Hierarchical Community Detection in Complex Networks using Growing Hierarchical Self-Organizing Maps

David McDonald
University of Birmingham,
Birmingham, UK
dxm237@cs.bham.ac.uk

Shan He
University of Birmingham,
Birmingham, UK
s.he@cs.bham.ac.uk

Abstract

Detecting the community structure of complex networks is a challenging task that has occupied researchers of many disciplines for decades. Communities often emerge of different scales and can contain within them smaller sub-communities. We show that the so-called Growing Hierarchical Self-Organizing Map can detect the communities at multiple scales that comprise complex networks and can preserve the topological information contained within them. Results on small real-world benchmark networks, as well as synthetic networks, suggest that this approach can find the underlying community structure of a network, even when the mixing factor between communities is high.

1 Introduction

Most real-world networks contain subsets of nodes that contain a higher degree of inter-connectivity than the rest of the network [Lancichinetti *et al.*, 2009]. These subsets are commonly referred to as communities. A rigorous definition for a 'community' within a network still seems to elude the scientific community [Lancichinetti *et al.*, 2009]. However, the most popular definition among scholars is the planted l-partition model. This was popularised thanks to Girvan and Newman in their seminal work [Girvan and Newman, 2002] and states that as long the probability of a node being connected to its group is greater than the probability of it being connected to the rest of the graph, then those groups are communities.

'Community detection' is the name given to the problem of finding the underlying community structure in a given network [Girvan and Newman, 2002]. Since many complex problems can be represented as a network, community detection has proven to be useful in seemingly disconnected areas of study, such as social [Barber, 2007], biological [Fortunato and Barthelemy, 2007], and world-wide-web [Danon *et al.*, 2006] analysis.

Most real world networks do not solely contain communities at one scale [Lancichinetti *et al.*, 2009; Yang *et al.*, 2013]. They contain super communities that may contain sub-communities, that may, in turn, contain their own sub communities. And so on. Delving deeper into the community structure may offer some insight into how these processes work. For example, hierarchical analysis of protein-protein interaction networks may help to identify subsets of functionally-related proteins that interact together strongly within an established community representing some biological process.

But, a general community detection algorithm does not yet exist. Many existing algorithms suffer from a number of issues. To name a few: The number and scale of communities must be known a-priori, which in most real applications, is infeasible. Additionally, the relationships between communities, both one the same level and at different ones, is lost. Identifying not only the community itself, but its position in the network as a whole, provides further insight into the often abstract interactions that comprise complex networks and so preserving this information when analysing a network is paramount. And, in some cases, the algorithms cannot deal with special cases: for example, modularity-based methods suffer from the so-called 'resolution limit' [Fortunato and Barthelemy, 2007].

The Growing Hierarchical Self-Organizing Map (GHSOM), which was first described by Dittenbach et al. [Dittenbach *et al.*, 2000], offers a unique solution to these problems . It uses the topology-preserving property of self-organizing maps, in conjunction with the ability for maps to grow into arbitrary shape completely autonomously. GHSOM is capable of detecting the community structure of complex networks and adapt the shape of its maps to fit the structure of the data. And, by selecting promising areas of the input space to expand and produce maps of finer granularity, GHSOM is capable of studying interesting areas of the input space in greater detail. These properties suggest that GHSOM is a suitable candidate for a hierarchical community detection algorithm, and this study will explore the usefulness of GHSOM for this task.

2 Related Work

Over the years, numerous and varied approaches have arisen to tackle the problem of community detection. Readers interested in thorough reviews and in-depth algorithm comparisons are directed to [Fortunato, 2010].

2.1 Cut-Based and Spectral Approaches

Most early work on community detection focused on 'cutting' the network into modules, in such a way that the number of edges cut was minimized. In theory, this resulted in the best partition of the network into communities [Kernighan and Lin, 1970]. However, this often favoured cuts of small, peripheral subgraphs, so it was adapted into ratio cut, normalized cut and min-max cut that took the number of nodes in each resulting sub-graph into account, and thus resulted in a partition that was more balanced. Summaries of these cuts can be found in [Fortunato, 2010].

Spectral clustering dates back to the work of Donath and Hoffman in 1973 [Donath and Hoffman, 1973]. However, it was popularized in the early 2000s [Ng *et al.*, 2002]. Spectral methods rely upon constructing 'Laplacian' matrices from the raw network data and eigen-decomposing them. Clustering the resulting eigenvectors results in clusters of the original data points. Spectral approaches have many advantages over other techniques and, as a result, they have become popular in the machine learning community for clustering on non-linear manifolds. Spectral methods can also be combined with traditional hierarchical modularity methods [Donetti and Munoz, 2004].

Further work includes the Markov Clustering algorithm (MCL) that simulates a diffusion process on a graph by repeatedly performing stages of expansion and inflation and only keeping the k largest elements for efficiency [Van Dongen, 2001].

2.2 Modularity-Based Approaches

The seminal work of Girvan and Newman [Girvan and Newman, 2002] marked a significant advance in the field by providing the first quantitative measure of a community: modularity. The modularity of a partition of a network scores a network partition by comparing the number of links inside a given module with the expected number that would be found in a random graph of the same size and degree sequence. Girvan and Newman propose a hierarchical divisive algorithm that removes edges based on their 'betweenness' (the number of shortest paths from two nodes in the network that go through them) until the modularity quality function is maximized. The early work of Girvan and Newman has since been expanded upon. For example, edge clustering in favour of edge-betweenness [Radicchi *et al.*, 2004], iteratively adding links to a module based on their expected increase in modularity [Clauset *et al.*, 2004], and multi-stage local optimization [Blondel *et al.*, 2008].

2.3 Statistical Approaches

Statistical approaches have been shown to deduce the best model to fit the data represented in the graph structure, which may not necessarily be communities [Newman and Girvan, 2004]. They have also been used to show the trade off between using heuristics to reduce the search space and optimization of a constrained quadratic function [Yang *et al.*, 2013].

2.4 Neural Network Approaches

The deep learning community has began to explore the possibilities of using neural networks for clustering in the graph domain. Convolutional neural networks (CNNs), powerful machine learning tools that have proven very successful for challenging classification tasks that have recently been generalised to take a graph input [Defferrard *et al.*, 2016]. CNNs have also been used for semi-supervised learning on graphs, where the they are capable of learning both graph structure and node features [Kipf and Welling, 2016].

3 Hierarchical Community Detection

The hierarchical nature of modularity-based clustering methods can allow them to detect communities at different scales. [Lancichinetti *et al.*, 2009] used local optimization to maximize a fitness function with a parameter that controlled the size of communities detected. Other work includes multi-scale quality functions that can uncover hierarchical communities and produce several different partitions of a graph, the post-processing of clusters found by hierarchical methods (encoded in a dendogram) [Pons and Latapy, 2011], and Bayesian non-negative matrix factorisation that performs 'soft-partitioning' and assigns node participation scores to modules [Psorakis *et al.*, 2011].

4 Self-Organizing Maps

Also known as the Kohonen Map after its creator Teuvo Kohonen, the Self-Organising Map (or SOM) is a topology-preserving neural network that performs unsupervised learning of a input space onto a low-dimensional discrete map [Kohonen, 1990]. Unlike other neural networks, SOM uses competitive learning - determining the 'winning' neuron by some distance function – called Learning Vector Quantization (LVQ), rather than a supervised training approach. Weights of all neurons are adjusted proportional to their distance to the winning neuron. After many epochs of organization in this way, the resulting map perseveres the topology of the input space, meaning that features that appear together in the input space will appear together in the map.

4.1 SOM in the Literature

Explicit community detection on graphs using SOM remains largely under-researched. But, by constraining neuron weights to move along edges in the graph, SOM has shown promise for use on raw graphs, at least for the simple case of the travelling salesman problem [Yamakawa *et al.*, 2006]. SOM has been shown to perform an implicit form of multi-dimensional scaling (MDS) for graph embedding and visualization in the two and three dimensional plane [Bonabeau, 2002].

For clustering in general with SOM, unsupervised and supervised algorithms have been proposed that control map growth and neuron removal [Fritzke, 1994]. Additionally, clustering the output of SOM with k-means has been shown to outperform k-means and hierarchical methods alone [Kiang, 2001].

5 Method

The entire algorithm presented in this work consists of two main phases: projection and map construction. Projection is concerned with embedding the vertices of the network into k-dimensional Euclidean space. GHSOM is then used to construct two-dimensional maps of the community structure of the network at different hierarchies.

5.1 MDS Projection of Vertices

Multi-Dimensional Scaling was used to first project the vertices of a given network $\mathbf{G}$ onto a Euclidean plane $\mathbf{X} \subset \mathbb{R}^k$. Explicit MDS was used in favour of running GHSOM on the raw network as MDS embeds based on a similarity measure than can be changed. The embedding for one measure – day shortest path distance – may result in different communities than for another measure – say sum of node degree along shortest path. And, the communities that are preserved across multiple measures may be important to the network as a whole.

MDS takes as input a matrix $\mathbf{D}$ of distances or dissimilarities and attempts to find an embedding of each point such that the relative distances are preserved. Here, we use the shortest path between two nodes as the metric, as in [Yamakawa *et al.*, 2006]. This distance is well defined and finite for all pairs of nodes in the network since $\mathbf{G}$ is a connected graph.

We then use double centring to construct a kernel matrix $\mathbf{K}$ from $\mathbf{D}$ and eigen-decompose it. In order to prevent the need to specify the dimensionality of the data, k, a-priori, we determine k dynamically as follows:

$$k = \underset{k}{\operatorname{argmax}} \left(\frac{\Sigma_{i=1}^{k} \lambda_i}{\Sigma_{j=1}^{n} \lambda_j} \geq 0.95 \right) \tag{1}$$

This ensures that 95% of variation is preserved, no matter how many nodes in $\mathbf{G}$ there are. Then the k largest eigenvalues and their associated eigenvectors are used to compute $\mathbf{X}$, giving an $n \times k$ matrix where the ith row gives the coordinates of node i in k-dimensional space.

5.2 The Growing Hierarchical Self-Organizing Map

Embedding the n nodes into k dimensional space translates the problem of finding communities in a graph to identifying clusters in the Euclidean Space $\mathbf{X}$. Since the distance (by whatever metric used) between two nodes within the same community should be smaller than two nodes belonging to two different communities, then all of the nodes within a community should be embedded closely together in $\mathbf{X}$. For clustering in $\mathbf{X}$, we use GHSOM.

5.3 Hyper-Parameters

GHSOM takes two additional hyper parameters to SOM. The first is ϵ_{sg} and this determines the size of the maps that will be created. The second, ϵ_{en}, determines the granularity of the maps produced, or when the network should grow a new layer. Both of these will be discussed in more detail later in this paper. Additionally, GHSOM uses a measure of error, that its creators, Dittenbach et al. called Mean Quantization Error (or MQE). Following their example, let the MQE of a single neuron i be denoted $\mathbf{mqe}_i$ and the MQE of an entire map m be denoted $\mathbf{MQE}_m$.

5.4 MQE$_0$

GHSOM begins with layer 0 comprising a single neuron map. This neuron is given a weight vector $\mathbf{w}$ that points to the centre (the mean) of all the data points:

$$\mathbf{w} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i \tag{2}$$

The MQE of this neuron (and therefore this first map) is the variance of the data set:

$$\mathbf{MQE}_0 = \mathbf{mqe}_0 = \frac{1}{n} \sum_{i=1}^{n} ||\mathbf{w} - \mathbf{x}_i|| \tag{3}$$

This is used as a benchmark for all the following layers to beat.

5.5 Training the First Layer

GHSOM then begins by training the first layer map. The first layer is initialized to just one neuron. The weights of this neuron is initially set to be a random vector in $\mathbb{R}^k$. The map is then trained as in the standard SOM algorithm, found in [Kohonen, 1990] for $\lambda = 10000$ epochs.

One training epoch consists of presenting each training pattern (the embedded nodes of $\mathbf{G}$) to the map. For each pattern $\mathbf{x}$, the 'winning' neuron i^* is determined as:

$$i^* = \underset{i}{\operatorname{argmin}} ||\mathbf{x} - \mathbf{w}_i|| \tag{4}$$

The weights of every neuron in the map are then adjusted to move a little closer to $\mathbf{x}$, based on how close to i^* they are in the map.

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta(t) h(i, i^*, \sigma(t))(\mathbf{v} - \mathbf{w}_i) \tag{5}$$

The weights are adjusted proportionally to the learning rate $\eta(t)$, and the neighbourhood function $h(i, i^*, \sigma(t))$, given by

$$h(i, i^*, \sigma(t)) = \exp\left(- \frac{||\mathbf{r}_i - \mathbf{r}_{i^*}||^2}{2\sigma(t)^2} \right) \tag{6}$$

where $\mathbf{r}_i$ is the position of neuron i in the map. The neighbourhood function preserves the topological ordering of the map, as neurons closer to the winning neuron in the map move towards $\mathbf{x}$ more. At the end of each training epoch, the order of the training patterns is shuffled. Then the neighbourhood width $\sigma(t)$ is adjusted in the following manner:

$$\sigma(t) = \sigma_0 \exp\left(- \frac{2\sigma_0 t}{\lambda} \right) \tag{7}$$

5.6 Growing the Map

After $\lambda = 10000$ training epochs have passed, every node in $\mathbf{G}$ is assigned to the neuron in the map whose weight vector $\mathbf{w}$ points closest to it in $\mathbb{R}^k$. Then the $\mathbf{mqe}_i$ for each neuron is calculated, as well as the overall $\mathbf{MQE}_1$ of the map. If

$$\mathbf{MQE}_1 \geq \epsilon_{sg} \mathbf{mqe}_0 \tag{8}$$

then the network is expanded, by inserting a single new neuron into the map, else expansion terminates.

Network expansion largely follows the example of Bernd Fritzke in [Fritzke, 1994] and aims to preserve a simplicial structure within the map whenever a new neuron is inserted. This ensures that neurons do not have too many neighbours. Firstly, one must identify the 'error unit' e – the neuron with the greatest error in the map. Then retrieve the list of all of the adjacent neurons (the neighbours) of e in the map.

In the general case, the 'furthest neighbour', n_1, of e is identified. This is the neighbour whose weight vector points furthest away from the weight vector of e in the input space. The furthest neighbour is then determined from the list of mutual neighbours of n_1 and e, called n_2. Next, the connection between n_1 and n_2 is broken and a new neuron is inserted with a weight vector equal to the mean of the weight vectors of e, n_1, and n_2. Finally, the new neuron is connected to every neuron that is connected to both n_1 and n_2 (including e itself). When there are e has no neighbours in the map, then a new neuron is connected to e with a random weight vector. If e has one neighbour, then the new neuron is connected to both neurons in the map and given a weight equal to the mean of their weights.

5.7 Termination of Map Growth

The map is again trained for $\lambda = 10000$ epochs and $\mathbf{MQE}_1$ is recalculated. If

$$\mathbf{MQE}_1 < \epsilon_{sg}\mathbf{mqe}_0 \tag{9}$$

then map growth terminates. Otherwise, another neuron is inserted and the training and growth process repeats.

5.8 Adding a New Layer to the Map

After growth for the current map has terminated, GHSOM determines whether or not to build another layer by expanding specific neurons in this one. Whether or not to expand is determined by ϵ_{en}. GHSOM scans across all the neurons i in the map and checks for

$$\mathbf{mqe}_i \geq \epsilon_{en}\mathbf{mqe}_0 \tag{10}$$

If a neuron i^* is located with this property then a new map is constructed (again starting from one neuron). This time, only the subset of nodes in $\mathbf{G}$ that were assigned to i^* are passed to the new map. Training growth and expansion of new map follows exactly the same procedure as for the current map.

5.9 Termination of GHSOM

GHSOM continues to train, grow and expand maps until

$$\mathbf{MQE}_m < \epsilon_{sg}\mathbf{MQE}_0 \tag{11}$$

for all maps m and

$$\mathbf{mqe}_i < \epsilon_{en}\mathbf{MQE}_0 \tag{12}$$

for all neurons i. The output is then a set of maps where each neuron represents a community in $\mathbf{G}$. Neurons that were expanded contain pointers to the map that was grown from their nodes.

Network	Description
Karate	Social network of karate club [Zachary, 1977].
Dolphin	Social network of dolphins living in New Zealand [Lusseau *et al.*, 2003].
Polbooks	Network of books sold by `www.amazon.com` and published around the 2004 presidential election [Girvan and Newman, 2002].
Football	A network of college football games in Fall 2000 [Girvan and Newman, 2002].

Table 1: Description of four real-world networks.

Parameter	Description	Network Type	
		Real-World	Synthetic
η	Learning rate	0.0001	0.0001
w	Weight range	0.0001	0.0001
σ	Neighbourhood size	1	1
ϵ_{sg}	Stop map growth	0.8	0.8
ϵ_{en}	Grow new map	0.8	0.8

Table 2: Parameter setting for both types of network.

6 Results

Results were obtained using two sets of parameters for GHSOM, one for the real-world networks and one for the synthetic ones. Parameters were kept consistent across all networks to allow for fair comparison, but individual optimization of these parameters produced better results. Table 2 details the consistent parameter settings used for each network type.

Four representative algorithms from the literature are used for comparison: MCL [Van Dongen, 2001], FM [Clauset *et al.*, 2004], FUC [Blondel *et al.*, 2008], and PMC [Yang *et al.*, 2013].

6.1 Dependence upon Parameter Settings

After some experimentation, it became apparent that the performance of the algorithm depended greatly upon parameter settings. In particular ϵ_{sg}, which controlled final map size. One neuron in the map corresponds to one community in the network, and so the overall map size is equal to the number of communities in the network. Because of this, a network with many communities would require a lower setting for ϵ_{sg} than networks with few communities.

In support of this hypothesis, an experiment was devised to measure the impact of community size on ϵ_{sg}. Synthetic networks were generated with $c \in [3, 6]$ communities. ϵ_{sg} varied varied over the interval $[0.5, 1.0]$ and the setting with the best mean NMI score over 10 repeats was recorded. Figure 1 shows the results of this experiment.

6.2 Synthetic Hierarchical Benchmarks

In ordert to investigate the strength of GHSOM at uncovering the hierarchical community structure of complex networks, synthetic benchmark networks were generated using software

Algorithm	**Network**			
	Karate	Dolphin	Polbooks	Football
MCL	**1.000**	0.424	0.515	**0.935**
FM	0.693	0.509	0.531	0.757
FUC	0.587	**0.636**	**0.575**	0.855
PMC	0.837	0.620	0.574	0.887
GHSOM	0.733	0.575	0.547	0.528
SE	1.48e-3	2.94e-4	2.09e-4	1.56e-4
#comms	2	4	3	12
#comms det.	2	2	2	3

Table 3: Table of NMI scores of GHSOM versus several algorithms in the literature. The best NMI score for each network is written in bold. Results for algorithms are taken from [Yang *et al.*, 2013] (table 2) without permission. Hyperparameters are kept consistent across all networks, given in table 2.

ϵ_{sg}	**#communities**			
	3	4	5	6
0.5	0.567	0.900	0.978	**0.987**
0.6	0.601	0.965	**0.981**	0.952
0.7	0.746	**0.977**	0.912	0.931
0.8	0.763	0.974	0.947	0.953
0.9	**0.889**	0.843	0.856	0.827
1.0	0.209	0.119	0.208	0.335

Figure 1: Community size against ϵ_{sg}. Communities contained 32 nodes and each node has 16 connections to nodes within its own community, and 16 to nodes outside. NMI scores in bold show the best NMI score for the number of communities. Each setting was repeated 10 times.

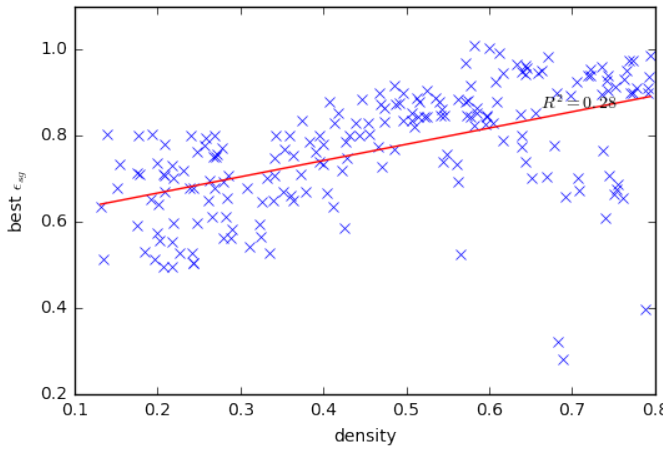


Figure 2: Plot of network density vs. best setting for ϵ_{sg}. 200 random networks of 64 nodes were generated with a random number of edges and communities.

provided by Lancichinetti et al. [Lancichinetti *et al.*, 2009]. The generated networks were comprised of 512 nodes divided into 16 communities of 32 nodes. These 16 communities formed 4 super-communities of 128 nodes. This produced networks of nodes belonging to communities at two levels: one at the micro level and one at the macro level.

The algorithm allows for the adjustment of three mixing parameters: z_1, z_2, and z_3, which control the number of connections between nodes of the same micro community, same macro community and other communities in the network respectively. As in [Lancichinetti *et al.*, 2009] and [Yang *et al.*, 2013], we keep z_1 and z_2 fixed at 16, and vary z_3, the number of connections between nodes of different macro communities, from 16 to 36, and plot NMI score for both levels of community. For $z_3 > 32$ then the external degree of nodes (the number of links outside of their micro/macro community) is greater than their internal degree so searching for communities in this case becomes challenging. Figure 3 shows a plot of the results of this experiment against three other algorithms from the literature.

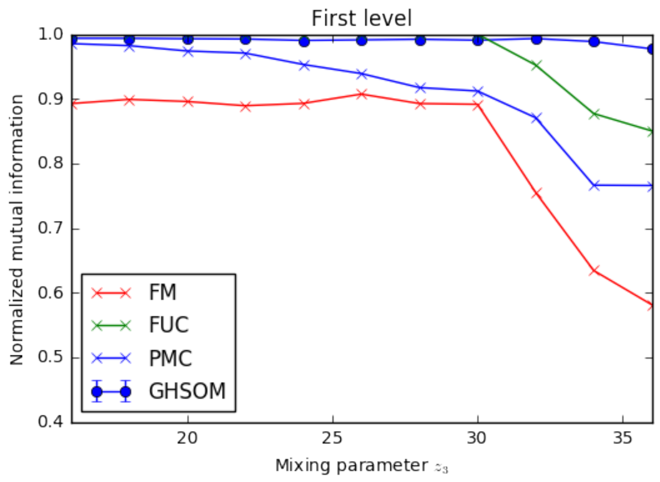
6.3 Preserving Topology

7 Discussion

The aim of this study was to investigate the usefulness of GHSOM as a tool for hierarchical community detection. We found that GHSOM was capable of community detection on both small real world and synthetic networks. With some optimization, GHSOM was able to produce results as good as the other algorithms in the literature on the real world networks (**??**). However, the results achieved by the spearmint software are difficult to reproduce and seem to rely heavily on a lucky random initialization.

On the synthetic experiments, GHSOM consistently achieved almost perfect NMI scores for both level of community, even when the external degree of nodes was greater than the internal degree (fig. 3). Both [Lancichinetti *et al.*, 2009] and [Yang *et al.*, 2013] struggled with these cases and the NMI score for $z_3 > 32$ dropped noticeably (to less than 0.9). In this author's opinion, this is the result of embedding using MDS. In the cases of $z_3 > 32$, each node had 16 connections to nodes in the same micro community (31 other nodes), 16 connections to nodes in the same macro, but not micro community (96 other nodes) and 32-36 connections to the 384 nodes of the network. Even though the external degree of each node was greater than the internal degree, the ratio of connections to possible target nodes was still decreasing. Because of this, the distance between nodes of different communities was still larger than nodes of the same community and so MDS was able to embed these nodes in such a way that identifying communities was still possible. Bonabeau [Bonabeau, 2002] cites [Kernighan and Lin, 1970] who identified that MDS can allow for clusters and communities within a graph to be more easily identifiable even before a cover is found.

But, for practical community detection, GHSOM will require some work. Firstly, GHSOM showed poor scalability. Self-organizing map algorithms are notorious for issues with scalability [Smith, 2002]. All of the experiments here dealt



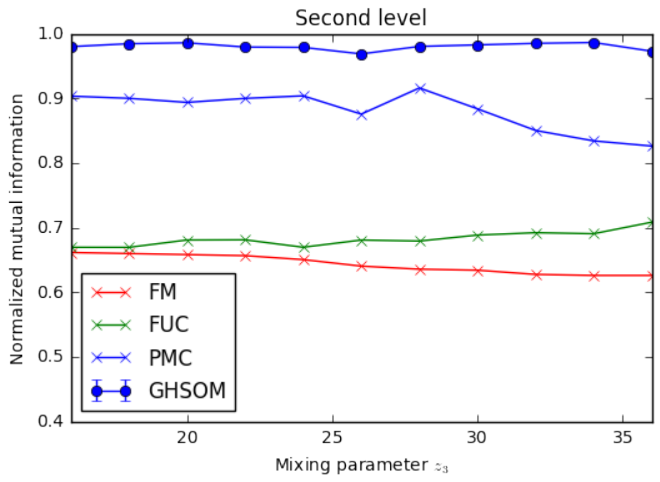


Figure 3: Plot of mixing parameter z_3 against NMI score for both levels of community. 100 networks were generated with each mixing parameter. Results extracted without permission from [Yang *et al.*, 2013] (fig 4).

with small networks (of no more than 512 nodes), but running the algorithm could take upto twenty minutes on a personal laptop. Clearly, as it is presented in this work, GHSOM cannot scale up to networks of thousands of nodes and still be practical. Its scalability could be improved, however. A more intelligent GHSOM implementation would take advantage of GPU processing for the all of the numerical computation. It should also be noted that the training of separate maps from the same level could be implemented to run in parallel, as they deal with entire separate data points, further speeding up run time.

It was also apparent that, while achieving good NMI scores, the number of neurons in the network could overestimate the number of communities in the network. This happened when a neuron was not pointing close to any data points and so was assigned no nodes in the graph. The error of these neurons was zero and this erroneously brought down the error of the entire map. This problem was alleviated somewhat by keeping the learning rate small and training for 10000 epochs, rather than the initial 1000, as as to entire the weights moved sufficiently enough to point to the centre of some data points. However, a better sampling of the input space when inserting neurons, as well as some form of neuron deletion, would be a fruitful addition in the future.

Only one similarity measure (shortest path distance) was used in this study, however, as previously noted, changing the similarity measure could potentially cause GHSOM to identify different communities. An investigation into the impact of the choice of similarity measure and the communities that are conserved across many of them would be an informative future study. One such potential metric is the DSD metric of Cao et al., a metric shown to be more suitable for protein-protein interaction (PPI) networks with few nodes of very high degree [Cao *et al.*, 2013]. One could also experiment with different neuron weight representations, such as the direct-to-graph representation of [Yamakawa *et al.*, 2006], and examine the impact of removing MDS altogether. Additionally, increasing the dimensinoality of the map representation could potentially remove 'false neighbours' from the map [Bonabeau, 2002] and present a more accurate representation of the underlying community structure in the data. By giving neurons overlapping receptive fields in the input space, one could potentially detect overlapping communities in the network as well as hierachical ones [Lancichinetti *et al.*, 2009].

Performance of GHSOM was shown to be heavily dependant upon parameter settings. Fine-tuning parameters is a challenging task, but this affords the algorithm a great deal of customizability, especially in the case of exploratory data mining, where community labels are not known a-priori. One could use the modularity of the communities uncovered as a guide for parameter settings and use a range of parameters to get multiple candidate covers. Additionally, one could overcome the need for so much difficult fine-tuning of parameters altogether by making the entire algorithm more principled by following the example of Bishop et al. in their Generative Topographical Mapping (GTM) algorithm [Bishop *et al.*, 1996].

Overall, GHSOM showed potential as a hierarchical community detection algorithm. With some fine-tuning, it achieved good NMI scores on the hierarchical benchmarks, even with the high mixing factor that other algorithms struggled with. However, work still needs to be done to deal with issues of scalability and make the entire algorithm more principled.

References

[Barber, 2007] Michael J Barber. Modularity and community detection in bipartite networks. *Physical Review E*, 76(6):066102, 2007.

[Bishop *et al.*, 1996] Christopher M Bishop, Markus Svensén, and Christopher KI Williams. Gtm: A principled alternative to the self-organizing map. In *ICANN*, volume 96, pages 165–170, 1996.

[Blondel *et al.*, 2008] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[Bonabeau, 2002] Eric Bonabeau. Graph multidimensional scaling with self-organizing maps. *Information Sciences*, 143(1):159–180, 2002.

[Cao *et al.*, 2013] Mengfei Cao, Hao Zhang, Jisoo Park, Noah M Daniels, Mark E Crovella, Lenore J Cowen, and Benjamin Hescott. Going the distance for protein function prediction: a new distance metric for protein interaction networks. *PloS one*, 8(10):e76339, 2013.

[Clauset *et al.*, 2004] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 70(6):066111, 2004.

[Danon *et al.*, 2006] Leon Danon, Albert Díaz-Guilera, and Alex Arenas. The effect of size heterogeneity on community identification in complex networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2006(11):P11010, 2006.

[Defferrard *et al.*, 2016] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *arXiv preprint arXiv:1606.09375*, 2016.

[Dittenbach *et al.*, 2000] Michael Dittenbach, Dieter Merkl, and Andreas Rauber. The growing hierarchical self-organizing map. In *IJCNN (6)*, pages 15–19, 2000.

[Donath and Hoffman, 1973] William E Donath and Alan J Hoffman. Lower bounds for the partitioning of graphs. *IBM Journal of Research and Development*, 17(5):420–425, 1973.

[Donetti and Munoz, 2004] L Donetti and MA Munoz. Detecting network communities: a new and systematic approach. *Journal of Statistical Mechanics: Theory and Experiment P*, 10012:2004, 2004.

[Fortunato and Barthelemy, 2007] Santo Fortunato and Marc Barthelemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[Fortunato, 2010] Santo Fortunato. Community detection in graphs. *Physics reports*, 486(3):75–174, 2010.

[Fritzke, 1994] Bernd Fritzke. Growing cell structuresa self-organizing network for unsupervised and supervised learning. *Neural networks*, 7(9):1441–1460, 1994.

[Girvan and Newman, 2002] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[Kernighan and Lin, 1970] Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *Bell system technical journal*, 49(2):291–307, 1970.

[Kiang, 2001] Melody Y Kiang. Extending the kohonen self-organizing map networks for clustering analysis. *Computational Statistics & Data Analysis*, 38(2):161–180, 2001.

[Kipf and Welling, 2016] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[Kohonen, 1990] Teuvo Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, 1990.

[Lancichinetti *et al.*, 2009] Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New Journal of Physics*, 11(3):033015, 2009.

[Lusseau *et al.*, 2003] David Lusseau, Karsten Schneider, Oliver J Boisseau, Patti Haase, Elisabeth Slooten, and Steve M Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

[Newman and Girvan, 2004] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[Ng *et al.*, 2002] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[Pons and Latapy, 2011] Pascal Pons and Matthieu Latapy. Post-processing hierarchical community structures: Quality improvements and multi-scale view. *Theoretical Computer Science*, 412(8):892–900, 2011.

[Psorakis *et al.*, 2011] Ioannis Psorakis, Stephen Roberts, Mark Ebden, and Ben Sheldon. Overlapping community detection using bayesian non-negative matrix factorization. *Physical Review E*, 83(6):066114, 2011.

[Radicchi *et al.*, 2004] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 2004.

[Smith, 2002] Andrew James Smith. Applications of the self-organising map to reinforcement learning. *Neural networks*, 15(8):1107–1124, 2002.

[Van Dongen, 2001] Stijn Marinus Van Dongen. *Graph clustering by flow simulation*. PhD thesis, 2001.

[Yamakawa *et al.*, 2006] Takeshi Yamakawa, Keiichi Horio, and Masaharu Hoshino. Self-organizing map with input data represented as graph. In *International Conference on Neural Information Processing*, pages 907–914. Springer, 2006.

[Yang *et al.*, 2013] Bo Yang, Jin Di, Jiming Liu, and Dayou Liu. Hierarchical community detection with applications to real-world network analysis. *Data & Knowledge Engineering*, 83:20–38, 2013.

[Zachary, 1977] Wayne W Zachary. An information flow model for conflict and fission in small groups. *Journal of anthropological research*, pages 452–473, 1977.