Hierarchical Community Detection in Complex Networks using Growing Hierarchical Self Organising Map

David McDonald

Abstract

content...

Acknowledgements

Contents

Chapter 1

Introduction

introduce the problem of community detection here talk about how there is no firm definition etc.

Chapter 2

Background & Literature Review

This chapter aims to give the reader the relevant background information for this work. It will first introduce the problem of community detection and the highlight some of the important work that has already been done. The reader will then be introduced to a topology-preserving neural network, the self-organizing map, which will be used in this work for the purpose of detecting communities.

2.1 Community Detection

This section aims to give the reader a brief oversight of the thoroughly researched field of community detection. Readers looking for an in depth surveys on the various approaches mentioned are pointed towards [1, 2, 3, 4].

2.1.1 Early Work & Cut-Based Approaches

Most early work on community detection was concerned with 'cutting' the graph into modules, in such a way that the number of edges cut was minimised. This, in theory, resulted in the best partition of the network [5, 6, 7]. However, this often favoured cuts of small, peripheral subgraphs, so it was adapted into ratio cut [8], normalised cut [9] and min-max cut [7] that took the number of nodes in each resulting subgraph into account, and thus resulted in a partition that was more balanced.

$$Rcut(A_1, ..., A_k) = \sum_{i=1}^{K} \frac{cut(A_i), cut\bar{(}A_i)}{|A_i|}$$

More recently, 'conductance' has become a common term for defining a good

cut. Conductance, defined as:

$$\phi(S) = \frac{c_s}{\min(Vol(S), Vol(V \setminus S)}$$

with

$$c_s = |\{(u, v) : u \in S, v \notin S\}|$$

is concerned with edges, rather than vertices, and has been used to detect communities in bipartite networks [10] and combined with PageRank [11].

Additional noteworthy early work includes the Metis algorithm [12], that finds the best split of a graph into two equal sized pieces, and the MQI algorithm [13, 14]. The Markov Clustering algorithm proposed by Van Dongen in his PhD thesis [15] simulates a diffusion process on a graph by repeatedly performing stages of expansion and inflation and only keeping the k largest elements for efficiency. His algorithm has found use in Miru, a commercial 3D network visualisation software, to cluster very large graphs quickly.

2.1.2 Modularity-Based Approaches

The work by Girvan and Newman [16] in 2002 marked a significant advance in the fielf by providing the first quantitative measure of a community: modularity. The modularity of a partition of a network [17], defined as

$$Q = \sum_{s=1}^{m} \left[\frac{l_s}{L} - \left(\frac{d_s}{2L} \right)^2 \right] \tag{2.1}$$

scores a 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 [18]. Here, m is the number of modules in the partition, l_s is the number of links in module s, L is the total number of links in the network and d_s is the total degree of the nodes in s. The first term in eq. 2.1 represents fraction of links inside s and the second term represents the expected fraction of random links in the graphs to be in the module. 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 maximised.

The work of Girvan and Newman has been greatly expanded upon in the following years. Radicchi used a similar divisive algorithm, but instead of edge betweenness, used an edge clustering co-efficient based on loops in the network. He also proffers the notion of 'strong' and 'weak' communities based on their internal and external degrees. Clauset [19] sped up Newman's the agglomerate algorithm in [20] to iteratively add links to a module based on their expected increase in modularity. A similar approach was performed by Guimera [21] with simulated annealing. Blondel [22] offers a multi-stage local optimisation of Girvan and Newman's algorithm that iteratively replaces communities with a single node.

Other work with modularity-based quality scores include the work of Rosvall [23] that translates the problem of community detection into the problem of optimally compressing the information in a graph so that the most information can be uncovered when the compression is decoded. He used simulated annealing to minimise a function that represented both compression and data loss. While slow, this approach was also shown to work with dynamic processes in his later work [24]. Newman, himself, offered an alternative approach to his earlier modularity-maximisation algorithm. He used Bayesian inference to deduce the best model to fit the data represented in the graph structure. His algorithm was capable of finding the best group structure for any graph, no just a community structure, but required the number of groups to be know a-priori. Probability theory was also used by Yang in [25] for the construction of his Probabilistically Mining Communities (PMC) algorithm. PMC offers a trade off between using a random walk as a heuristic to reduce the search space and optimising using a constrained quadratic optimisation function. Furthermore, 'label propagation' methods proposed by Raghavan in [26] have had success at detecting communities in real time [27].

Unfortunately, it has been shown that modularity-based approaches have limitations. In particular, Fortunato [18] highlighted what he referred to as the 'resolution limit'. He showed that modularity-based approaches can fail to identify communities that are smaller in size than a scale that depends on the size of the network, and this results in incorrect community division in the cases when even small communities must be considered. Hope is not lost, however, as the resolution limit can be overcome. Ronhovde [28] does not compare to a null model as in eq. 2.1, but instead penalise communities for missing edges. This provided excellent results even on very small communities compared to the size of the overall network.

2.1.3 Spectral-Based Approaches

Spectral-based approaches, in particular spectral clustering, dates back to the work of Donath and Hoffman in 1973 [29]. However, it was popularised by the works of Shi and Malik [9], Meila and Shi [30], Ng et al. [31], and Ding [32]. Typically, they rely upon constructing 'Laplacian' matrices from the raw graph data and eigen-decomposing them. Clustering the resulting matrix results in clusters of the original data points. This method has many advantages over other techinuiques and this has resulted in it becoming extremely popular in the machine learning community. According to [33], 'these methods do not make assumptions about the form of the clusters' and are capable of correctly identifying typically challenging clusters, such as the famous two spirals example. For community detection, they have the additional benefit of efficiency, especially if the graph adjacency matrix is sparse. Additionally, they are able to detect which nodes form connected groups in the graph.

Spectral methods have been combined with traditional hierarchical modularity methods, as in [34], where the graph was eigen-decomposed and projected into a g-dimensional subspace and then the resulting points were clustered to

maximise modularity.

2.1.4 Evolutionary Approaches

Approaches using evolutionary algorithms typically encode the community structure of a graph using the locus based adjacency first proposed in [35] where the genotype consists of N genes and j appearing in gene position i is interpreted as nodes i and j belonging to the same community. This representation was used to great effect as part of the MOCK algorithm [36] which solved the community (or in this case, clustering) problem using a multi-objective optimisation algorithm that attempted to balance compactness and connection of the clusters, and used several novel genetic operators. Variation modifiers are used in [37] to reduce the search pace by taking into account the correlations of the nodes.

2.1.5 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 have recented been generalised for to take a graph input [38]. CNNs have also been used for semi-supervised learning on graphs [39]. Here, some nodes are labelled the labels of every other node are inferred by the model, which is capable of learning both the graph structure and node features.

2.2 Overlapping and Hierarchical Community Detection

In many practical networks, partitioning a network into a cover does not accurately reflect the inherent community structure of the data. Sometimes, nodes can belong to more than one community – sometimes communities overlap. The first algorithm to consider overlapping communities was CFinder in 2006 [40]. Drawingf from the earlier work of Palla and the Clique Percolation Method (CPM) [41], CFinder considered communities as the unions of k-cliques and so rolled k-cliques across the graph to detect communities. While computationally expensive, it was able to deal with overlapping cases, and opened the door for further study. Shen proposed EAGLE in 2009 [42] that used maximal cliques, an agglomerative hierarchical structure and a modified modularity quality function that detected complex overlapping community structures.

The hierarchical nature of many modularity-based clustering method allows them to detect communities at different scales. Complex networks very often contain communities at different scales – communities within communities – and it is informative to investigate the communities identified at different levels of hierarchical algorithms to uncover these. The first algorithm to look for both

overlapping and hierarchical communities was proposed by Lancincetti et al. [43], where they aim to locally optimise

$$f_G = \frac{k_{in}^G}{(k_{in}^G + k_{out}^G)^\alpha}$$

where k_{in}^G, and k_{out}^G are the total internal and external degree of the nodes going into community G, and α controls the size of communities.

Potts methods have also shown good tolerance towards communities that overlap. Both [44] and [45] look for the ground state of spin and interpret the spin configuration that minimises energy of spin glass as the underlying community structure of the state. Due to user-controlled resolution parameters, they are able to identify overlapping and hierarchical communities.

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) [46], and Bayesian non-negative matrix factorisation that performs 'soft-partitioning' and assigns node participation scores to modules [47].

2.3 Self-Organising Maps

This section aims to introduce the reader to the Self-Organising Map, the particular machine learning tool that will be adapted for multi-scale community detection later in this work.

2.3.1 Introduction

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 to a low-dimensional descete map [48, 49, 50]. Unlike other neural networks, SOM uses competitive learning - determining the 'winning' neuron by some distance function – called Learning Vector Quantization (LVQ). Then weights of all neurons are adjusted proportional to their distance to the winning neuron. This results in a map where the topology of the input space is preserved.

2.3.2 SOM for Clustering

SOM for classification has been throughly researched [51] thanks to Learning Vector Quantization (LVQ), an extension to SOM that divides the input space into 'Voronoi Regions' based on Euclidean distance to the weight vector of each neuron, proposed by Kohonen humself [50]. SOM has had successes with medical diagnosis [52], image and character recognition [53, 54], and speech recognition [55]. Clustering with SOM, however, remains relatively under researched. Fritzke [56] offers both an unsupervised and supervised SOM algorithm for clustering. Additionally, his algorithm automatically finds the best network

structure and size using controlled map growth and neuron removal. Both [51] and [57] cluster the output of SOM using k-means and this outperforms the clustering of k-means and hierarchical clustering alone.

2.3.3 SOM with Graph Input

Very little research has been undertaken for using SOM for community detection in graphs. Yamakawa et al. [58] implemented a SOM variation that could take a graph as an input. They constrain the weights of the neurons in the network to only point to edges on the graph. They represent weights as

$$w = (n_i, n_j, \beta)$$

where n_i is a node on the graph, n_j is a neighbour of n_i, and $\beta \in [0, 1]$ presents how far along the edge connecting n_i and n_j the weight vector is pointing. The SOM algorithm remains largely the same, with shortest path on the graph being used as the distance function.

Chapter 3

Method

This chapter aims to give the reader an overview of the community detection algorithm presented in this work.

3.1 MDS projection

Suppose a connected graph $\mathbf{G}$.The first step of the algorithm is to project $\mathbf{G}$ into the Euclidean space. For this, we use Multi-Dimensitonal Scaling (MDS). MDS takes as input a matrix of distances or dissimilarities and attempts to find an embedding of each point such that the relative distances are preserved. An initial guess for a good metric to use here was the shortest path length for each node to each other node in the graph. This distance is finite for all pairs of nodes since $\mathbf{G}$ is connected. Alternative metrics could have been experimented with, such as Diffusion State Distance (DSD) [59] (a metric shown to be more suitable for protein-protein interaction (PPI) networks with few nodes of very high degree), and that is a potential future direction of this work.

After obtaining a dissimilarity matrix $\mathbf{D}$, a kernel matrix $\mathbf{K}$ can be computed using double centring as follows:

$$\mathbf{K} = -\frac{1}{2}\mathbf{C}_n\mathbf{D}\mathbf{C}_n$$

where C_n is the centring matrix, given by

$$\mathbf{C}_n = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$$

$\mathbf{K}$ is eigen-decomposed and factored into

$$\mathbf{K} = \mathbf{U}\Lambda\mathbf{U}^{-1}$$

Since Euclidean spaces have the property that

$$\mathbf{K} = \mathbf{X}\mathbf{X}^T$$

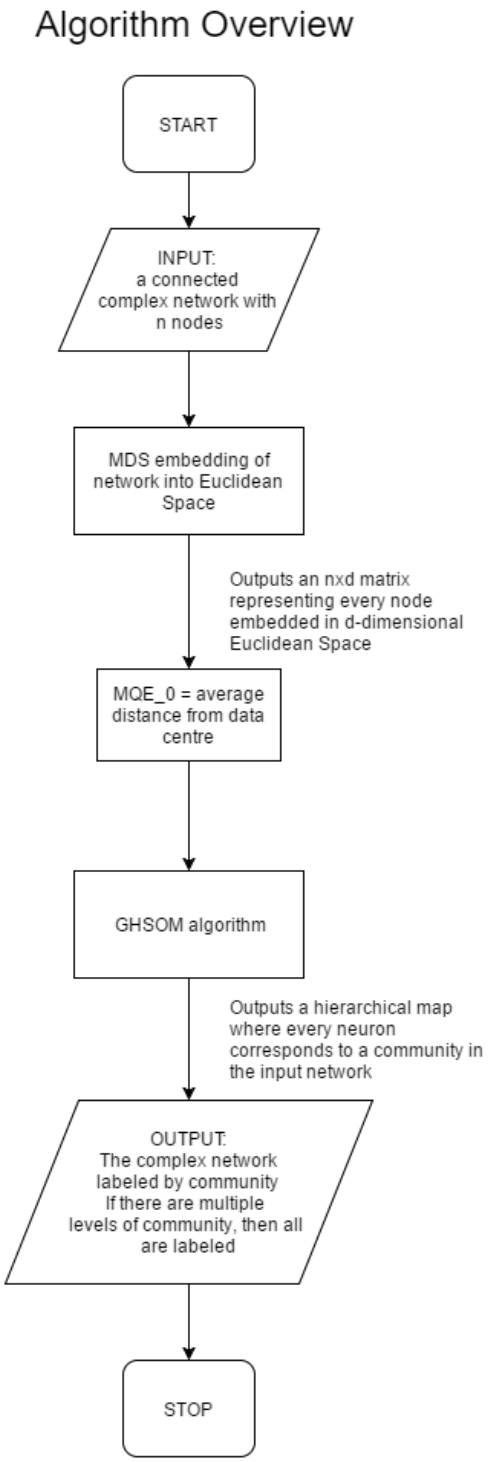


Figure 3.1: An overview of the algorithm

then we have that the embedding matrix $\mathbf{X}$ is given by

$$\mathbf{X} = \mathbf{U}\Lambda^{\frac{1}{2}}$$

MDS is used as a form of dimensionality reduction. A pre-determined dimension k (usually 2 or 3 for data visualisation) is given to the algorithm and only the k eigenvectors and eigenvalues are used to compute $\mathbf{X}$ that minimise strain, given by

$$strain_{\mathbf{D}}(x_1, x_2, ..., x_n) = \left(\frac{\Sigma_{i,j}(k_{i,j} - \langle x_i, x_j \rangle)^2}{\Sigma_{i,j} k_{i,j}^2} \right)^{\frac{1}{2}}$$

In order to prevent the need to specify k a-priori, this algorithm determined k dynamically by ordering the eigenvalues into descending order and calculating their sum. k was initialised to $k = n$ and then the sum of the first k eigenvalues was computed and then divided by the sum of all the eigenvalues. If this was greater than 0.95, then k was decreased by 1 and this process was repeated. In other words, we are looking for the largest k such that

$$\frac{\Sigma_{i=1}^{k} \lambda_i}{\Sigma_{j=1}^{n} \lambda_j} \geq 0.95$$

holds. 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 co-ordinates of node i in k-dimensional space.

3.2 GHSOM

Embedding the n nodes into k dimensional space translates the problem of finding communities to identifying clusters. This is because the distance (by whatever metric used) between two nodes within the same community should be smaller than two nodes belonging to two different communities. Therefore, all of the nodes within a community should be embedded closely together and so identifying clusters in $\mathbb{R}^k$ is the same as identifying communities in $\mathbf{G}$.

The Growing Hierarchical Self Organising Map (GHSOM) algorithm is used to identify clusters in this work. GHSOM has a number of advantages over other clustering algorithms. Firstly, the number of clusters does not need to be known a-priori, and also GHSOM is capable of detecting the hierarchical structure of clusters (ie. clusters within clusters). GHSOM also inherits the primary advantage of the standard SOM algorithm in that the topology of the input space is preserved in the output map. This means neighbouring features of the input space will appear together in the output map. This makes GHSOM an excellent choice for a general tool for detecting communities in biological networks that so commonly feature a hierarchical structure. The topology preservation property will show the overall position of the communities within the network as a whole.

3.2.1 Hyper-Parameters

Two hyper-parameters must be given to GHSOM before it begins. The first is ϵ_{sg} and this determines the size of the maps that will be created, and the second, ϵ_{en}, determines the granularity of the maps. Both of these will be discussed more later. Additionally, GHSOM uses a measure of error, that its creators have called Mean Quantisation Error (or MQE). Let the MQE of a single unit i be denoted $\mathbf{mqe}_i$ and the MQE of an entire network m be denoted $\mathbf{MQE}_m$.

3.2.2 Initial Step

GHSOM begins with layer 0 consisting of a single node. This node is given a weight $\mathbf{w}$ that points to the centre of all the datapoints. The MQE of this node is computed as

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

3.2.3 Training the First Layer

GHSOM then begins training the first layer of the network. The first layer is initialised to just one neuron. The weights of these two neurons are initially set to be random vectors in $\mathbb{R}^k$. The network is then trained as in the standard SOM algorithm.

The network is then trained for $\lambda = 1000$ epochs. One training epoch consists of presenting each training pattern (the embedded points of $\mathbf{G}$) to the network. For each pattern $\mathbf{x}$, the 'winning' neuron i^* is determined as

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

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

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

The weights are also adjusted proportionally to the learning rate $\eta(t)$, which is typically a small number of the order $1e-3$, and the so-called neighbourhood function $h(i, i^*, \sigma(t))$, given by

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

where $\mathbf{r}_i$ is the position of neuron i in the network. The neighbourhood function maintains the topological ordering of the map, as neurons closer to the winning neuron in the map move towards $\mathbf{x}$ more.

Both $\eta(t)$ and $\sigma(t)$ can vary with time and typically decrease as the number of training epochs increases. This allows the movements of the weights to gradually decrease and become fine-tuned.

At the end of each training epoch, the order of the training patterns is shuffled. The neighbourhood width $\sigma(t)$ is also dropped in the following manner

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

3.2.4 Growing the Map

After $\lambda = 1000$ training epochs have passed, every node in $\mathbf{G}$ is assigned to the neuron whose weight vector is closest and the $\mathbf{mqe}_i$ for each neuron is calculated as well as the overall $\mathbf{MQE}_1$ of the network. If

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

then the network is expanded, by inserting a single neuron.

Network expansion largely follows the example proposed by Bernd Fritzke [56] and aims to preserve a simplicial structure within the network whenever a new neuron is added. Firstly, one must identify the 'error unit' e – the neuron with the greatest error in the network. Then retrieve the list of all of the neighbours of e.

Single Neuron in Network

In the case that e has no neighbours (the network consists of one neuron) then a new neuron is added to he network with a random weight vector and connected to the e.

Two Neurons in Network

In the case of one neighbour, the weight vector of the new neuron is given the mean of the weight vectors of the the existing two neurons in the network and then connected to both neurons.

Greater than Two Neurons

In the case of two or greater neighbours, 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).

3.2.5 Termination of Network Growth

The network is again trained for $\lambda = 1000$ epochs and $\mathbf{MQE}_1$ is calculated. If

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

then expansion terminates, else another neuron is inserted and the process repeats.

3.2.6 Adding a New Layer to the Network

After growth for this layer has terminated the GHSOM determines whether or not to build another layer of the network by expanding specific neurons in this layer. This is determined by the choice of ϵ_{en}. GHSOM scans across all the neurons i in the network and checks if

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

If a neuron i^* is located with this property then a new network is build (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 network.

3.2.7 Termination

GHSOM continues to train and expand until

$$\mathbf{MQE}_m < \epsilon_{sg}\mathbf{mqe}_0$$

for all networks m and

$$\mathbf{mqe}_i < \epsilon_{en}\mathbf{mqe}_0$$

for all neurons i.

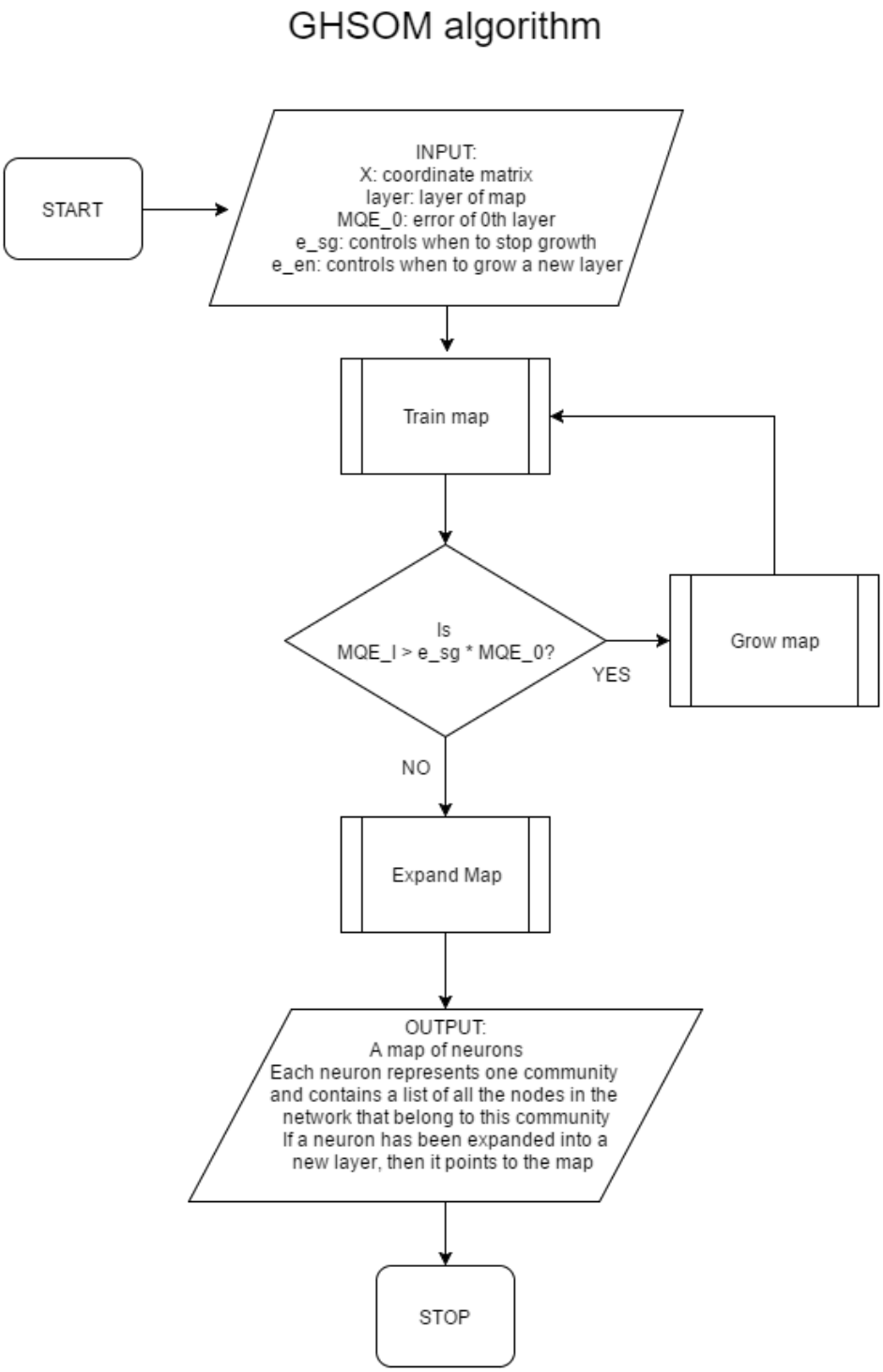
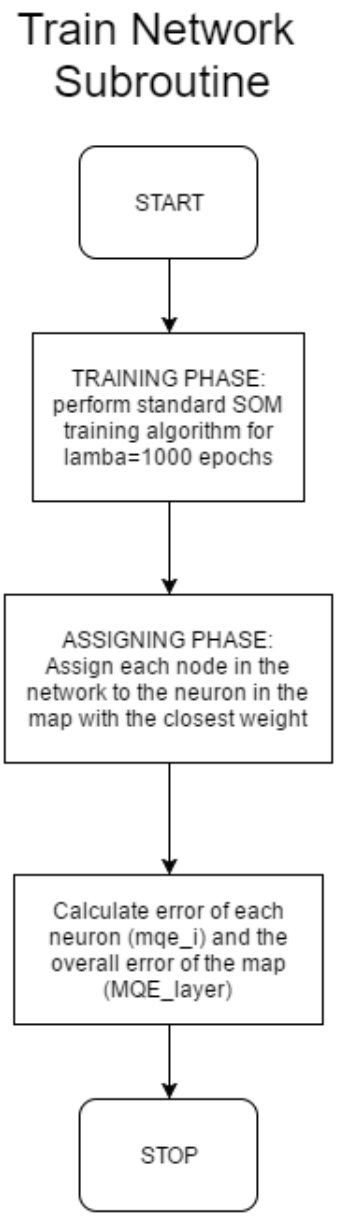
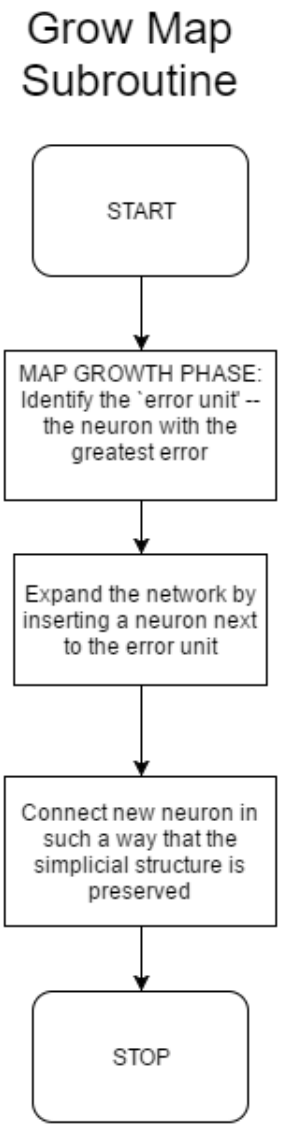


Figure 3.2: An overview of the GHSOM algorithm



(a) The procedure for training the map.



(b) The procedure for growing the map.

Figure 3.3: Procedures for training and expanding.

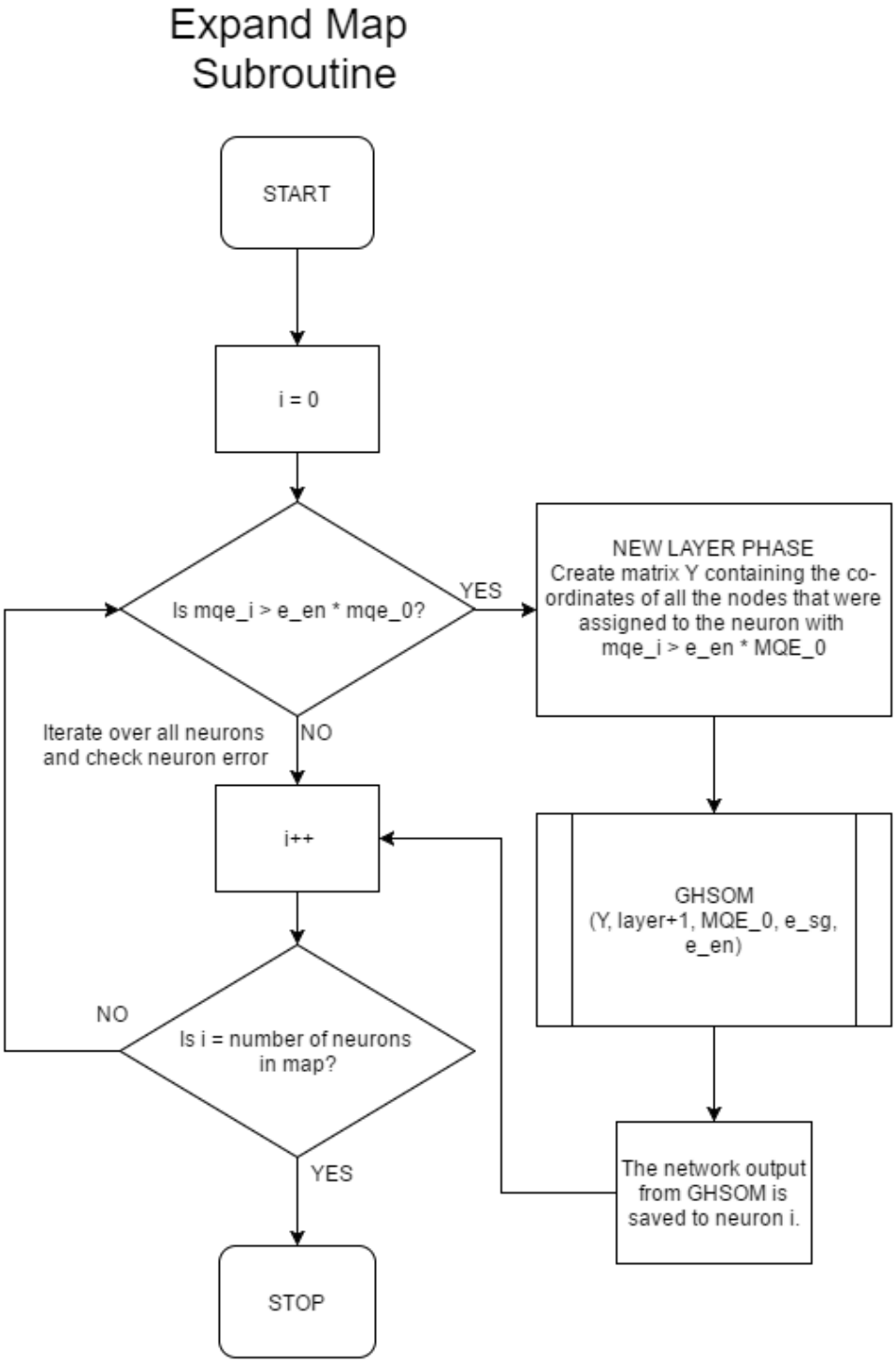


Figure 3.4: The procedure for expanding the map.

Chapter 4

Results

This chapter will evaluate the performance of GHSOM for hierarchical community detection against some of the other leading algorithms in the literature.

4.1 Bayesian Optimisation

The choice of ϵ_{sg} affects the detail of the resulting maps in GHSOM and the choice of ϵ_{en} affects the depth of the resulting hierarchy. Optimal values for these proved to be very problem-dependant. As a result, a Bayesian Optimisation approach was used to optimise these along with the training parameters of SOM (the learning rate η, the starting neighbourhood width σ_0, and the bounds for the initialisation of weights w).

The Bayesian Optimisation software Spearmint was used for this. Written in Python 2.7 by Jasper Snoek et al. and based on the work from the following publications: [60, 61, 62, 63, 64], Spearmint automatically runs experiments and adjusts a number of parameters to minimise an objective in as few runs as possible.

4.1.1 Objective Function

Spearmint was used to minimise

$$f(X, Y) = 1 - \mathrm{NMI}(X, Y)$$

with $\mathrm{NMI}(X, Y)$ as the Normalised Mutual Information score between the predicted set of class labels X and the actual set of class labels Y, defined as

$$\begin{aligned}\mathrm{NMI}(X : Y) &= \frac{H(X) + H(Y) - H(X, Y)}{(H(X) + H(Y))/2} \\ &= \frac{1}{\sqrt{H(X)H(Y)}} \sum_{i=1}^{R} \sum_{j=1}^{C} P(i, j) \log \frac{P(i, j)}{P(i)P'(j)}\end{aligned}$$

where R and C are the number of clusters in X and Y respectively, and $P(i)$ is the probability of a random sample occurring in cluster X_i and $P'(j)$ is the probability of a random sample occurring in cluster Y_j. $H(X)$ is the entropy of random variable X, given by

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i)$$

NMI gives a score between 0 (indicating no mutual information) and 1 (indicating perfect mutual information). In other words, the greater the NMI score, the more nodes are assigned to the correct community. Therefore, we aim to minimise f.

4.2 Real-World Benchmarks

4.2.1 Zachary's Karate Club

[65]

4.2.2 American College Football

[16]

4.2.3 Polbooks

4.2.4 Dolphins

[66, 67]

4.2.5 Comparisons with other algorithms

TABLE OF RESULTS HERE

4.3 Synthetic Hierarchical Graphs

generated with the software given by Lancichinetti and Fortunato binary: [68] hierarchical: [43]

Chapter 5

Implementation Details

To be omitted from paper

5.1 Network Representation

The representation of a network in this work was different to that of GHSOM's creators. Networks are represented as NetworkX graph objects, with neurons represented as nodes on the graph. The distance ($||\mathbf{r}_i - \mathbf{r}_{i^*}||$) between neuron i and neuron j is defined as the length of the shortest path between the nodes on the graph. The nodes in the graph contain attributes that hold its weight vector and the list of nodes in $\mathbf{G}$ assigned to this neuron's cluster. This representation was selected to allow for the expansion of the network to happen one neuron at a time, rather than inserting an entire row or column of neurons at once. The rational here was that by slowly expanding the network one neuron at a time, the resulting map would more accurately reflect the number of communities in the data. In contrast with expanding the network by adding an entire row or column at once and potentially over estimating that number.

Chapter 6

Discussion

6.1 Conclusions

6.2 Future Directions

Bibliography

[1] Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

[2] Qing Cai, Lijia Ma, Maoguo Gong, and Dayong Tian. A survey on network community detection based on evolutionary computation. *International Journal of Bio-Inspired Computation*, 8(2):84–98, 2016.

[3] Lingjia Zhang. *Community detection in network analysis: a survey*. PhD thesis, NA, 2016.

[4] Jierui Xie, Stephen Kelley, and Boleslaw K Szymanski. Overlapping community detection in networks: The state-of-the-art and comparative study. *ACM Computing Surveys (csur)*, 45(4):43, 2013.

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

[6] Charles M Fiduccia and Robert M Mattheyses. A linear-time heuristic for improving network partitions. In *Design Automation, 1982. 19th Conference on*, pages 175–181. IEEE, 1982.

[7] Chris HQ Ding, Xiaofeng He, Hongyuan Zha, Ming Gu, and Horst D Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 107–114. IEEE, 2001.

[8] Y-C Wei and C-K Cheng. Ratio cut partitioning for hierarchical designs. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 10(7):911–921, 1991.

[9] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

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

[11] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 475–486. IEEE, 2006.

[12] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

[13] Giorgio Gallo, Michael D Grigoriadis, and Robert E Tarjan. A fast parametric maximum flow algorithm and applications. *SIAM Journal on Computing*, 18(1):30–55, 1989.

[14] Kevin Lang and Satish Rao. A flow-based method for improving the expansion or conductance of graph cuts. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 325–337. Springer, 2004.

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

[16] 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.

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

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

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

[20] Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

[21] Roger Guimera and Luis A Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

[22] 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.

[23] Martin Rosvall and Carl T Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18):7327–7331, 2007.

[24] Martin Rosvall and Carl T Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[25] 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.

[26] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

[27] Ian XY Leung, Pan Hui, Pietro Lio, and Jon Crowcroft. Towards real-time community detection in large networks. *Physical Review E*, 79(6):066107, 2009.

[28] Peter Ronhovde and Zohar Nussinov. Local resolution-limit-free potts model for community detection. *Physical Review E*, 81(4):046114, 2010.

[29] 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.

[30] Marina Meila and Jianbo Shi. A random walks view of spectral segmentation. 2001.

[31] 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.

[32] Chris Ding. A tutorial on spectral clustering. In *Talk presented at ICML.*, 2004.

[33] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

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

[35] YoungJa Park and ManSuk Song. A genetic algorithm for clustering problems. In *Proceedings of the third annual conference on genetic programming*, pages 568–575, 1998.

[36] Julia Handl and Joshua Knowles. An evolutionary approach to multiobjective clustering. *IEEE transactions on Evolutionary Computation*, 11(1):56–76, 2007.

[37] Clara Pizzuti. Ga-net: A genetic algorithm for community detection in social networks. In *International Conference on Parallel Problem Solving from Nature*, pages 1081–1090. Springer, 2008.

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

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

[40] Balázs Adamcsek, Gergely Palla, Illés J Farkas, Imre Derényi, and Tamás Vicsek. Cfinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 22(8):1021–1023, 2006.

[41] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

[42] Huawei Shen, Xueqi Cheng, Kai Cai, and Mao-Bin Hu. Detect overlapping and hierarchical community structure in networks. *Physica A: Statistical Mechanics and its Applications*, 388(8):1706–1712, 2009.

[43] 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.

[44] Jörg Reichardt and Stefan Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1):016110, 2006.

[45] Peter Ronhovde and Zohar Nussinov. Multiresolution community detection for megascale networks by information-based replica correlations. *Physical Review E*, 80(1):016109, 2009.

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

[47] 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.

[48] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological cybernetics*, 43(1):59–69, 1982.

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

[50] Teuvo Kohonen. Learning vector quantization. In *Self-Organizing Maps*, pages 175–189. Springer, 1995.

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

[52] L Vercauteren, G Sieben, and M Praet. The classification of brain tumours by a topological map. 1990.

[53] Michael Sabourin and Amar Mitiche. Modeling and classification of shape using a kohonen associative memory with selective multiresolution. *Neural Networks*, 6(2):275–283, 1993.

[54] Alberto Delbimbo, Leonardo Landi, and Simone Santini. Three-dimensional planar-faced object classification with kohonen maps. *Optical Engineering*, 32(6):1222–1234, 1993.

[55] Lea Leinonen, Tapio Hiltunen, Kari Torkkola, and Jari Kangas. Self-organized acoustic feature map in detection of fricative-vowel coarticulation. *The Journal of the Acoustical Society of America*, 93(6):3468–3474, 1993.

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

[57] Juha Vesanto and Esa Alhoniemi. Clustering of the self-organizing map. *IEEE transactions on neural networks*, 11(3):586–600, 2000.

[58] 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.

[59] 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.

[60] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*, pages 2951–2959, 2012.

[61] Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in neural information processing systems*, pages 2004–2012, 2013.

[62] Jasper Snoek. *Bayesian Optimization and Semiparametric Models with Applications to Assistive Technology*. PhD thesis, Citeseer, 2013.

[63] Jasper Snoek, Kevin Swersky, Richard S Zemel, and Ryan P Adams. Input warping for bayesian optimization of non-stationary functions. In *ICML*, pages 1674–1682, 2014.

[64] Michael A Gelbart, Jasper Snoek, and Ryan P Adams. Bayesian optimization with unknown constraints. *arXiv preprint arXiv:1403.5607*, 2014.

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

[66] David Lusseau. The emergent properties of a dolphin social network. *Proceedings of the Royal Society of London B: Biological Sciences*, 270(Suppl 2):S186–S188, 2003.

[67] 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.

[68] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: a comparative analysis. *Physical review E*, 80(5):056117, 2009.