Graph, Algorithms, and Models

Andrea Civilini, Xinyi Xu, and Sam Creedon

Whenever

Overview

Intuitive Definition

- Intuitively, given a graph $G = (V, E)$, we say that G has a community structure if V can be partitioned into subsets called communities, say $\{C_1, C_2, \ldots, C_k\}$, where the nodes of a given community C_i are more densely connected to one another than to the nodes in other communities.
- Of course such a definition is open to interpretation, especially when trying to express it in a more rigours manner. As such there is no universal definition of a community, but many interpretations share common features.
- An important class of motivating examples of networks with community structure are those of social networks, where many natural communities would arise due to location, hobbies, occupation, etc.

Generalisations and Variations

- There are many generalisations and related concepts in the research of communities in networks:
  - Communities in weighted or directed graphs
  - Overlapping communities
  - Hierarchical community structure
  - Evolution of community structure over time
- Our main focus will be on undirected graphs with no weighted edges. Also the community structure we will be examining will be focused on partitions of the vertices.

Why study Communities

First and foremost, the importance of studying networks is made clear due to their large use in numerous disciplines, ranging from natural and social sciences, to computer sciences and engineering. Community structure appears in many networks throughout such disciplines, and is certainly a large feature in many real world networks. As such the topic of communities has seen significant interest in recent times. Understanding the underlying communities which make up a network allows us to:

- Gain a better picture of the entire network as a whole
- Obtain local data on how the network operates
- Identify communities with properties which strongly differ from the average properties of the network.

Modularity

- Suppose we have a potential grouping of a network into communities, say $\mathcal{P} = \{C_1, C_2, \ldots, C_k\}$.
- A quality function is an assignment of such a partition to a numerical value, with the intention of describing whether such a partition is a good fit for grouping the network into communities.
- The most popular quality function is called *Modularity*. It is defined by

$$Q_{\mathcal{P}} = \frac{1}{2K} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(a_{ij} - \frac{k_i k_j}{2K} \right) \delta(C_i, C_j).$$

- Here a_{ij} are the elements of the adjacency matrix, k_i is the degree of node i, and δ is the Kronecker delta function.

Modularity

- What the *Modularity* function is doing, is evaluating the difference between the internal density of each community with the expected density within a random network.
- The random network most commonly used, and the one used to derive the previous formula, is a network where nodes are connected uniformly at random, but we have the same number of nodes as the original network, and where each node has exactly the same degree as the original network.
- In a sense, *Modularity* is comparable to statistic significant testing, as it gives us a measure on how different the internal density of the communities are compared to how they are in a random graph which retains some fundamental structure of the original network. As such positive modularity suggests a "good" partition, and the higher the better.

Checking all Partitions of a Graph

- A natural question now is how one would go about finding a partition of a network into communities.
- Simply calculating all partitions of the vertices of a network is highly impractical.
- The number of partitions of a set of size n is given by B_n, the n^{th} Bell number. From some combinatorial arguments one can show the following:

$$B_n = \sum_{k=1}^{n-1} \binom{n-1}{k} B_k \geq \sum_{k=1}^{n-1} \binom{n-1}{k} = 2^{n-1}.$$

- This highlights the fact that B_n has 2^n as an asymptotic lower bound, i.e. B_n grows at best in exponential time, which is very "costly" in terms of computation. In particular, enumerating all partitions of a set of vertices of a network is practically impossible with the few exceptions when the network has a very low node count.

Girvan-Newman Method

- The Girvan-Newman algorithm was one of the first algorithms created to find community structure within a network.
- It is based on the idea that the edges between communities are sparse, and hence many shortest paths between nodes within the network have a higher chance of passing through these "bridges" between the communities. So if one could find such edges and remove them, we would separate the communities into connected components, allow us to identify them.

Girvan-Newman Method

- To find such edges, we assign a value to each edge referred to as the *betweenness* of the edge. It is the number obtained by dividing the number of shortest paths passing through the given edge by the total number of shortest paths.
- The edge *betweenness* gives us a measure of the importance of the edge in traversing the network. As such, we expect the few edges which connect the communities together to have high *betweenness*.
- The Girvan-Newman algorithm is then given as follows:
  (1) Evaluate the betweenness of each edge in the network
  (2) Remove the edge of maximal betweenness (chosen arbitrarily if draws occur)
  (3) Re-evaluate the edge betweenness of each edge in the new network obtained from the above step
  (4) Repeat steps 2 and 3 above until no more edges remain

Girvan-Newman Method

- The connected components making up the graphs obtained in the above process give us a collection of potential partitions of the original network into communities (in fact a dendrogram). Using modularity one can find the best candidate out of the collection.
- Remarks:
  - Has a rather high time complexity of $\mathcal{O}(K^2 N\log(N))$ where $N = |V|$ and $K = |E|$.
  - Has natural generalisations to directed graphs
  - Not as easy to generalise to covers ("overlapping partitions"), but there do exist such modified versions
  - Since we obtain a dendrogram, we obtain hierarchical knowledge on the community structure.

WalkTrap

- There are numerous algorithms which employ random walks on a network as a means of finding communities. We give a very brief and intuitive summary of such an algorithm called WalkTrap.
- Undergoing a random walk in a network, one would expect to be trapped (spend a lot of time) within densely connected areas, i.e. communities. So the idea behind WalkTrap is to try and exploit this behaviour of random walks as a way of identifying what collection of nodes are likely to be members of the same community.
- An important ingredient in this algorithm is the *transition matrix* P, whose ij-th entry gives the probability of travelling from node i to j in a single step. Then for some length t, the matrix P^t describes the probabilities of going from one node to another in t steps.

WalkTrap

- A length of the random walks, t, is fixed. Then one uses (skipping the details) the matrix P^t to define a distance among the nodes of the network. The idea is that nodes of the same community are considered "close", while nodes of distinct communities are considered "far away". The notion of distance can be generalised to subsets of nodes.
- Then the WalkTrap algorithm is intuitively given as follows: Start with the finest partition $\mathcal{P}_1 = \{\{v\}|v \in V\}$. Compute the distances between all adjacent nodes. Then this partition evolves by repeating the following operations. At each step k:
  - Choose the two communities C_1 and C_2 of $\mathcal{P}_k$ whom uphold some minimality condition based on the distance function.
  - Merge these two communities into a new community $C_3 = C_1 \cup C_2$ and create the new partition $\mathcal{P}_{k+1} = (\mathcal{P}_k \backslash \{C_1, C_2\}) \cup C_3$
  - Update the distances between the communities and repeat

WalkTrap

- Just like with the Girvan-Newman algorithm, we obtain a collection of partitions of the network which form a dendrogram. We start with the finest partition into singletons, and end with the partition containing a single community containing all nodes. The partitions obtained along the way have been chosen in a greedy fashion, trying to minimise some distance condition between the communities.
- Using modularity one can find the best candidate out of the collection, or even a stopping point of the dendrogram.
- Remarks:
  - At worst it's time complexity is $\mathcal{O}(KN^2)$. For sparse networks this can be improved to $\mathcal{O}(N^2 \log(N))$.
  - Hierarchical community structure obtained via the dendrogram
  - Can be directly usable for weighted networks
  - Surprising much of the proofs and computations are not upheld in the case of directed graphs
  - If the graph is very large, instead of computing P^t directly, one can obtain an approximation by sampling many random walks on the graph.

Modularity Optimization

- As seen in the previous examples, we use modularity as a means to pick out the best partition from a collection. This is often done for algorithms which find community structure, since modularity is a very popular concept within this field of study.
- One might suggest to make an algorithm with the objective of maximising modularity from the get-go. This is the goal of Modularity Optimization algorithms. There are many such algorithms, but one of the most popular ones is given by Newman, which is simply the greedy algorithm employed in a natural fashion.

Modularity Optimization

- We start with finest partition $\mathcal{P}_1 = \{\{v\}|v \in V\}$. We obtain a series of partitions by, at a given stage k with partition $\mathcal{P}_k$, we obtain a new partition by:
  1. Evaluate the modularity of all partitions obtainable from $\mathcal{P}_k$ by merging a pair of communities
  2. Let the new partition $\mathcal{P}_{k+1}$ be the one of maximal modularity from the partitions of (1) above
  3. Repeat until we have $\mathcal{P} = V$ (at stage $n-1$)
- Remarks:
  - With the use of clever data structures to maintain and update the information regarding merging and modularity increases, the time complexity is $\mathcal{O}(K \log^2 N)$.
  - Since modularity can be generalised to both directed and weighted graphs, so can this algorithm.
  - Hierarchical community structure obtained via the dendrogram

The Label Propagation algorithm

- The Label Propagation algorithm is a community finding algorithm. It works by propagating labels throughout the network and forming communities based on this process of label propagation.
- The idea behind the algorithm is that a single label can quickly become dominant in a densely connected area, but will have trouble crossing a sparsely connected region. Labels will get trapped inside a densely connected group of nodes, and those nodes that end up with the same label when the algorithms finish can be considered part of the same community.
- The algorithm goes as follows:
  (1) Give every node a distinct label
  (2) At every iteration of propagation labels, each node updates its label to the one that is most popular with its neighbours (draws are broken uniformly at randomly).
  (3) The process terminates when each node has the majority label of its neighbours.

The Label Propagation algorithm

- As labels move around, densely connected areas of nodes quickly reach a consensus on a unique label. At the end of the algorithm nodes that have the same label at are said to belong to the same community.
- Remarks:
  - The algorithm can be semi-supervised by pre-assigning nodes labels. This gives great flexibility in its use.
  - The time complexity is $\mathcal{O}(EK \log(N))$, where E is the number of iterations of the process.
  - May get trapped in loops, but there are means to remedy this issue.

Benchmark Networks

- The algorithms we have discussed are constructed with fundamental intuitive properties of what it means to be a community in mind. As such we expect that these algorithms would serve well in finding community structure in networks when they are present. However we have not yet mentioned whether the algorithms are indeed good at their job.
- The main reason for this is that, at the moment, there is no known best way of validating an algorithm for finding communities. This mainly comes down to the fact that there is no unifying definition of community.
- So the most popular way of testing out an algorithm and comparing it to others it by running it on benchmark networks whose underlying community structure is known, and comparing the known structure to the one obtained by the algorithm.

Benchmark Networks

- There are both computer-generated benchmark networks and real-world ones. The most used benchmark networks range the community structure in precise ways to try an isolate certain difficulties and aspect one would want a community finding algorithm to over come if it is to be considered good.
- Many popular benchmark models are inspired by what is called the Stochastic Block Models. The idea is we pre-group the nodes, and assign an edge to two given nodes with probability depending on what groups the nodes belong. A simple case would be to have two probabilities, p_{in} for pairs of nodes within the same group, and p_{out} for pairs of nodes of distinct group, and having $p_{in} > p_{out}$.
- Such a random graph would thus be built with a community structure in mind, and the greater the difference between p_{in} and p_{out}, the easier we would expect the communities to be detectable.

Benchmark Networks

- Ranging the size of the groupings (possibly of different sizes), and changing various values for the probability of edges between different groups, would give scores of potential benchmark networks to work with.
- Many of the popular computer-generated benchmarks are special cases of the above random graph construction.