

Statistical Physics of Community Detection

Keegan Go (keegango), Kenji Hata (khata)

December 8, 2015

1 Introduction

Community detection is a key problem in network science. Identifying communities, defined as densely connected groups of nodes with relatively fewer outgoing edges, can inform properties of nodes and give insight into the nature of the network. However, current methods of community detection often fail to reproduce groupings associated with recorded graph metadata on real-world networks. Although there are many different methods for community detection, ones that incorporate statistical physics are gaining traction. In this project, we focus on the implementation and insight of community detection methods that use statistical physics.

Traditional methods of community detection generally fall under the following areas: [1]

- Graph partitioning (min-cut), spectral clustering: These techniques try to partition of the graph in order to minimize the number of edges between different groups. Although relatively simple to implement, these problems are NP-hard and can currently only be used on large networks by relaxing the original problem.
- Hierarchical clustering: In these methods, one defines a similarity measure between node pairs and groups similar nodes into communities, by either agglomerative or divisive algorithms. Hierarchical clustering does not require any preliminary knowledge, but the results greatly depend on the similarity measure chosen. Moreover, it is computational expensive and often yields partitions with hierarchical structure, even if the graph itself does not contain any.
- Spectral clustering: These techniques take a pairwise similarity function and produce a symmetric, non-negative matrix, whose eigenvectors become a set of points that can be clustered. The major problem is that these methods cannot identify weak community structure and are sensitive to very low and high-degree vertices.
- Model fitting: After a statistical model is assumed, parameters and communities are found such that the likelihood of the assumed model is maximized. Although theoretically sound for networks that follow the assumed models, it is unclear how applicable these models may be for any arbitrary real-world network. Furthermore, solving these maximum likelihood problems may be computationally expensive or become stuck at local minima.

- Modularity maximization: In an attempt to maximize a modularity function, these methods search over all possible partitions of a network for one that has a high modularity. Since a brute-force search over every division is not possible, these algorithms often are greedy or iteratively optimize local communities until global modularity cannot be improved by these small adjustments.

2 Related Work

Zhang and Moore [2] describe a general problem with modularity based methods which is that the maximum modularity over all partitions of a ER random networks is often large even if the graph itself has no community structure. Their solution is essentially to score community structure by the number of partitions with large modularity. To this end they give a belief propagation (BP) algorithm to compute these scores. They show that their algorithm, under certain parameter settings, converges to a non-trivial partition, for SBM generated networks. On real-world networks, the authors find their method detects partitions with close similarity to ground truth (using overlap as a metric).

Saade et al. [3] focused on improving performance of spectral clustering on sparse networks by introducing the Bethe Hessian operator. They argue that traditional methods involving BP are not adequate, as the BP algorithm scales quadratically with the number of clusters and needs to be given an accurate estimate of the stochastic block model matrix. Furthermore, these methods do not handle sparse networks well, as they fail to identify any communities. However, spectral algorithms based on a non-backtracking walk of directed edges has been used to rectify these problems and has been proven to work well for stochastic block model generated graphs. The caveat is that this algorithm uses a high-dimensional, non-symmetric matrix linear in the number of edges, resulting in poor runtime efficiency. The authors instead use the Bethe Hessian operator, which aims to solve both problems, as it is a symmetric, real matrix that is not parameter-dependent.

3 Data Generation and Collection

A common method for generating community-structured networks is the stochastic block model (SBM). The SBM takes in three parameters:

- k : a scalar value that denotes the number of groups within the network
- \hat{z} : a $n \times 1$ vector where z_i indicates the group of the i -th node
- M : a $k \times k$ block matrix where M_{ij} gives the probability of a node in group i connected to a node in group j .

By adjusting these three parameters of the SBM, we can generate networks with different community properties. Thus, it gives us an adjustable method for testing the community detection algorithms.

Furthermore, we will test the algorithms on real-world networks as given in [4].

4 Description of Algorithms Used

4.1 Modularity Based Message Passing

The algorithm described in [2] is a form of belief propagation known as the cavity method in statistical physics. The method is relatively straightforward. At each iteration, messages are sent between all adjacent nodes.

Let us define a few symbols

- β : convergence parameter
- m : the number of edges
- d_i : the degree of the i -th node
- ∂i : the set of neighbors of node i
- $\psi_t^{i \rightarrow k}$: the message from node i to k of the belief that node k is in community t
- ψ_t^i : the marginal belief that node i is in community t
- θ_t : approximation of the long range effect (known as the field in statistical physics)

Formulas for the computed quantities are given below.

$$\theta_t = \sum d_i \psi_t^i$$
$$\psi_t^i \propto \exp \left[\frac{-\beta d_i \theta_t}{2m} + \sum_{j \in \partial i} \log(1 + \psi_t^{j \rightarrow i} (e^\beta - 1)) \right]$$
$$\psi_t^{i \rightarrow k} \propto \exp \left[\frac{-\beta d_i \theta_t}{2m} + \sum_{j \in \partial i \setminus k} \log(1 + \psi_t^{j \rightarrow i} (e^\beta - 1)) \right]$$

The first two are normalized to sum to 1 each time they are computed. The algorithm works by first computing θ_t , and then updating the marginals and the messages using this theta and the previous messages. This process is iterated until convergence.

4.2 Spectral Clustering using Bethe Hessian Operator

For a graph $G = (V, E)$ with n vertices in V and m edges in E , the Bethe Hessian operator is

$$H(r) := (r^2 - 1)\mathbf{1} - rA + D$$

where r is a regularization term, A is the adjacency matrix, and D is the diagonal matrix where D_{ii} is the degree of node i . Saade et. al found that clustering the eigenvectors of $H(r_c)$ and $H(-r_c)$, where r_c is the square root of the average degree of the graph, is an optimal choice in deciding community structure.

4.3 rNMI

A traditional metric for measuring partition accuracy given some true partition P_A and another partition P_B is the normalized mutual information (NMI):

$$\text{NMI}(P_A, P_B) = \frac{2I(P_A, P_B)}{H(P_A) + H(P_B)}$$

where $I(P_A, P_B)$ is the mutual information between the two partitions P_A and P_B and $H(P)$ is the entropy of the partition P . However, Zhang [5] argues that NMI prefers a larger number of partition, which skews its reliability. To account for this preference, Zhang proposes an adjusted accuracy metric called the relative normalized mutual information (rNMI):

$$\text{rNMI}(P_A, P_B) = \text{NMI}(P_A, P_B) - \langle \text{NMI}(P_A, P_C) \rangle$$

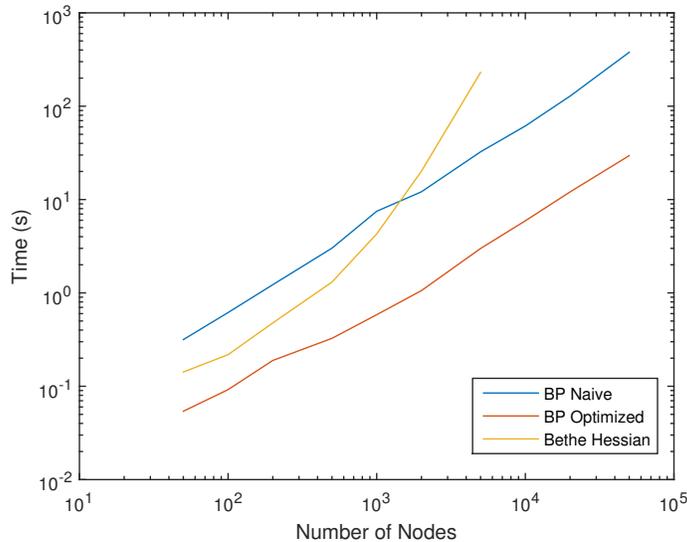
where $\langle \text{NMI}(P_A, P_C) \rangle$ is the expected NMI between the ground-true configuration P_A and a random partition P_C .

5 Results

5.1 Implementation and timing tests

We implemented modularity based message passing and the Bethe Hessian spectral clustering algorithm. The precise algorithms used in both cases are given in appendix A.

For the message passing method, we provide two implementations: a pedantic version following the description of the algorithm given by Zhang and Moore, and an optimized version. We timed the methods on different networks (excluding loading times) with the results shown below.



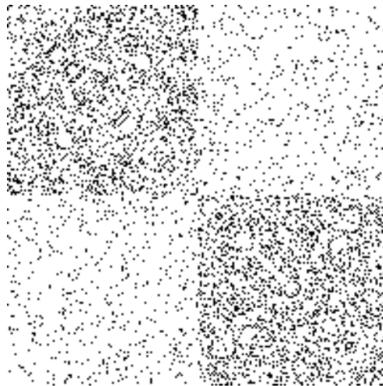
Our optimized implementation is drastically faster (nearly a factor of 10 over the naive implementation) and scales nicely over a large range of network sizes. It is still a little slower than the authors C implementation but this seems acceptable given that we are working with Python’s memory management.

Additionally, we implemented message passing using the GraphLab’s `sframe` library [6]. This library aims to provide access machine learning methods that scales to massive data. While we were excited about using this library to apply message passing to large networks, it turns out that GraphLab’s python generic mapping functions (needed to implement custom graph methods) works very slowly in practice. We did not test their C++ interface.

The Bethe Hessian spectral clustering is functional up to about 10000 nodes. At this point, computing the eigenvalues and eigenvectors of the Bethe Hessian in Python takes a very long time. We noticed that in the original paper, the authors only tested the algorithm on networks up to 1000 nodes. It is clear that Bethe Hessian spectral clustering may not be a sufficient method for fast community detection on extremely large networks.

5.2 Validation

To ensure correctness we validated our data on both synthetic and real-world networks. We first implemented SBM. For message passing, as in [2], we found that choosing $\beta = 1$ worked for most of the models. Below is an example of a SBM network we used and successfully detected communities in with modularity based message passing.



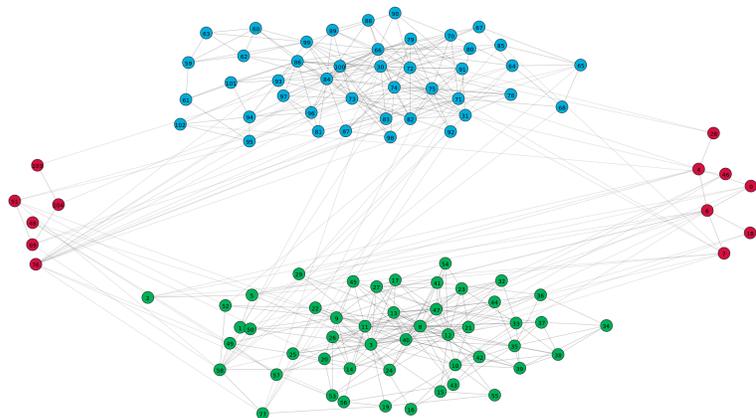
Similar results were found using Bethe Hessian spectral clustering.

5.3 Real world networks

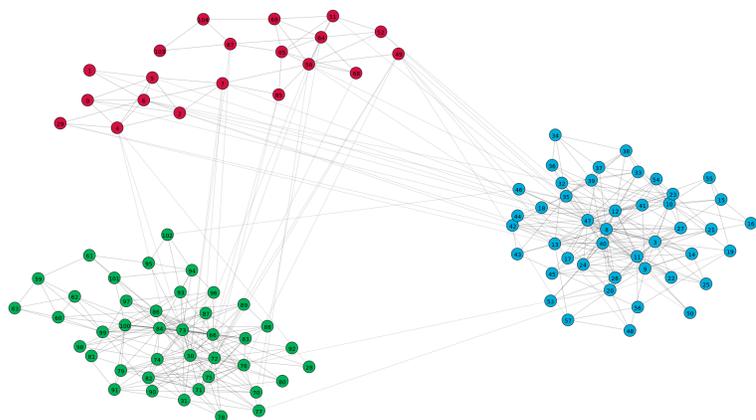
We ran our implementations on karate, football, dolphins, polbooks, and polblogs. We highlight the results for polbooks below to help understand the results of these methods.

For the polbooks dataset, we show (a) the real partition (b) partition achieved by message passing (c) partition achieved by Bethe Hessian spectral clustering.

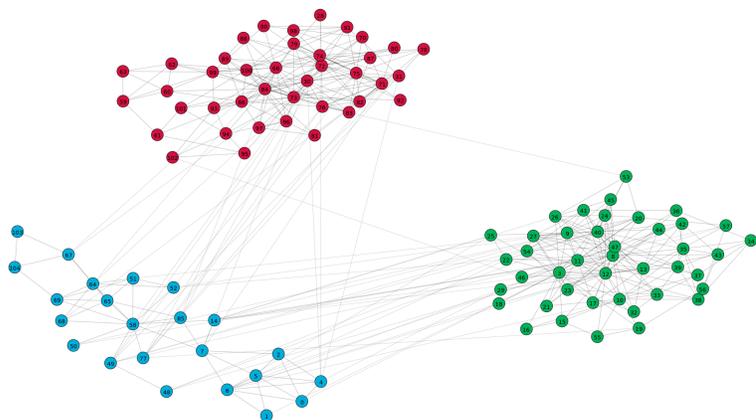
(a) Real partition



(b) Partition achieved by message passing

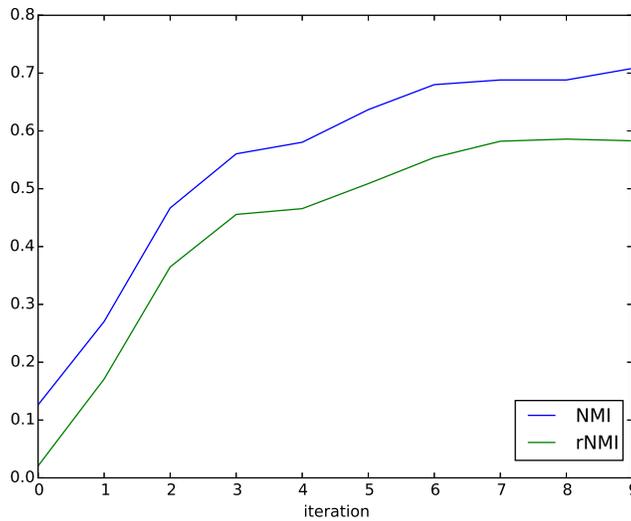


(c) Partition achieved by Bethe-Hessian spectral clustering



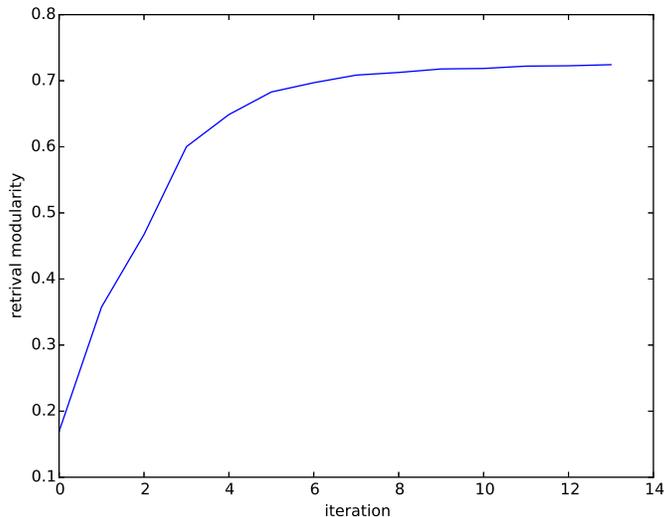
Overall, both the message passing algorithm and Bethe Hessian spectral clustering arrive at a partition different than that of the real partition. However, we notice that, in the real-partition, the neutral, red cluster has interesting topological features, as two halves of the cluster are disconnected. These topological properties make it difficult for both the message passing and Bethe Hessian methods to detect the true partition. Furthermore, it seems to suggest the findings made by Hric et al. [7] that only using topological features may be insufficient for community detection.

We also noticed the difference between using NMI and rNMI from these real data sets. The plot below shows these values over iterations of the message propagation.



Note that in this plot, the value of NMI consistently overestimates how well the current partitions matches with the ground truth of the network.

To confirm that our method scaled correctly, we ran it on the web-google dataset. This data set has approximately 900000 nodes and 5 million edges. The plot below shows the retrieval modularity by iteration.



This matches the results in [2] in approximately the same running time for the same number of clusters. However, the authors used a factor of 10 more iterations, which we found to be unnecessary by tuning β .

6 Conclusion

We have implemented a fast, scalable python version of the message passing algorithm described in [2], as well as the Bethe Hessian spectral clustering algorithm. From testing on a variety of both synthetic and real world networks, we validate that our algorithms give similar results to those described in the corresponding papers, and we were able to reproduce results showing the difference between different metrics of similarity between partitions.

In continuing work, we would like to extend the work to networks with more clusters. Though not mentioned in the original work, the authors implicitly obtained their results on graphs with small numbers of partitions (2-10). As the message passing algorithm scales with the product of the number of edges and the number of clusters, this method rapidly becomes infeasible for networks with possibly many communities in terms of both memory and compute time. Handling this would probably require scaling up to a large distributed system.

References

- [1] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.
- [2] Pan Zhang and Cristopher Moore. Scalable detection of statistically significant communities and hierarchies, using message passing for modularity. *Proceedings of the National Academy of Sciences*, 111(51):18144–18149, 2014.

- [3] Alaa Saade, Florent Krzakala, and Lenka Zdeborová. Spectral clustering of graphs with the bethe hessian. In *Advances in Neural Information Processing Systems*, pages 406–414, 2014.
- [4] Online social networks research - imc 2007 data sets. <http://socialnetworks.mpi-sws.org/data-imc2007.html>. Accessed: 2015-10-15.
- [5] Pan Zhang. Evaluating accuracy of community detection using the relative normalized mutual information. [abs/1501.03844](https://arxiv.org/abs/1501.03844), 2015.
- [6] Yucheng Low, Joseph E Gonzalez, Aapo Kyrola, Danny Bickson, Carlos E Guestrin, and Joseph Hellerstein. Graphlab: A new framework for parallel machine learning. *arXiv preprint arXiv:1408.2041*, 2014.
- [7] Richard K. Darst Darko Hric and Santo Fortunato. Community detection in networks: Structural communities versus ground truth. *Physical Review*, 90(6), 2014.

7 Appendix A

This sections gives the algorithms we implemented. Our implementation can be found at <https://bitbucket.org/kgo/cs224w-project>.

Modularity based message passing

```

Given arguments graph  $G$ , scalar  $\beta$ , and number of clusters  $T$ 
for each node in  $G$  do
    Randomly initialize node.marginals ( $T$ -vector)
    Randomly initialize node.messages ( $T$ -vector for each neighbor)
end for
for iterations do
    // Compute theta
     $\theta \leftarrow 0_T$ 
    for each node in  $G$  do
         $\theta += \text{node.degree} * \text{node.marginals}$ 
    end for
    // Perform common modification of messages
    for each node in  $G$  do
        for each neighbor of node do
            // element-wise application of log and exp
             $\text{node.message}[\text{neighbor}] \leftarrow \log(1 + \exp(\beta - 1)(\text{node.message}[\text{neighbor}]))$ 
        end for
    end for
    // Compute marginals and messages
    for each node in  $G$  do
        total  $\leftarrow$  sum of messages for node from all neighbors
         $\text{node.marginals} \leftarrow \exp(\frac{-\beta(\text{node.degree})}{2M}\theta + \text{total})$ 
        for each neighbor of node do
            m  $\leftarrow$  message for node from neighbor
             $\text{node.message}[\text{neighbor}] \leftarrow \exp(\frac{-\beta(\text{node.degree})}{2M}\theta + \text{total} - m)$ 
        end for
    end for

```

```

    end for
  end for
  // Normalize probabilities
  for each node in  $G$  do
    normalize node.marginals
    for each neighbor of node do normalize node.message[neighbor]
    end for
  end for
end for

```

Bethe Hessian spectral clustering

Given arguments adjacency matrix A
 Let r be the square root of average degree
 Let D be diagonal matrix where D_{ii} is degree of each node i
 Let $H_p = (r^2 - 1)\mathbf{1} - rA + D$
 Let $H_m = (r^2 - 1)\mathbf{1} + rA + D$
 Let q be a predefined number or the number of negative eigenvalues in both H_p and H_m
 Let v be the list of eigenvectors of H_p and H_m .
 Normalize v by the l-2 norm.
 Find q clusters o v using k-means