, they form group with like-minded people and that results in communities. A single person can be a part of a large number of communities. Whereas there can be some individuals who do not take part in any community. Thus this community network analysis helps to understand many real life aspects.

One of the best example provided is that telecom service providers always take additional care for a customer who has a strong connectivity in a community even if the individual generates less revenue. The loss of the individual will generate a bad reputation in the community and the other members of the community may soon leave such telecom provider. The marketing sector always targets such communities with large members to market their product. Community detection has a vast range of applications in business intelligence, social analysis, understand political views etc.

Since data in today's world is increasing exponentially and we have to deal with networks with millions/billions of nodes. The method should be faster and should not have memory constraint. Since our main concern is time, we make sure that the algorithm does not take additional time in trivial computations. Existing methods always depict the relationship of two nodes using the snapshot of the network, but these snapshots cannot reveal the real relationships, especially when the

**Author for correspondence*

connection history among nodes is considered. The problem of detecting the stable community in mobile social networks has been studied in this paper. Community cores are considered as stable subsets of the network in previous work. Based on these observations, this paper divides all nodes into a few of communities due to the community cores. Meanwhile, communities can be tracked through incremental computing. In a social system, individuals tend to group with others who are like-minded or with whom they interact more regularly and intensely than others. This process leads to the formation of communities. In a community the participant actors are densely connected to each other, whereas nodes that belong to different communities do not interact much. Furthermore, nodes with interests and purposes in different fields result in overlapped communities. Here we build the communities with the help of some features using them as scores. The problem of community detection is to identify naturally existing groups of actors such that nodes within a group are densely connected with each other while being sparsely connected to the nodes that belong to different groups. Experimental results based on real-world social networks demonstrate that our proposed method performs well. Once we can fit the distribution, then the score based on its connectedness can help to determine the what is the probability of a node based on its connectedness score to be part of any community.

2. Related Work

As the network sizes are increasing rapidly, consisting of millions/billions of data, we can't apply graph clustering methods to them. It can only be applied to small communities. A number of graph clustering methods are present to detect communities. In[4] created a partition of network into clusters based on compression of information on random walk taken on the network. In[5] found hierarchical disjoint communities in massive networks based on modularity optimization.

There are also methods which aim to find n-cliques, k cores as communities but these methods are extremely computationally expensive.

Label propagation algorithm assigns a unique label to each node and then assigns maximum share of their neighbour. COPRA[6] modified classical LPA so that each node can retain multiple labels in order to find overlapped community. But these mainly find disjoint clusters. The ones which find overlapped clusters are computationally expensive. We trace our work based on FOCS which evolves on the basis on locally computed scores and scales well over large sized networks.

FOCS algorithm[1]:
A graph G(V,E) is assumed to be simple that is it should not contain any self loop or parallel edges. FOCS solves the problem of community detection by finding a set of subgraphs in the network. The subgraphs should be such that, the nodes in the subgraph are more connected than in any other network.

FOCS works in four phases namely initialization, leave, expand, delete phase.

The problem of community detection is to find family of subgraphs $S = \{Si|Si \subset V\}$ such that for any node vj in a subgraph Si, it is more *connected* in the subgraph Si than in another subgraph Sj′. Here, $S'j = (Sk|vj \in / Sk \wedge Sk \in S)$ is any subgraph in family S not containing node vj. Each subgraph $Si \in S$ is a community.

	community connectedness score		
neighborhood connectedness score		low	high
	low	less interest and low belongingness (not this community)	less interest but high belongingness (strong community with few neighbors)
	high	high interest but low belongingness (not accepted by community)	high interest and high belongingness (node is central to the community)

For each node vj, $\forall j \in \{1, 2, .., |V|\}$, let $S(vj) = \{Si | vj \in Si \wedge Si \in S\}$ be the collection of communities containing node vj. Further, let $S'(vj) = S - S(vj)$ be the collection of communities not containing node vj. If each node v j belongs to exactly 1, or no community at all, i.e., $|S(vj)| \leq 1$, then it is called disjoint clustering, overlapped clustering otherwise. FOCS algorithm, proposed in this paper, explores overlapped clusters in a given graph.

Initially every node vi, $\forall i \in \{1, 2, .., |V|\}$, that has atleast K neighbors, builds a community Si with its neighbors. The initial community is denoted as S_o.

N(vj) be the set of neighbors of a node $vj \in V$.

$$N(vj) = \{vk | (vj, vk) \in E\}$$

Now, let Ni(vj) be the within community neighborhood of node vj defined for community $Si \in S(vj)$ as follows:

$$Ni(vj) = \{vk | (vj, vk) \in E \wedge vk \in Si\}$$

The community connectedness score $\tilde{\zeta}_j^i$, thus, assigned to each node vj in each community $Si \in S$ is,

$$\tilde{\zeta}_j^i = |Ni(vj)| / |Si| - 1$$

A node leaves a community if it's is not sufficiently connected in a network. A node is said to be connected in a network if its community connectedness score is greater than the cut off score known as the stay cut-off. The stay cut off is chosen with the means of bucket. The entire range of scores is divided into max (20, N(vj)) number of buckets of equal sizes. Initially the score of bucket is 0, and is incremented as a score falls in the range. A bucket with count greater than 0 is marked and is scanned left until, a bucket that has a count lesser than or equal to that of marked bucket is found and the count of the bucket to its left is greater than or equal to that of the current one, the leftmost bucket is reached. The lower bound of the bucket is the stay cut off.

The neighbourhood connected score is defined as

$$\xi_j^i = |N_i(v_j)| / |N(v_j)|$$

The neighbours of the peripheral nodes are chosen and their ratio of total number of connections in the proposed community to the degree of the node is calculated. This is neighbourhood connectedness score. The expand phase works similarly as when a node with neighborhood connectedness score greater than join cut off is found, it is added to the community. The join cut off is calculated similar to stay cut off but here the only difference lies in the fact that the lower quartile is chosen as the join cut off.

Delete phase mainly removes duplicate communities in a network. If two communities have similar members, the community with the minimum number of members are deleted.

Duplication removal helps the computation in many ways, as it removes the unnecessary community members.

OVL as input. OV L sets a threshold for the maximum overlap allowed between two communities, before they can be identified as *near-duplicates*. The smaller of the two communities Si and Sj is deleted when similarity measure (Si, Sj) crosses this threshold. An OV L = 1 implies elimination of a duplicate community when it is exactly identical to another.

Reasonably, OV L must be set to ≥ 0.5 and less than 1 for early identification of duplicates in community structure. We have taken OV L = 0.6 in our work. However, we have experimented with different values of OV L and observed stability in output in qualitative terms when set in the range 0.5 to 0.7.

The entire iteration stops when there are no more peripheral nodes left. to leave or to expand. The final communities are taken and their connectedness scores are fitted into a distribution that best fits them. Once the distribution is found out, the probability of a node with certain score to be a part of any community can be easily found out. This

probality estimation also given us a range of community connectedness scores for which a node will have high probability of being part of any community.

Algorithm

Input: G = (V, E): input graph, K: minimum connectionsfor a node within a community, OV L: maximum allowed overlap between communities
Output: S = {Si|Si ⊆ V and Si is a community}
Addedi = Nodes added to community Si in last round
1: **procedure** PREFERREDCOMMUNITIES
2: InitializeCommunities
3: **while** peripheral communities present
4: Perform LeaveCommunities operation
5: perform ExpandCommunities operation
6: perform delete community operation
7: **end while**
8: **end procedure**
9: **function** INITIALIZECOMMUNITIES
Community of each node in initialized by the node v ∈ V and its neighbors N(v) if |N(v)| ≥ K
10: **for** each i ∈ {1, 2, .., n} **do**
11: **if** |N(vi)| ≥ K then
12: Si = {vi} S N(vi)
13: **end function**
14: function LEAVECOMMUNITIES
/* In each community Si node v ∈ Si leaves Si if its community connectedness score is less than stay cut-off.Updated communities of size less than K are deleted*/
15: Eliminate near-duplicate community
16: Compute community connectedness scores and neighborhood connectedness scores
17: Compute stay cut-off
18: **for** each community Si ∈ S **do**
19: **if** the node has connectedness score less than cut off score It stays in the community
20: **else**
21: the node leaves
22: **end if**
23**: end for**
24: **end function**
25: **function** EXPANDCOMMUNITIES
/*For each community Si, each adjacent u ∈ N(v) of each node v ∈ Addedi is included in Si if u is not in Si and it builds up a neighborhood connectedness score greater than its join cut-off */
26: Compute join cut-off
27: **for** each community Si ∈ S **do**
28: **if** the node has neighbourhood connectedness score > join cut off
Then the node joins the community
29: **end if**
30: **end for**
31**: end function**

3. Method

3.1 Probability Estimation

The entire iteration stops when there are no more peripheral nodes left. to leave or to expand. The final communities are taken and their connectedness scores are fitted into a distribution that best fits them. Once the distribution is found out, the probability of a node with certain score to be a part of any community can be easily found out. This probability

estimation also given us a range of community connectedness scores for which a node will have high probability of being part of any community.

4. Conclusion

Community detection is the core methods in any Social Network analysis. The process that we follow here is a novel method because of its simplicity. There are other methods too for community detection using k-clique method. But such methods increase the computation time and in large growing network which is doubling every year it would become quite impossible to implement methods which are efficient in terms of analyzing but not in terms of computation time.

One of the key features used in community detection is scores. Various other strategies can be developed or some other entities can be added to make the cut off score more stringent. It depends on the analyzer whether to analyze and detect communities negatively (make the conditions stringent to develop small communities) or positively (make loose conditions and develop large communities).

Therefore we take up the solution through local optimization. Once the model can be fitted in any probability distribution model, the probability of any node being part of any community can be easily found out. Thus to analyze a community any random member can be easily identified whether it is a part of any community or not.

5. References

1. Bandyopadhyay S, Chowdhary G, Sengupta D. FOCS: Fast Overlapped Community Search. IEEE Transactions on Knowledge and Data Engineering.
2. Lozano S, Duch J, Arenas A. Community detection in a large social dataset of European Projects.
3. Dhillon IS. Multi-scale spectral decomposition of massive. Department of Computer Science, University of Texas at Austin.
4. Rosvall M, Bergstrom CT. Maps of random walks on complex networks reveal community structure. Proceedings of the National Academy of Sciences. 2008; 105(4):1118–23.
5. Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E. Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment. 2008; 2008(10):10008.
6. Gregory S. Finding overlapping communities in networks by label propagation. New Journal of Physics. 2010; 12(10):103018.