Notes on Modularity

Liangjie Hong

January 7, 2010

1 Modularity

The main idea behind Modularity is that the links within a community is higher than the expected links in that community. Thus, we can use a function named Q to denote the difference between real number of links and the expected links:

$$Q = (\text{number of edges within communities}) - (\text{expected number of such edges}) \tag{1}$$

Let us use A_{ij} to denote the links between node i and j where it is 1 meaning there exists a link from i to j. In addition, let us use P_{ij} to denote the expected number of links between i and j. We use g_i to indicate which community node i belongs to. Therefore, the Q function can be formulated as :

$$Q = \frac{1}{2m} \sum_i \sum_j (A_{ij} - P_{ij}) \delta(g_i, g_j) \tag{2}$$

where $\delta(x, y) = 1$ if $x = y$ and m is the total number of links in the graph. Note, the term $\frac{1}{2m}$ is totally conventional and it is not a requirement in the Modularity optimization since it is a constant for a fixed graph.

Now, the question becomes how to determine P_{ij}. Several criteria that Q needs to satisfy. First, if all nodes are in the same community, Q should be 0, which indicates that :

$$\sum_{ij} A_{ij} = \sum_{ij} P_{ij} = 2m \tag{3}$$

. Second, although under this assumption, there are still many null models to choose, we want the expected number of links for node i is exactly the same as the real number of links that node i has, meaning :

$$\sum_j P_{ij} = k_i \tag{4}$$

. Third, the probability that a link links to node i only depends on the total links node i has, k_i. We use $f(k_i)$ to denote this probability. Therefore, P_{ij} can be represented as :

$$P_{ij} = f(k_i) f(k_j) \tag{5}$$

. Combine the results from Equation 4 and Equation 5, we can obtain:

$$\sum_j P_{ij} = f(k_i) \sum_j f(k_j) = k_i \tag{6}$$

. Since we do not know the exact form of function f, we can not direct compute $\sum_j f(k_j)$. Let us just assume $f(k_i) = Ck_i$. Therefore, plug back into Equation 5, we can obtain:

$$P_{ij} = C^2 k_i k_j \tag{7}$$

. Furthermore, combined with Equation 3, we can get :

$$\sum_i \sum_j P_{ij} = \sum_i \sum_j C^2 k_i k_j = 2m \tag{8}$$

$$C^2 \sum_i k_i \sum_j k_j = 2m \tag{9}$$

$$C^2 4m^2 = 2m \tag{10}$$

$$C^2 = \frac{1}{2m} \tag{11}$$

where $\sum_i k_i = 2m$. Therefore,

$$P_{ij} = \frac{k_i k_j}{2m} \tag{12}$$

. So, even we do know know the exact form of function f, we obtained the expected links between node i and j as shown in Equation 12. Using this knowledge, Equation 2 becomes

$$Q = \frac{1}{2m} \sum_i \sum_j (A_{ij} - \frac{k_i k_j}{2m}) \delta(g_i, g_j) \tag{13}$$

. This is the definition of Modularity usually used in literature.

2 Leading Eigenvector Method

In the previous section, we have seen how Modularity is defined. First, we want to see some structures of Modularity definition. To simplify the notation, we first use $s_i = 1$ to denote whether node i belongs to community 1 and $s_i = -1$ to denote node i belongs to community 2. We are only dealing with two communities now. It is easily to show that

$$\delta(g_i, g_j) = \frac{1}{2}(s_i s_j + 1) \tag{14}$$

Plug it back into Equation 13, we can obtain:

$$\begin{aligned} Q &= \frac{1}{4m} \sum_i \sum_j (A_{ij} - \frac{k_i k_j}{2m})(s_i s_j + 1) \\ &= \frac{1}{4m} \sum_i \sum_j (A_{ij} - \frac{k_i k_j}{2m}) s_i s_j + (A_{ij} - \frac{k_i k_j}{2m}) \\ &= \frac{1}{4m} \sum_i \sum_j (A_{ij} - \frac{k_i k_j}{2m}) s_i s_j \end{aligned}$$

because of Equation 3. If we use B_{ij} to denote $A_{ij} - \frac{k_i k_j}{2m}$, we can obtain a much simpler form :

$$Q = \frac{1}{4m} s^T B s \tag{15}$$

Two important conclusion:

1. B is a symmetric matrix.
2. B has n normalized eigenvectors and they are orthogonal with each other.

Due to the second conclusion, we can represent s as a linear combination of n normalized eigenvectors as $s = \sum_i a_i v_i$ where v_i is the ith normalized eigenvector. The coefficients a_i can be obtained as follows:

$$\begin{aligned} s &= \sum_i a_i v_i \\ v_i^T s &= v_i^T \sum_j a_j v_j \\ v_i^T s &= \sum_j a_j v_i^T v_j \\ v_i^T s &= a_i \end{aligned}$$

where $v_i^T v_i = 1$ and $v_i^T v_j = 0$. Since $s^T s = n$, we can also verify that :

$$\begin{aligned} s &= \sum_i a_i v_i \\ s^T s &= s^T \sum_i a_i v_i \\ n &= \sum_i s^T a_i v_i \\ n &= \sum_i a_i s^T v_i \\ n &= \sum_i a_i^2 \end{aligned}$$

where $a_i = s^T v_i$. Note, a_i is a scalar. Therefore, Equation 15 becomes :

$$\begin{aligned} Q &= \frac{1}{4m} \sum_i a_i v_i^T B \sum_j a_j v_j \\ &= \frac{1}{4m} \sum_i a_i v_i^T \sum_j a_j \lambda_j v_j = \frac{1}{4m} \sum_i \sum_j a_i a_j \lambda_j \delta_{ij} \\ &= \frac{1}{4m} \sum_i a_i^2 \lambda_i \end{aligned}$$

where $v_i^T v_j = \delta_{ij}$ (Conclusion 2) and λ is the eigenvalue of matrix B.

From this formulation, we can see that to maximize Q is one of choosing the quantities a_i^2 so as to place as much as possible of the weight in the sum corresponding to the largest (most positive) eigenvalues. As with ordinary spectral partitioning, this would be a simple task if our choice of s were unconstrained (apart from normalization): we would just choose s proportional to the leading eigenvector v_1 of the modularity matrix. But the elements of s are restricted to the values $s_i = 1$, which means that s cannot normally be chosen parallel to v_1. Again as before, however, good approximate solutions can be obtained by choosing s to be as close to parallel with v_1 as possible, which is achieved by setting $s_i = +1$ if $u_i^{(i)} \geq 0$ and $s_i = -1$ otherwise. Therefore, we obtain a simplest algorithm for community detection: find the eigenvector corresponding to the most positive eigenvalue of the modularity matrix and divide the network into two groups according to the signs of the elements of this vector.

3 Other Eigenvectors

4 Original Definition

The original definition of Modularity, which is not the same as the Equation 13, is more straightforward and intuitive. It is like this :

$$Q = \sum_i (e_{ii} - a_i^2) \tag{16}$$

where e_{ij} is the fraction of edges in the community i to community j and $a_i = \sum_j e_{ij}$ that can be understood as the total degree of community i. Again, the rationale is the traction of edges that fall within communities, minus the expected value of the same quantity if edges fall at random without regard for the community structure. In fact, Equation 13 and 16 is "equivalent" (not exact the same). First, it is easily to see that $\delta(c_v, c_w) = \sum_i \delta(c_v, i)\delta(c_w, i)$. Therefore, from Equation 13:

$$\begin{aligned}
Q &= \frac{1}{2m}\sum_v\sum_w (A_{vm} - \frac{k_v k_w}{2m})\delta(c_v, c_w) \\
&= \frac{1}{2m}\sum_v\sum_w (A_{vm} - \frac{k_v k_w}{2m})\sum_i \delta(c_v, i)\delta(c_w, i) \\
&= \sum_i [\frac{1}{2m}\sum_v\sum_w A_{vw}\delta(c_v, i)\delta(c_w, i) - \frac{1}{2m}\sum_v k_v\delta(c_v, i)\frac{1}{2m}\sum_w k_w\delta(c_w, i)]
\end{aligned}$$

The first term $\frac{1}{2m}\sum_v\sum_w A_{vw}\delta(c_v, i)\delta(c_w, i)$ is essentially equal to e_{ij}, which is the fraction of edges that join vertices in community i to community j. The element of $\frac{1}{2m}\sum_v k_v\delta(c_v, i)$ in the second term is the fraction of edges that are attached to vertices in community i, which is indeed a_i. Thus, two formulation is equivalent. Note, in the early development of Modularity, the coefficent $\frac{1}{2m}$ is usually ignored. However, definition 13 is more accurate.

5 Greedy Algorithm

Most greedy approximation of Modularity is based on Equation 16. It is not totally surprising because the two parts of the equation only require *local* information. One way is to measure the difference of function Q if we join community i with community j. If the function Q increases, then we join them; otherwise, not. ΔQ can be computed as follows:

$$\begin{aligned}
\Delta Q &= [(e_{ii} + e_{ij} + e_{ji} + e_{jj}) - (a_i + a_j)^2] - [(e_{ii} - a_i^2) + (e_{jj} - a_j^2)] \\
&= e_{ij} + e_{ji} - 2a_i a_j
\end{aligned}$$

Note, here we are joining two communities. Another cheaper way is to join an isolated node i into a community j. In this case, ΔQ can be calculated as:

$$\begin{aligned}
\Delta Q &= [(e_{jj} + 2k_i^{\to j}) - (a_j + k_i)^2] - [(e_{jj} - a_j^2) - k_i^2] \\
&= 2k_i^{\to j} - 2a_j k_i
\end{aligned}$$

where $k_i^{\to j}$ represents the number of links that node i links to community j. Note, for an isolated node i, $e_{ii} = 0$. Remember, from Section 4, we can further know that the more accurate form is:

$$\begin{aligned}
\Delta Q &= \frac{2}{2m}(k_i^{\to j} - \frac{a_j k_i}{2m}) \\
&\propto k_i^{\to j} - \frac{a_j k_i}{2m}
\end{aligned}$$

The algorithm based on this equation can be applied to millions of nodes.

6 Directed Graph

Up to now, we mainly focus on **Undirected Graph**. Similar arguments can be applied to **Directed Graph** too. Our starting point is again Equation 2:

$$Q = \frac{1}{m}\sum_i\sum_j (A_{ij} - P_{ij})\delta(c_i, c_j) \tag{17}$$

One difference is that the normalizing coefficient becomes m because $\sum_{ij} A_{ij} = m$ for a directed graph. Again, $\sum_{ij} P_{ij} = m$ should be satisfied. Now, Equation 4 becomes :

$$\sum_j P_{ij} = k_i^{\text{out}} \tag{18}$$

where k_i^{out} is the out degree of k_i. Therefore, Equation 5 becomes:

$$P_{ij} = f(k_i^{\text{out}})f(k_j^{\text{in}}) \tag{19}$$

. Combining the above two, we get:

$$\sum_j P_{ij} = f(k_i^{\text{out}})\sum_j f(k_j^{\text{in}}) = k_i^{\text{out}}$$

We do not know function f. So, we assume $f(k_i^{\text{out}}) = C_1 k_i^{\text{out}}$ and $f(k_i^{\text{in}}) = C_2 k_i^{\text{in}}$. Thus,

$$P_{ij} = C_1 C_2 k_i^{\text{out}} k_j^{\text{in}} \rightarrow \sum_{ij} P_{ij} = m = C_1 C_2 m^2$$

So, it is obvious that $C_1 C_2 = \frac{1}{m}$. Therefore,

$$P_{ij} = \frac{k_i^{\text{out}} k_j^{\text{in}}}{m}$$

and we plug it back to Equation 17 obtaining:

$$Q = \frac{1}{m}\sum_i\sum_j (A_{ij} - \frac{k_i^{\text{out}} k_j^{\text{in}}}{m})\delta(c_i, c_j) \tag{20}$$

This is the "directed" version of Modularity. Now, similar as Section 4, we can obtain a simpler version of the definition:

$$\begin{aligned}
Q &= \frac{1}{m}\sum_v\sum_w (A_{vw} - \frac{k_v^{\text{out}} k_w^{\text{in}}}{m})\delta(c_v, c_w) \\
&= \frac{1}{m}\sum_v\sum_w (A_{vw} - \frac{k_v^{\text{out}} k_w^{\text{in}}}{m})\sum_i \delta(c_v, i)\delta(c_w, i) \\
&= \sum_i [\frac{1}{m}\sum_v\sum_w A_{vw}\delta(c_v, i)\delta(c_w, i) - \frac{1}{m}\sum_v k_v^{\text{out}}\delta(c_v, i)\frac{1}{m}\sum_w k_w^{\text{in}}\delta(c_w, i)]
\end{aligned}$$

Again, $\sum_v\sum_w A_{vw}\delta(c_v, i)\delta(c_w, i)$ is the number of links in community i, which can be viewed as e_{ii}. $\sum_v k_v^{\text{out}}\delta(c_v, i)$ is the out-degree of community i, defined as a_i^{out} and $\sum_w k_w^{\text{in}}\delta(c_w, i)$ is the in-degree of community i, defined as a_i^{in}. Therefore, the simpler version is :

$$Q = \sum_i (e_{ii} - a_i^{\text{out}} a_i^{\text{in}}) \tag{21}$$

Now, let us to see how Greedy algorithms become according to Equation 21.

$$\begin{aligned}\Delta Q &= [(e_{ii} + e_{ij} + e_{ji} + e_{jj}) - (a_i^{\text{out}} + a_j^{\text{out}})(a_i^{\text{in}} + a_j^{\text{in}})] - [(e_{ii} - a_i^{\text{out}} a_i^{\text{in}}) + (e_{jj} - a_j^{\text{out}} a_j^{\text{in}})] \\ &= e_{ij} + e_{ji} - a_i^{\text{out}} a_j^{\text{in}} - a_j^{\text{out}} a_i^{\text{in}}\end{aligned}$$

Similarly, if we want to join an isolated node i to community j, the algorithm becomes:

$$\begin{aligned}\Delta Q &= [(e_{jj} + k_i^{\to j} + k_i^{\leftarrow j}) - (k_i^{\text{out}} + a_j^{\text{out}})(k_i^{\text{in}} + a_j^{\text{in}})] - [(e_{jj} - a_j^{\text{out}} a_j^{\text{in}}) - k_i^{\text{out}} k_i^{\text{in}}] \\ &= k_i^{\to j} + k_i^{\leftarrow j} - k_i^{\text{out}} a_j^{\text{in}} - a_j^{\text{out}} k_i^{\text{in}}\end{aligned}$$

where $k_i^{\to j}$ is the number of links from node i to community j and $k_i^{\leftarrow j}$ is the number of links from community j to node i. If we incorporate normalizing coefficient:

$$\begin{aligned}\Delta Q &= \frac{k_i^{\to j} + k_i^{\leftarrow j}}{m} - \frac{(k_i^{\text{out}} a_j^{\text{in}} + a_j^{\text{out}} k_i^{\text{in}})}{m^2} \\ &\propto (k_i^{\to j} + k_i^{\leftarrow j}) - \frac{(k_i^{\text{out}} a_j^{\text{in}} + a_j^{\text{out}} k_i^{\text{in}})}{m}\end{aligned}$$