Weighted bipartite modularity

Stephen Beckett

October 15, 2014

Biosciences, College of Life and Environmental Sciences, University of Exeter, EX4 4QE

Note this document is currently a draft. It is not yet finished!

Introduction

networks important across sciences. Bipartite networks also widely used and interesting and now need methods for finding modules in weighted networks. different methods - simulated annealing... Here we focus on a fast method of finding modularity - label propogation.

Barber and Clark, 2009 introduced the bipartite modularity-specialising label propogation algorithm (LPAb) as a method to efficiently find high modularity scores - however prone to falling into suboptimal solutions. Indeed the landscape of modularity tends to be weird...flat smooth etc. Later Liu and Murata (2010) introduced a modification of the LPAb algorithm known as the LPAb+ algorithm which combines label propagation with a greedy multi-step agglomerator (Schuetz and Caflisch, 2008,?). Liu and Murata (2010) use a two step method that reduces the ability to fall into suboptimal solutions, whilst still retaining a fast algorithmic time complexity. This is done by using a top down...followed by a bottom up step. Additionally LPAb+ has performed well in comparison tests against other methods of finding bipartite modularity in binary networks.

(Dormann and Strauss, 2014) recently modified a simulated annealing algorithm to investigate bipartite modularity of weighted networks. Here I show how the LPAb+ algorithm can be modified for analysing modularity in weighted bipartite networks and test this algorithm against that of Dormann and Strauss (2014). Due to good performance in binary bipartite networks it is expected that LPAb+ may provide a good algorithmic procedure for finding modularity in weighted networks.

Methods

We first outline the LPAb+ algorithm(Liu and Murata, 2010) then show how it can be extended to weighted networks.

Barbers modularity

Modularity in binary unipartite networks is defined as(Newman, 2006):

$$Q = \frac{1}{2m} \sum_{u=1}^{n} \sum_{v=1}^{n} \left(A_{uv} - P_{uv}\right) \delta\left(l_u, l_v\right) \tag{1}$$

where A is the adjacency matrix of the network with n nodes, P is the probability matrix associated with the network describing the probability of a connection occuring between two nodes if the total number of edges between nodes (m) is constrained. Each element is calculated as $P_{uv} = \frac{k_u k_v}{m}$ where k is the node degree of the network and k_u is the number of nodes that connect to node u. The Kronecker delta δ is equal to one if two nodes are classified as being in the same module, hence having the same label l, or zero otherwise. If the network is bipartite, for example describing the visitations of r pollinator species to c different plant species ($n = r + c$), then there are no connections between the nodes of the same type e.g. pollinators don't pollinate other pollinators and plants cannot visit each other. As such A and P can be described in block diagonal form as:

$$\begin{cases} A = \begin{pmatrix} 0_{r\times r} & \tilde{A}_{r\times c} \\ \tilde{A}^T_{c\times r} & 0_{c\times c} \end{pmatrix} \\ P = \begin{pmatrix} 0_{r\times r} & \tilde{P}_{r\times c} \\ \tilde{P}^T_{c\times r} & 0_{c\times c} \end{pmatrix} \end{cases} \tag{2}$$

where $\tilde{A}$ and $\tilde{P}$ are the biadjacency matrices describing the connections between nodes of one type (pollinators) and nodes of the other type (plants) and the interaction probabilities between these nodes respctively; and T indicates the matrix transpose. This formulation allowed Barber to quantify bipartite modularity as:

$$Q_B = \frac{1}{m} \sum_{u=1}^{r} \sum_{v=1}^{c} \left(\tilde{A}_{uv} - \tilde{P}_{uv}\right) \delta\left(g_u, h_v\right) \tag{3}$$

where g is the labels for pollinator nodes and h are the labels for plant nodes. I will leave this equation in the form:

$$Q_B = \frac{1}{m} \sum_{u=1}^{r} \sum_{v=1}^{c} \left(\tilde{A}_{uv} - \frac{k_u d_v}{m}\right) \delta\left(g_u, h_v\right) \tag{4}$$

where k describes row degree (number of plant species each pollinator type is connected to) and d describes column degree (number of pollinator species each type of plant is visited by).

LPAb+

We first use the dimensions of the network to decide how to run the algorithm; this is because a bipartite community can have at most $F = min(r, c)$ communities. We then initialise the LPAb+ algorithm by giving a unique label to each of the nodes in the smallest of the two sets.

Stage 1 - label propagation stage -bottom up

Iterating the update of labels on the network is performed to locally maximise modularity. This means that for each node we essentially want to maximise the condition given by equation (4). For a particular plant node x this can be written as choosing a new label g_x by trying to maximise the condition:

$$g_x = \left(\sum_{v=1}^{c} \left(\tilde{A}_{xv} - \frac{k_x d_v}{m} \right) \right) \delta\left(g, h_v\right) = \left(\sum_{v=1}^{c} \tilde{A}_{xv} \delta\left(g, h_v\right) - \sum_{v=1}^{c} \left(\frac{k_x d_v}{m} \right) \delta\left(g, h_v\right) \right) \tag{5}$$

As the network is bipartite pollinator nodes only use information about plant nodes in order to update their labels (g) and similarly plant node labels (h) are updated only using information about the pollinator nodes an are thus mutually exclusive of one another. Simplifying equation (5) and creating an analogue for the updating rules for plant node labels leads to the following set of conditions:

$$\begin{cases} g_x^{new} = \arg\max\limits_{g} \left(N_{xg} - \frac{k_x D_g}{m} \right) \\ h_x^{new} = \arg\max\limits_{h} \left(N_{xh} - \frac{K_h d_x}{m} \right) \end{cases} \tag{6}$$

where the new label assigned to nodes x of type g (pollinators) or h (plants) is that which maximises g or h on the right-hand side (If the statement can be maximised in more than one way, one of these solutions is chosen at random). Here N_{xg} is the number of x's neighbours labelled g. UPDATE - min->max, max->...These updating rules are applied iteratively such that pollinator labels are updated, then plant nodes are updated, then pollinator nodes are updated and so on until equation (4) stops increasing.

Stage 2 - agglomeration stage -top down

When Barber's modularity can no longer be increased via stage 1's bottom up steps, a localised maximum of modularity for the network is reached, though this may not be the global maximum. The second stage seeks to prevent the algorithm getting stuck at a local maximum by merging communities using MSG (define). Specifically communities are formed from the unique labels in a

network, such that pollinator nodes and plant nodes are in the same community t when $g_u = h_v$. If there are F communities in total, then the merging of two communities t_i and t_j ($1 < i < j <= F$, $i \neq j$) can only occur if this would result in an increase in network modularity and if there is no third community t_k ($1 < k <= F$, $i \neq j \neq k$) whose merger with t_i or t_j would result in a larger increase to modularity.

Once the division of communities has been completed the algorithm is repeated, by going through stages 1 and 2 repeatedly until the networks communities can no longer be subdivided - which occurs when are no combination of two communities in the network that would result in increased network modularity if they were joined together. The communities in the network correspond to the modules found by the LPAb+ algorithm which has maximised bipartite modularity in the network based on the initial conditions - the way in which the initial set of nodes are labelled.

Weighted bipartite modularity

Dormann and Strauss (2014) follow Newmann 2004 and define weighted bipartite modularity similarly to before as:

$$Q = \frac{1}{2M} \sum_{u=1}^{n} \sum_{v=1}^{n} (W_{uv} - K_{uv}) \, \delta (l_u, l_v) \tag{7}$$

where W is the weighted adjacency matrix, M is the sum of all the edge weights in W and K is a probability matrix defined as $K_{uv} = \frac{T_u T_v}{M}$ with T_u being the marginal total of nodes connected to node u. As before this can be rewritten in biadjacency matrix form, simplifying to:

$$Q_W = \frac{1}{M} \sum_{u=1}^{r} \sum_{v=1}^{c} \left(\tilde{W}_{uv} - \tilde{K}_{uv} \right) \delta (g_u, h_v) = \frac{1}{M} \sum_{u=1}^{r} \sum_{v=1}^{c} \left(\tilde{W}_{uv} - \frac{y_u z_v}{M} \right) \delta (g_u, h_v) \tag{8}$$

where y is the row marginal totals and z is the column marginal totals of $\tilde{W}$. Q_W is the quantity that we wish to maximise with this algorithm for our observed networks. For a binary network W will become the adjacency matrix, the marginal totals will equal the node degrees and M will equal the fill - thus equation (8) will be reduced to equation (4) for a binary network. We can now use this definition of weighted bipartite modularity in the modified framework of Liu and Murata's LPAb+ algorithm.

LPAwb+

As before I follow Liu and Murata in defining two stages for the new algorithm. As the definition of modularity has altered it is now necessary to modify the node label updating rules in the first stage of the algorithm. The updating rules instead are defined as:

$$\begin{cases} g_x^{new} = \arg\max_g \left(\Omega_{xg} - \frac{y_x Z_g}{M} \right) \\ h_x^{new} = \arg\max_h \left(\Omega_{xh} - \frac{Y_h z_x}{M} \right) \end{cases} \quad (9)$$

where Z_g is the sum of the values in z (column marginal totals) whose plant nodes are labelled by g, whilst Y_h is the sum of the values in y (row marginal totals) for the pollinator nodes labelled h. Ω_{xg} is the sum of the weighted interactions between node x and it's neighbours labelled g. Note as before that equation (9) will reduce to equation (6) for a binary network.

The second stage of the LPAwb+ algorithm is as before, but hinges on the new definition of weighted modularity, such that communities are merged together only if it results in the best (positive) of the possible changes to the networks modularity.

Discussion

Null models...are the traditional thing...need a meaningfull null model...not an easy choice. But can evaluate the pattern in other ways.[strength of pattern] - this allows us to say something about members within different modules.

NP-hard problem in bipartite binary networks – hard! Assume it is also NP hard in weighted binary networks, as binary is just a special case of the formulation displayed here. However, depending on the edge distribution surfaces may be substantially less 'glassy' than in binary networks where the edge weight distribution is completely homogeneous.

Dormann's approach was novel - but we show here that it can be very slow and return far from optimal solutions when compared to the LPAwb+ algorithm. We chose to modify the LPAb+ algorithm to deal with weighted modularity due to its performance (both in time and accuracy) in binary bipartite networks. There are a large number of other approaches to maximising the modularity of a network however which have been applied to binary but not weighted networks; several of these could in principal be altered to work in weighted cases by using the modularity formulation of Dormann as shown here - and some of these may potentially perform better than either of these methods. However, we recommend using LPAwb+ in preference to the algorithm of Dormann due to improvements in speed and modularity scores obtained. As Dormann and Strauss highlight in their paper - it is not the precise measurement of modularity that is important, but rather the interpretation and explanation of modularity in the network and why certain nodes are placed into which groups with certain other nodes which is important in modularity analysis.

Acknowledgements

I thank Xin Liu for hints and nudges that helped me to setup my version of the LPAb+ algorithm. I acknowledge funding from the University of Exeter that

supported this research.

References

Barber, M. J., and Clark, J. W. (2009) Detecting network communities by propagating labels under constraints. *Physical Review E 80*, 026129.

Liu, X., and Murata, T. (2010) An Efficient Algorithm for Optimizing Bipartite Modularity in Bipartite Networks. *JACIII 14*, 408–415.

Schuetz, P., and Caflisch, A. (2008) Efficient modularity optimization by multistep greedy algorithm and vertex mover refinement. *Physical Review E 77*, 046112.

Schuetz, P., and Caflisch, A. (2008) Multistep greedy algorithm identifies community structure in real-world and computer-generated networks. *Physical Review E 78*, 026112.

Dormann, C. F., and Strauss, R. (2014) A method for detecting modules in quantitative bipartite networks. *Methods in Ecology and Evolution 5*, 90–98.

Newman, M. E. (2006) Modularity and community structure in networks. *Proceedings of the National Academy of Sciences 103*, 8577–8582.