CSCI-GA.G22.2340: ELEMENTS OF DISCRETE MATHEMATICS

AGGLOMERATIVE METHODS FOR COMMUNITY DETECTION THROUGH MODULARITY MAXIMIZATION

August 9, 2016

Aditi Nair
Net ID: asn264
New York University

Contents

ABSTRACT

In this paper, I will briefly present the problem of community detection in networks and derive "modularity", a commonly-used metric in community detection methods. I will present two agglomerative, hierarchical algorithms - Newman's Greedy Algorithm and the Louvain Method - which recover community structures by approximately maximizing network modularity. Finally, I will discuss results from each algorithm on the popular Karate dataset.

INTRODUCTION TO COMMUNITY DETECTION

The advent of online networks has fundamentally changed the way that people connect and communicate with each other. As in real-life, people tend to form communities within these networks - that is, groups of users within which members are generally more closely-connected with each other than they are with those outside of the community. The science of attempting to discover such communities within networks is called "community detection". Community detection may be used to recommend friends or connections to users of social networks, to detect communities of criminals in online networks, and even to recover emerging research fields from citation networks.

Network scientists represent such networks by graphs. For example, in a social network graph, individuals are represented by nodes and connections between them are represented by edges. Such graphs may be directed - for example, to reflect one individual "following" another on Twitter - or un-directed - for example, to reflect that two individuals are friends on Facebook. These graphs can also be weighted or un-weighted depending on the precise relationships that they attempt to model. Discovering communities within such networks is equivalent to finding partitions of graph nodes such that the partitions generate subgraphs that satisfy some measure of "community-ness".

The optimal number of partitions or communities for a given network is usually unknown; it is even possible that the optimal choice is to place the entire network into a single community. Therefore the problem of community detection involves choosing an optimal number of partitions as well as finding a partition that respects some notion of community. For this reason, community detection can be prohibitively expensive for large networks. Given an extremely large graph - that is, a graph with many nodes - for a fixed number of communities, there is often a large number of permutations of communities that may be formed. Since the optimal number of communities is unknown, this already-costly process must be repeated

for every possible number of communities. Moreover, in practice, large networks are often extremely complex and do not organize themselves into clear and discrete communities - meaning that one partition of graph nodes may not be preferable to another partition in an obvious or accessible way.

EDGE-COUNTING AND MODULARITY FOR COMMUNITY DETECTION

It was been mentioned briefly that community detection methods attempt to partition graphs into subgraphs or communities within which nodes are more connected to each other than to nodes outside of the communities. Next, we will make this notion of "community-ness" more precise.

Intuitively, it is apparent that the number of edges within and outside of a community should be indicative of how "tightly-knit" a community is: an optimal partition of nodes would produces many edges within communities and few edges between communities. Accordingly, to measure how "good" any given partition is, we could simply compare the aggregate number of edges within communities to the number of edges between communities - optimizing for larger values of the former and smaller values of the latter.

Though this method is intuitively satisfying and simple, it should be apparent that it is also insufficient. For any network, we could simply partition the network into exactly one community. Then the number of edges within the community would equal exactly the number of edges in the network and there would be zero edges between communities. Though this would be an optimal partition according to our edge-counting metric, it is generally trivial and, in most cases, would fail to identify interesting communities within networks, were they to exist.

Instead, we ask: is the number of edges within communities greater than what would be expected if edges were distributed between nodes at random? (Equivalently, we might ask: is the number of edges between communities less than what would be expected if edges were distributed between nodes at random?) That is to say, in a network lacking any community structure, we would find that for any possible partition of network nodes, the distribution of edges within and between communities would be completely random. However, if there were a partition of nodes that exhibited significant community structure, the distribution of edges within and between communities would respectively be significantly greater than and less than if the distribution of edges were completely random.

MODULARITY: DEFINITION AND DERIVATION

We quantify the above notion with the modularity metric Q. Note that the methods discussed can be applied to both weighted and un-weighted un-directed graphs. Though we will generally use the language of un-weighted graphs, the same reasoning can be applied to weighted graphs simply by noting that an edge between two nodes with weight x can be viewed as x distinct edges of weight 1 between the same nodes. (We can also apply the methods discussed to directed graphs by simply ignoring the directions attached to edges.)

The modularity Q equals the fraction of edges observed within communities minus the fraction of edges expected within communities, were the distribution of edges completely random. Q can take any value between 1 and -1. It is positive when there are more edges within communities than if the edges were randomly distributed and negative when there are fewer edges within communities than if the edges were randomly distributed. A modularity value of 0 suggests that there is no deviation from random chance.

To derive an equation for Q, first consider an arbitrary network with some fixed number of communities. Let A_{ij} be the weight of the edge between node i and node j. (A_{ij} is only 0 when there is no edge between i and j. In a network represented by an un-weighted graph, A_{ij} will be either 0 or 1.) Also recall the Kronecker delta function δ defined as:

$$\delta(x,y) = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases} \tag{1}$$

Let c_i be the community label for node i. Then the total number of within-community edges in the network is:

$$\frac{1}{2}\sum_{ij} A_{ij} \cdot \delta(c_i, c_j) \tag{2}$$

In the expression above, we count the number of edges between each pair of nodes (i, j) with A_{ij}. If i and j belong to the same community, then $\delta(c_i, c_j)$ will be 1, and we will add A_{ij} to the total count of within-community edges in the network. If i and j do not belong to the same community, then $\delta(c_i, c_j)$ will be 0 and we do not add A_{ij} to the count of within-community edges. Because $A_{ij} = A_{ji}$ and $\delta(i, j) = \delta(j, i)$, the sum over all pairs (i, j) double-counts the number within-community edges - so we divide by two.

Next, let k_i be the degree of node i. Then, m, the total number of edges in a network, is defined as:

$$m = \frac{1}{2}\sum_{i} k_i \tag{3}$$

This definition follows because the degree of a node i indicates the number of edges that connect node i to another node in the graph. In the expression above, by adding the degrees of all nodes in the graph, we double-count each edge in the graph. To find the total number of edges, we simply divide the sum of node degrees by two.

Now we can write the expected total number of within-community edges in a network with randomly-distributed edges as:

$$\frac{1}{2}\sum_{ij} \frac{k_i k_j}{2m}\delta(c_i, c_j) \tag{4}$$

To derive this expression, it is important to realize that we are still assuming that the network contains the same total number of edges. We want to understand the community structure of the network given that the same number of edges are *randomly placed* between nodes.

Note that in a network with randomly-distributed edges, the number of edge-ends or stumps in a network is $2m$ because each edge can be split into two stumps that connect only to a single node and extend outward from that node. Moreover, any node i with degree k_i has k_i number of stumps attached to it. If we arbitrarily choose a single stump from node i, we can say that the likelihood of that stump connecting to a stump of node j with degree k_j is:

$$\frac{k_j}{2m} \tag{5}$$

This expression follows because every stump must connect to another stump and there are $2m$ such stumps in the network. However, only k_j stumps can actually connect the arbitrarily chosen stump of node i to node j.

Next, because node i has degree k_i, there are actually k_i possible stumps extending from node i which could connect node i to node j. Therefore the total number of expected edges between node i and node j in a network with randomly-distributed edges is:

$$\frac{k_i k_j}{2m} \tag{6}$$

Finally, we only want to count the number of edges expected *within* communities, given a random distribution of edges. Therefore we must modify the above expression to only count nodes that are within the *same* community. Recalling that c_i and c_j equal the community labels of nodes i and j respectively, we can apply the Kronecker delta function to compute

the expected number of within-community edges between node i and node j:

$$\frac{k_i k_j}{2m}\delta(c_i, c_j) \tag{7}$$

Finally, the total expected number of edges observed within communities, were the distribution of edges completely random, is:

$$\frac{1}{2}\sum_{ij}\frac{k_i k_j}{2m}\delta(c_i, c_j) \tag{8}$$

The above expression is derived simply by totaling the expression in (7) over all pairs (i, j), then dividing the sum by 2. We must divide by 2 because the expected number of randomly-distributed edges between node i and node j is exactly equal to the expected number of randomly-distributed edges between node j and node i, and the summation double-counts this value.

Finally, we can write the difference between the number of within-community edges observed in the network (expression (2)) minus the number of within-community edges expected in a network with randomly-distributed edges (expression (8)) as:

$$\frac{1}{2}\sum_{ij}A_{ij}\cdot\delta(c_i, c_j) - \frac{1}{2}\sum_{ij}\frac{k_i k_j}{2m}\delta(c_i, c_j) = \frac{1}{2}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j) \tag{9}$$

Recall that the modularity Q was defined as the *fraction* of edges observed within communities minus the *fraction* of edges expected within communities in a network with randomly-distributed edges. Therefore we must divide expression (9) by the total number of edges m:

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j) \tag{10}$$

A Practical Reformulation of the Modularity Metric

Entire networks are often described simply by a square matrix A which contains the values of A_{ij} for all possible pairs (i, j) at index ij. (Such a matrix is known as an adjacency matrix.) Though the above definition of the modularity metric is a popular theoretical approach, it can be cumbersome to apply in such situations, since we would have to manually compute the degree k_i of each node i. Given the large size of many networks of interest in the field of community detection, this can be unwieldy and, as we will show below, unnecessary.

First we define e_{rs}, the fraction of edges between community r and community s:

$$e_{rs} = \frac{1}{2m} \sum_{ij} A_{ij} \delta(c_i, r) \delta(c_j, s) \tag{11}$$

where $\delta(c_i, r)$ is 1 only if node i belongs to community r and $\delta(c_j, s)$ is 1 only if node j belongs to community s. Since A_{ij} is the number of edges between node i and node j, $A_{ij}\delta(c_i, r)\delta(c_j, s)$ will only be nonzero if node i is in community r, node j is in community s, and there is at least one edge between node i and node j. Since the summation iterates over all possible pairs (i, j), it counts every edge twice, so we must divide by 2. Finally, since we want the *fraction* of edges between community r and community s, we divide the entire expression by m, the total number of edges in the network.

Next define a_r, the fraction of edge stubs that belong to community r:

$$a_r = \frac{1}{2m} \sum_{i} k_i \delta(c_i, r) \tag{12}$$

Notice above that $\delta(c_i, r)$ is non-zero only for nodes i which belong to community r. Since k_i is the number of stubs belonging to node i, the sum of $k_i\delta(c_i, r)$ over all i is the total number of stubs in community r. Since m is the number of edges in the network, $2m$ is the number of stubs in the network, and we can divide the summation by $2m$ to get the fraction of network stubs that belong to community r.

Finally, we observe the following:

$$\delta(c_i, c_j) = \sum_{r} \delta(c_i, r) \delta(c_j, r) \tag{13}$$

To prove this, we consider the case $\delta(c_i, c_j) = 0$. This implies node i and node j do not belong to the same community. We can say that node i belongs to community r_1 and node j belongs to community r_2 such that $r_1 \neq r_2$. Therefore $\delta(c_i, r_1) = 1$ and $\delta(c_j, r_2) = 1$ but $\delta(c_j, r_1) = 0$ and $\delta(c_i, r_2) = 0$. Since each node belongs to exactly one community, this means there is no community r such that $\delta(c_i, r) = \delta(c_j, r) = 1$. Instead, for all r, at least one of $\delta(c_i, r)$ or $\delta(c_j, r)$ will always be zero, and the summation above will equal zero. Therefore $\delta(c_i, c_j) = 0$ implies $\sum_r \delta(c_i, r)\delta(c_j, r) = 0$.

In the case that $\delta(c_i, c_j) = 1$, then node i and node j must belong to the same community s. Then $\delta(c_i, s) = \delta(c_j, s) = 1$, but $\delta(c_i, t) = \delta(c_j, t) = 0$ for all communities t such that $t \neq s$. Therefore the summation above will be the sum of all zeroes except in the case that $r = s$, when $\delta(c_i, s)\delta(c_j, s) = 1$. Therefore $\delta(c_i, c_j) = 1$ implies $\sum_r \delta(c_i, r)\delta(c_j, r) = 1$.

Finally, we can use the above observations to derive a new formulation of the modularity

metric Q:

$$Q = \frac{1}{2m}\sum_{ij}\left(A_{ij} - \frac{k_i k_j}{2m}\right)\delta(c_i, c_j) \tag{14}$$

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

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

$$= \sum_{r}\left(\frac{1}{2m}\sum_{ij}A_{ij}\delta(c_i, r)\delta(c_j, r) - \frac{1}{2m}\sum_{i}\sum_{j}\frac{k_i k_j}{2m}\delta(c_i, r)\delta(c_j, r)\right)$$

$$= \sum_{r}\left(\frac{1}{2m}\sum_{ij}A_{ij}\delta(c_i, r)\delta(c_j, r) - \sum_{i}\frac{k_i}{2m}\delta(c_i, r)\sum_{j}\frac{k_j}{2m}\delta(c_j, r)\right)$$

$$= \sum_{r}\left(e_{rr} - a_r^2\right)$$

Now, in order to compute Q, for each community r, we only need to know the values for e_{rr}, the fraction of network edges between nodes within community r, and a_r, the fraction of edge stubs in community r. Both values are easy to compute given an adjacency matrix A. e_{rr} can be found simply by adding the number of unique edges along the diagonal of A which belong to community r, and dividing by the total number of edges. a_r can be found by counting the number of non-zero A_{ij} in columns that represent nodes in community r, and dividing by twice the number of edges in the network. Given the structure of the data, both computations are more straightforward than parsing the matrix to compute the degree of each node.

APPROXIMATION ALGORITHMS FOR MODULARITY MAXIMIZATION

As discussed earlier, larger positive values of modularity indicate that there are more edges within communities in the network than if edges were randomly distributed in the network. Therefore a reasonable community detection algorithm might directly maximize modularity over all possible network partitions. Unfortunately, as discussed in the Introduction, for a fixed number of communities, there are a huge number of permutations of partitions of network nodes that would need to be evaluated. This problem is further exacerbated by the fact that the optimal number of communities is unknown - so we would also have to search over all possible numbers of communities, equal to the number of nodes in the network.

Both issues are even further exacerbated by the large size of networks often used in modern community detection research.

As a result, network scientists generally resort to approximation algorithms which do not fully optimize over all possible partitions, but often succeed at finding partitions of nodes which indicate strong community structure according to the modularity metric. Below, we present two approximation algorithms for modularity maximization which take a hierarchical and agglomerative approach to community detection.

NEWMAN'S GREEDY ALGORITHM

This algorithm is an agglomerative method, which means it begins by placing each node into its own community. A graph with n nodes will begin the algorithm with n communities. Next we combine two of these communities by choosing the combination that results in the greatest increase (or smallest decrease) in modularity. The first combination will give us a network with n nodes and $n-1$ communities. We iteratively combine the modularity-maximizing pairs of communities until the network is divided into a single community. Finally, for each possible number of communities, we will have a community division of the network and a corresponding modularity value. We choose the community structure which exhibits the largest modularity value.

At each step, for each community s, we must evaluate the change in network modularity that results from combining community s with some community t into a community u. Note that $e_{uu} = e_{st} + e_{ts} + e_{ss} + e_{tt}$ since the fraction of edges between u and itself is equal to the sum of the fraction of edges from s to t, from t to s, from s to itself, and then from t to itself. Moreover the fraction of edge stubs that belong to community u is simply $a_s + a_t$, the sum of the fractions of edges belonging to community s and t. We can say that the modularity Q' corresponding to the new network structure is:

$$Q' = \left(\sum_{r \notin \{s,t\}} (e_{rr} - a_r^2) \right) + e_{st} + e_{ts} + e_{ss} + e_{tt} - (a_s + a_t)^2 \tag{15}$$

$$= \left(\sum_{r \notin \{s,t\}} (e_{rr} - a_r^2) \right) + e_{st} + e_{st} + e_{ss} + e_{tt} - (a_s^2 + a_t^2 + 2a_s a_t)$$

Then it follows that:

$$\Delta Q = Q' - Q \tag{16}$$

$$= \left(\left(\sum_{r \notin \{s,t\}} (e_{rr} - a_r^2) \right) + e_{st} + e_{ts} + e_{ss} + e_{tt} - (a_s^2 + a_t^2 + 2a_s a_t) \right) - \sum_r \left(e_{rr} - a_r^2 \right)$$

$$= e_{st} + e_{ts} - 2a_s a_t = 2(e_{st} - a_s a_t)$$

which can be computed in linear time.

Because modularity is the difference between the fraction of edges observed within communities and the fraction of edges expected within communities in a network with randomly-distributed edges, we observe that there is no change in modularity from combining two communities that do not have an edge between them; the fraction of within-community edges observed would remain constant as would the expected fraction of within-community edges in a randomly-distributed network. Therefore, the algorithm only requires that at each step we check the resulting change in modularity from connecting two communities that already share an edge. Therefore, at each step of the algorithm, in a network with m edges, we must compute ΔQ for at most m combinations. Once the modularity-maximizing combination is chosen, we must update the values for e_{rr} and a_r for all communities r, which takes $O(n)$ time in the worst case for a network with n nodes. Finally, each step of the algorithm takes $O(m+n)$ time in the worst case. Since the algorithm is agglomerative and hierarchical, each step must be repeated $n-1$ times to start with n communities and end with just 1 community. Therefore, in the worst case time the algorithm takes $O((m+n)(n-1))$ time. This complexity can also be expressed as $O(n^2)$. [2]

THE LOUVAIN METHOD

The Louvain Method is another method for community detection through modularity maximization that has exhibited strong results on popular datasets and a quick runtime. Like Newman's algorithm, it is agglomerative, meaning that it begins by placing each of the n network nodes into n distinct communities of size one.

In the first step of the algorithm, for each node i in the network, we evaluate the change in modularity that results from moving i from its original community into any other community j. As noted earlier, combining nodes and communities that do not share edges does not change the network's modularity, so it is only necessary to evaluate the change in modularity that results from moving node i to a community j that i shares an edge with. Amongst all communities j for which the combination results in a positive change in modularity, i is placed into the community that exhibited the maximal increase in modularity. If there are no such communities, then i remains in its original community. In the first stage of the algorithm, this process is iterated over all nodes. It is possible and common to consider each node multiple times as the community structure of the network shifts. It is also worth noting

that the results of this step are often dependent on the order in which the nodes are traversed.

In the second step of the algorithm, we build a new network. The nodes of this network represent the final communities generated by the first step of the algorithm. Each edge between two nodes is weighted equal to the sum of edge weights between the corresponding communities from the first step.

In the next pass of the algorithm, we apply the first step to the network resulting from the second step of the first pass. This way the algorithm iteratively builds a hierarchical community structure which is as "tall" as the number of passes completed.

As with Newman's method, this algorithm is particularly efficient because we can use expression (16) to compute ΔQ for different permutations of communities in step one. Moreover, step two allows us to radically reduce the number of nodes evaluated in step one during subsequent passes of the algorithm. Empirically, the algorithm has been observed to have $O(nlogn)$ complexity on a typical sparse network. [4]

COMPARING RESULTS ON ZACHARY'S KARATE CLUB NETWORK

Zachary's Karate Club network is a famous dataset which describes a small, socially-active karate club. At some point in the Club's history, a disagreement split the Club into two factions. The group and its eventual fissure can be modeled as a social network with two communities representing the disagreeing factions. Edges are drawn between individuals who "were observed to [consistently] interact outside the normal activities of the club". Zachary's Karate Club network has 34 nodes and 77 edges. [5] The task of identifying the two factions remains a popular problem in community detection.

Newman's Greedy Algorithm returns a community structure on the Karate dataset with $Q = 0.381$, where the network is divided into two communities of size 17. The communities are correctly described, except for a single individual. In comparison, the Louvain Method returned a community structure with a $Q = 0.42$ with only three passes through the algorithm. (Though the latter modularity is higher, it was not reported whether the algorithm determined the "true" community structure.) Because the network was relatively small, both runtimes were close to 0. [2, 4]

Generally, the Louvain method is preferable to Newman's method since the latter has complexity $0(n^2)$ whereas the former has only $O(nlogn)$. For this reason, there are few studies comparing the modularity maximization abilities of both methods on the same network. However, it should be noted that Clauset, Newman, and Moore eventually published a modification of Newman's original greedy algorithm which suggests specific computational

optimizations and data structures to reduce the complexity to $O(nlog^2 n)$. The method is fundamentally the same and recovers the same community structures but is significantly more efficient. [8]

CONCLUSION

Though modularity maximization is one of the most popular approaches to community detection, the methodology is not without its pitfalls. In particular, it is known to overlook particularly small communities - this is known as the resolution limit problem - and particularly large communities. Interested readers may find [6] and [7] helpful in understanding the cause and effect of the resolution limit in modularity maximization. Ultimately, however, modularity maximization remains a powerful mathematical approach through which scientists may recover latent community structures from social phenomena that can be modeled by graphs.

Bibliography

[1] M.E.J. Newman, *Networks: An Introduction*. New York, NY. Oxford University Press, Inc. 2010.

[2] M.E.J. Newman, "Fast algorithm for detecting community structure in networks". Physical Review E, vol. 69, Issue 6, id. 066133. 2003.

[3] M.E.J. Newman, "Modularity and Community Structure in Networks". Proceedings of the National Academy of Sciences of the United States of America, 103(23). 2006.

[4] Vincent D. Blondel *et al.*, "Fast unfolding of Communities in Large Networks". Journal of Statistical Mechanics: Theory and Experiment, Issue 10. 2008.

[5] Wayne W. Zachary, "An Information Flow Model for Conflict and Fission in Small Groups". Journal of Anthropological Research, Vol. 33, No. 4 (Winter, 1977), pp. 452-473. 2009.

[6] Mingming Chen *et al.*, "Community Detection via Maximization of Modularity and Its Variants". IEEE Trans. Computation Social System, vol. 1(1), 26-45. 2014.

[7] Santo Fortunato *et al*, "Resultion Limit in Community Detection". 2006. `http://www.pnas.org/content/104/1/36.full`

[8] Aaron Clauset *et al*, "Finding community structure in very large networks". Phys. Rev. E 70, 066111. 2004.