RSMG Progress Report 1

David McDonald
Superviser: Dr. Shan He

Contents

Chapter 1

Introduction

The overall goal of my PhD is to construct a complex network model of biological data, by combining gene expression levels, known pathways, protein interactions and known semantic information. Then, through feature learning techniques, use inference to generate testable hypotheses from this model about the complex and still largely unknown interactions that comprise the world in which we live. This can be broadly broken down into two areas: complex network analysis and feature learning.

A complex network is a graph that is comprised of non-trivial or uniform features. These networks often arise when modelling real-word systems. The early belief was that the interactions of such seemingly unrelated things as proteins, social interactions and the internet were random and unconnected. However, it has been shown that most real world systems have the same basic architecture [1]. Real world networks often scale-free in that the distribution of node degree is a power law. Connections are preferentially made between nodes with a probability proportional to their existing degree. Some complex networks also are characterised by the 'small world' phenomenon, where one would expect a small average shortest path length and a high degree of clustering [2]. The underlying similarity of the interactions between agents in real-world phenomena is surprising and launched the popularity of complex network research. Research that has swiftly captured the imagination of researchers of many fields; fields such as epidemiology, mathematics, computer science, sociology and biology.

Feature learning comprises a set of machine learning techniques that extract useful features from raw data. These features aim to model the underlying constructs of the data and can be used as the first step of more complex machine learning tasks, such as classification. The recent boom in popularity of deep learning and advent of efficient training algorithms has lead to deep models designed te hierarchically extract features from the data. For example, in the landmark computer vision paper of 2012, Krizhevsky et al. [3] trained a deep convolutional neural network to extract hierarchically features from colour images, achieving state-of-the art results. Visualisation of the trained filters shows that the network was, indeed, extracting features hierarchically, the first layer looked for edges, the next for shapes etc. This is analogous to how the human brain processes images.

Generative models, such as the Restricted Boltzmann Machine (RBM), suppose that there are hidden latent variables controlling the data and aim to use inference to learn the distribution of the data. RBMs suppose a conditional independence between components of a particular observance of the data, and between the hidden, latent variables. This allows for very efficient training algorithms such as the Geoffrey Hinton's contrastive divergence algorithm [4]. However, by adding intra-level connections between the hidden latent variables, we can also learn the relationships between these variables.

While feature learning commonly involves reducing the dimension of the data by condensing a large dataset to a small number of features, actually allowing for a number of features much larger than the dimension of the original data. By imposing a sparsity constraint on the latent representation of the data, such that the expected output of a particular hidden unit is close to 0, sparse coding is able to learn a 'dictionary' of codes whereby a particular data point is composed of few of these codes.

Chapter 2

Community Detection in Complex Networks

Most real-world complex networks contain subsets of nodes that contain a higher degree of inter-connectivity than the rest of the network [5]. These subsets are commonly referred to as communities. A rigorous definition for a 'community' within a network still seems to elude the scientific community [5]. 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 [6] 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 [6]. 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 [7], biological [8], and world-wide-web [9] analysis.

Most real world networks do not solely contain communities at one scale [5, 10]. 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' [8].

The Growing Hierarchical Self-Organizing Map (GHSOM), which was first described by Dittenbach et al. [11], 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.

2.1 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 comparisions are directed to [12].

2.1.1 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 [13]. 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 [12].

Spectral clustering dates back to the work of Donath and Hoffman in 1973 [14]. However, it was popularized in the early 2000s [15]. 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 [16].

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 [17].

2.1.2 Modularity-Based Approaches

The seminal work of Girvan and Newman [6] 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 [18], iteratively adding links to a module based on their expected increase in modularity [19], and multi-stage local optimization [20].

2.1.3 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 [21]. 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 [10].

2.1.4 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 [22]. CNNs have also been used for semi-supervised learning on graphs, where the they are capable of learning both graph structure and node features [23].

2.2 Hierarchical Community Detection

The hierarchical nature of modularity-based clustering methods can allow them to detect communities at different scales. [5] 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) [24], and Bayesian non-negative matrix factorisation that performs 'soft-partitioning' and assigns node participation scores to modules [25].

Chapter 3

The Growing Hierarchical Self-Organising Map

The Growing Hierarchical Self-Organising Map (GHSOM), proposed first in 2000 [11] for clustering data points into hierarchical clusters is the main focus of my research for this first six months. In particular, I am investigating its usefulness at community detection in complex networks that often contain an inherent hierarchical structure [5]. For example, a social network may contain a community corresponding to "academics" that could be further broken up into communities of "computer science", "physics", "biology" etc. Additionally, GHSOM can preserve the topological information of the data it is modelling. Features, such as communities, that are close together in the input space should be close together in the map and so I also aim to investigate cooperation between communities.

I primarily use two 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.

3.1 Embedding of Vertices using MDS

Multi-Dimensional Scaling (MDS) was used to first embed 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 – say shortest path distance between nodes in the network– may result in different communities than for another measure – semantic similarities of genes, based on enriched GO terms, in a co-expression network. The definition of "community" is not yet rigorous and depends largely on what a researcher is interested in investigating [26], and the ability to extract multiple community partitions based on the same data is desirable.

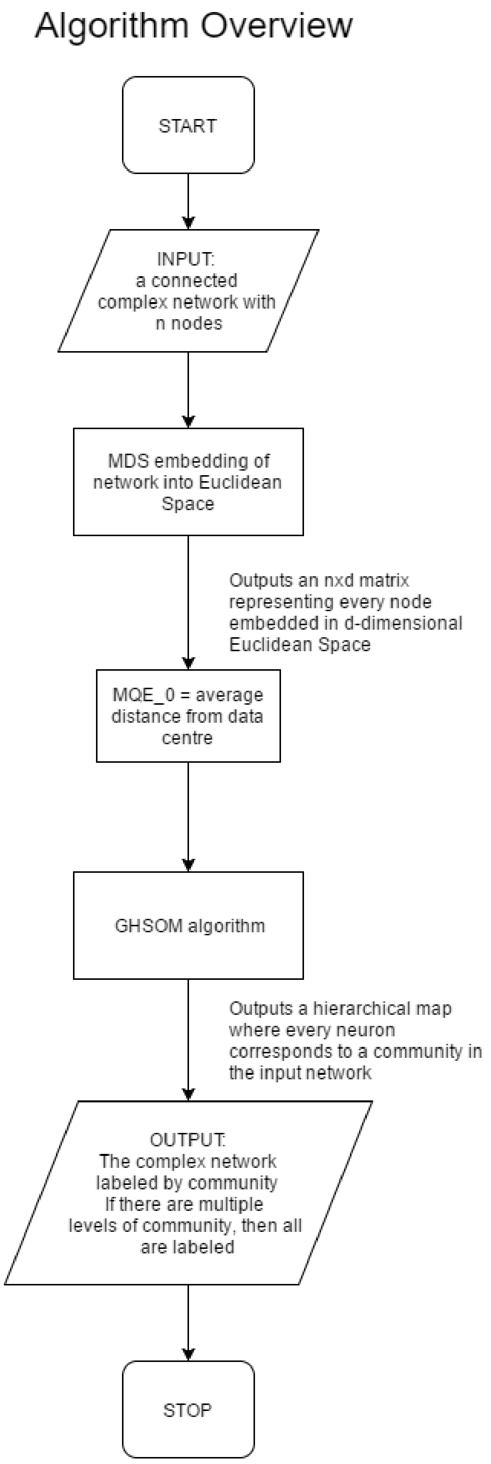

Algorithm Overview
START
INPUT: a connected complex network with n nodes
MDS embedding of network into Euclidean Space
Outputs an nxd matrix representing every node embedded in d-dimensional Euclidean Space
MQE_0 = average distance from data centre
GHSOM algorithm
Outputs a hierarchical map where every neuron corresponds to a community in the input network
OUTPUT: The complex network labeled by community If there are multiple levels of community, then all are labeled
STOP

Figure 3.1: A flowchart providing an overview of the entire algorithm.

MDS takes as input a matrix **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 [27]. This distance is well defined and finite for all pairs of nodes in the network since **G** is a connected graph.

We then use double centring to construct a kernel matrix **K** from **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:

$$k^* = \underset{k}{\operatorname{argmax}} \left(\frac{\Sigma_{i=1}^{k} \lambda_i}{\Sigma_{j=1}^{n} \lambda_j} \geq 0.95 \right) \tag{3.1}$$

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

3.2 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 **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 **X**. For clustering the data points in **X**, we use GHSOM.

3.3 Hyper-Parameters

GHSOM takes two hyper parameters. 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. [11, 28] 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 $\mathbf{MQE}_m$.

3.4 MQE$_0$

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

$$\mathbf{w} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x}_i \tag{3.2}$$

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

$$\mathbf{MQE}_0 = \mathbf{mqe}_0 = \frac{1}{n}\sum_{i=1}^{n} ||\mathbf{w} - \mathbf{x}_i|| \tag{3.3}$$

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

3.5 Training the First Layer

GHSOM then begins by training the first layer map. The first layer is initialized to just one neuron. The weight of this neuron is initially set to be equal to a random data point. The map is then trained as in the standard SOM algorithm, found in [29] 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:

$$i^* = \underset{i}{\operatorname{argmin}}\, ||\mathbf{x} - \mathbf{w}_i|| \tag{3.4}$$

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.

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta(t)h(i, i^*, \sigma(t))(\mathbf{x} - \mathbf{w}_i) \tag{3.5}$$

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

$$h(i, i^*, \sigma(t)) = \exp\left(- \frac{||\mathbf{r}_i - \mathbf{r}_{i^*}||^2}{2\sigma(t)^2} \right) \tag{3.6}$$

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:

$$\sigma(t) = \sigma_0 \exp\left(- \frac{2\sigma_0 t}{\lambda} \right) \tag{3.7}$$

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

3.6 Growing the Map

The original GHSOM paper describes map expansion by inserting a new row or column of neurons into the map. Since, every neuron represents a community, I elected not to follow this example, as I did not want to over estimate the number of communities in the network. As a result, I grow the map one neuron at a time. 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

$$\mathbf{MQE}_1 \geq \epsilon_{sg}\mathbf{mqe}_0 \tag{3.8}$$

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 [30] and aims to preserve a simplicial structure within the map whenever a new neuron is inserted. This ensures that neurons in the map 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. A new neuron, i is inserted into the map and assigned a weight vector v equal to a random data point that is assigned to e. A connection is then made between i and e. If e has any neighbours, then an additional connection is made between i and the neighbour of e that points closest to v.

3.7 Termination of Map Growth

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

$$\mathbf{MQE}_1 < \epsilon_{sg}\mathbf{mqe}_0 \tag{3.9}$$

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

3.8 Neuron Deletion

If a neuron is assigned data points after training, it is deleted from the map. If this results in a disconnected map, then a connected is added between the neurons of each separate connected component that point closest together in the input space.

3.9 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

$$\mathbf{mqe}_i \geq \epsilon_{en}\mathbf{mqe}_0 \tag{3.10}$$

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 data points in $\mathbf{G}$ that were assigned to i^* are passed to the new map. Training, growth and expansion of this new map follows exactly the same procedure as described above.

3.10 Termination of GHSOM

GHSOM continues to train, grow and expand maps until

$$\mathbf{MQE}_m < \epsilon_{sg}\mathbf{MQE}_0 \tag{3.11}$$

for all maps m and

$$\mathbf{mqe}_i < \epsilon_{en}\mathbf{MQE}_0 \tag{3.12}$$

for all neurons i in all maps. 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.

Chapter 4

Results so Far

4.1 Real World Benchmarks

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 4.2 details the consistent parameter settings used for each network type. Four representative algorithms from the literature are used for comparison: MCL [17], FM [19], FUC [20], and PMC [10].

4.2 Synthetic Hierarchical Benchmarks

In order to investigate how good GHSOM was at uncovering the hierarchical community structure of complex networks, synthetic benchmark networks were generated using software provided by Lancichinetti et al. [5]. 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

Network	**Description**
Karate	Social network of karate club [31].
Dolphin	Social network of dolphins living in New Zealand [32].
Polbooks	Network of books sold by `www.amazon.com` and published around the 2004 presidential election [6].
Football	A network of college football games in Fall 2000 [6].

Table 4.1: Description of four real-world networks.

Parameter	**Description**	**Network Type**	
		Real-World	Synthetic
η	Learning rate	0.0001	0.0001
σ	Initial neighbourhood size	1	1
ϵ_{sg}	Stop map growth parameter	0.8	0.8
ϵ_{en}	Grow new map parameter	0.8	0.8

Table 4.2: Parameter setting for both types of network.

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 4.3: 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 [10] (table 2) without permission. Hyper-parameters are kept consistent across all networks, given in table 4.2.

Parameter	**Network**			
	Karate	Dolphin	Polbook	Football
#jobs	28	521	562	181
η	0.0001	0.879	0.999	0.0587
σ	0.817	0.001	0.650	1.0
ϵ_{sg}	0.988	0.558	1.0	0.451
ϵ_{en}	1.0	0.3	0.393	0.3
#comms	2	4	3	12
#comms det.	2	4	2	11
NMI score	1.0	0.640	0.688	0.874

Table 4.4: Spearmint optimized parameter settings and NMI scores for real world networks (to 3 s.f.). **#jobs** is the number of completed jobs before the NMI score was recorded. Maps were trained for $\lambda = 1000$ training epochs before calculating quantization error.

networks of nodes belonging to communities at two levels: one at the micro level and one at the macro level.

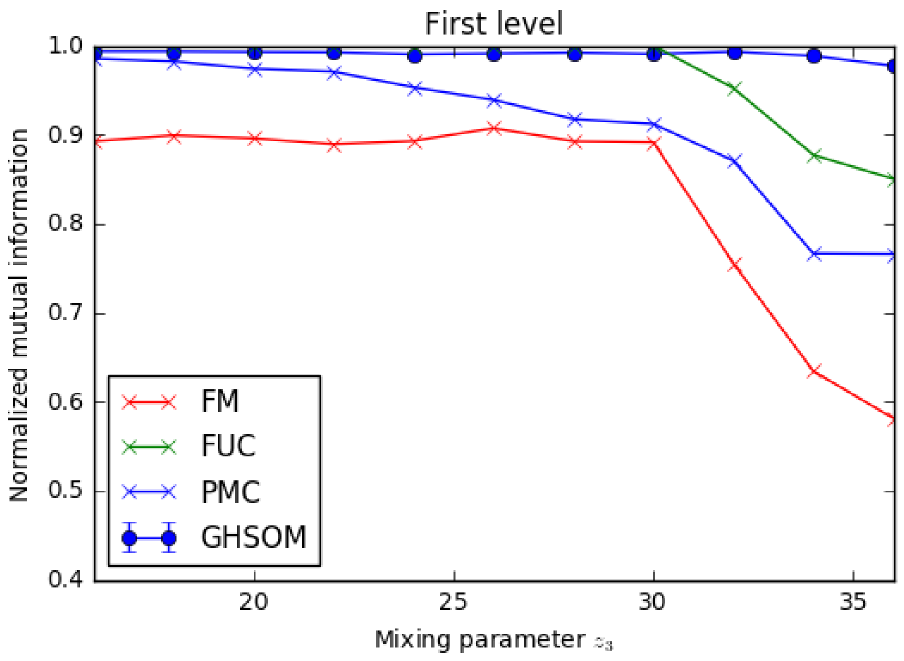
The software allows for the adjustment of three mixing parameters: z_1, z_2, and z_3, which control the number of connections between nodes of the same micro community, same macro community and other communities in the network respectively. As in [5] and [10], we keep z_1 and z_2 fixed at 16, and vary 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 4.1 shows a plot of the results of this experiment against three other algorithms from the literature. FM [19], FUC [20] and PMC [10].

4.3 Dependence upon Parameter Settings

After some experimentation, it became apparent that the performance of the algorithm depended greatly upon parameter settings.

Bayesian optimization [33] software was used to find the optimal parameter settings for each real world network. Table 4.4 shows the settings found and the NMI scores achieved. Spearmint does not record the random seed that generated these results so they are merely used here as an indication of the potential of GHSOM. Optimization achieved good results compared to the results found in the literature and so some time was spent attempting to make the setting of parameters more principled.

Since one neuron in the map corresponds to one community in the network,



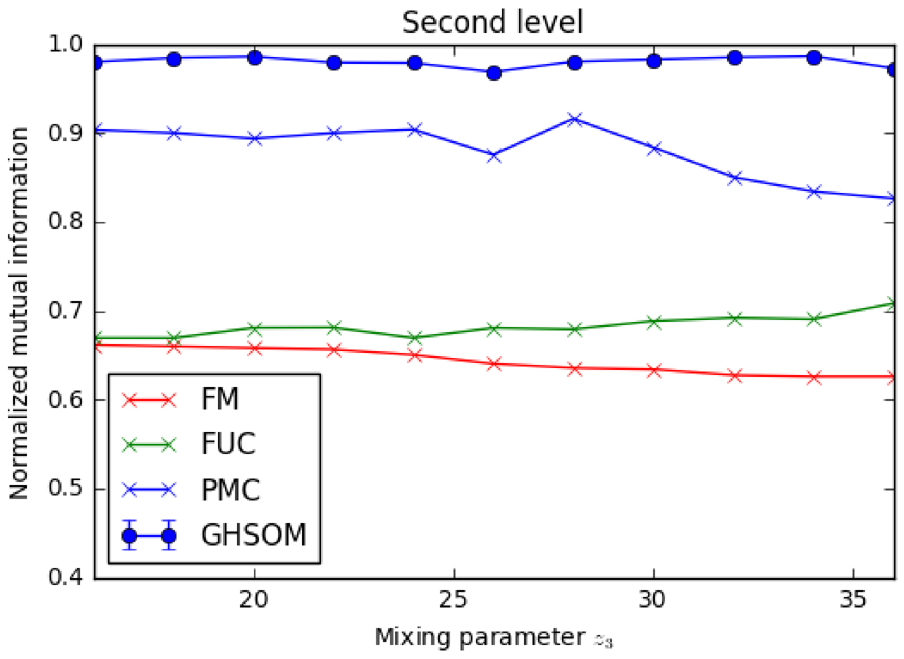


Figure 4.1: Plot of mixing parameter z_3 against NMI score for both levels of community. 100 networks were generated with each mixing parameter. Results extracted without permission from [10] (fig 4).

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 4.2: Community size against setting of 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.

and we desire the final number of neurons in the map to be equal to the number of communities in the data, the investigation focussed on ϵ_{sg}, which controlled final map size.

In order to make the choice of setting ϵ_{sg} more principled, an experiment was devised to measure the impact of community size on the best choice of 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 4.2 shows the results of this experiment. Since, in general, we do not know the number of communities in a network a-priori, the experiment was repeated, this time measuring network density ρ, a measure that is simple to calculate for any network in terms of number of nodes n and number of edges m as

$$\rho = \frac{2m}{n(n-1)}$$

Figure 4.3 shows a plot of the result of this experiment.

The experiments did not seem to show any significant correlations and so investigation into deriving a principled method of parameter setting was dropped, in favour of more a productive (and interesting) research goal.

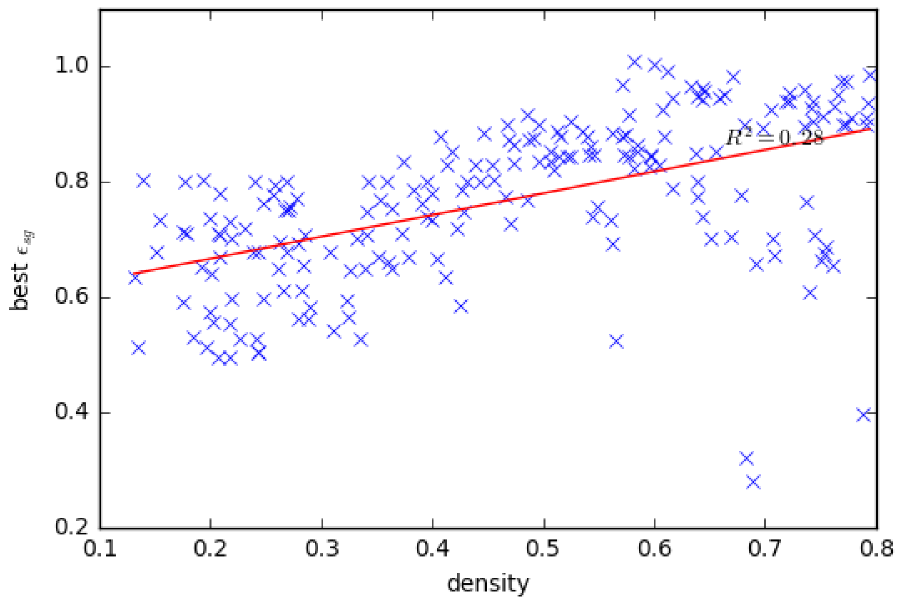


Figure 4.3: Plot of network density vs. best setting for ϵ_{sg}. 200 random networks of 64 nodes were generated with a random number of edges and communities.

Chapter 5

Functional Similarity of Neighbouring Communities

Each neuron on the map represents one community in the original complex network. And, due to the topology preserving property of SOM, adjacent neurons in the map represent adjacent communities (see fig. 5.1). Since, in complex networks, a community can have some real world meaning, then a natural question to ask is whether neighbouring communities interact (or even co-operate) in their function. Experiments were undertaken to investigate this.

5.1 Background: The Gene Ontology

The Gene Ontology (GO) provides a controlled universal vocabulary for the communication known information about genes across all *eukaryotes* (organisms with cells containing a nucleus) [34]. It is accessible world wide via the internet and update constantly as new discoveries are made by researchers. It is represented as three directed acyclic graphs (DAGs) that correspond to the three ontologies: Biological Process (BP), Molecular Function (MF), and Cellular Component (CC). Nodes on the DAG correspond to GO terms that are used to annotate genes through the use of annotation databases. Edges are the relationships between terms. The most common relationship is the *"is a"* relationship, but there is also the *"is part"*, *"has part"* and *"regulates"* relationships. The three ontologies are largely unconnected, with the exception of *"regulates"* relationships. Each relationship requires a different form of logical reasoning. These ontologies obey the true path rule in that if a gene is annotated with a particular GO term then its is automatically annotated with all of the parents of that term right to the root of the ontology.

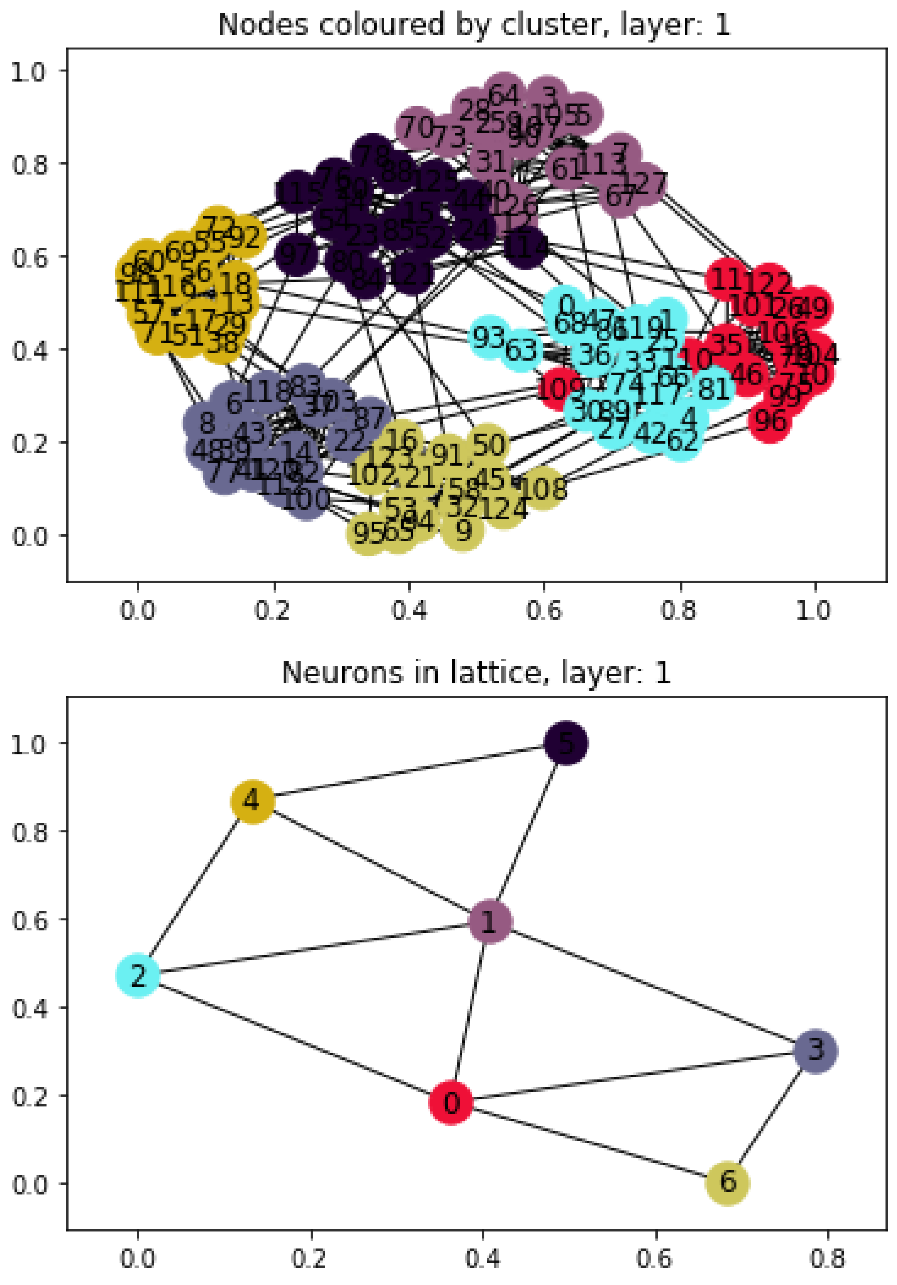
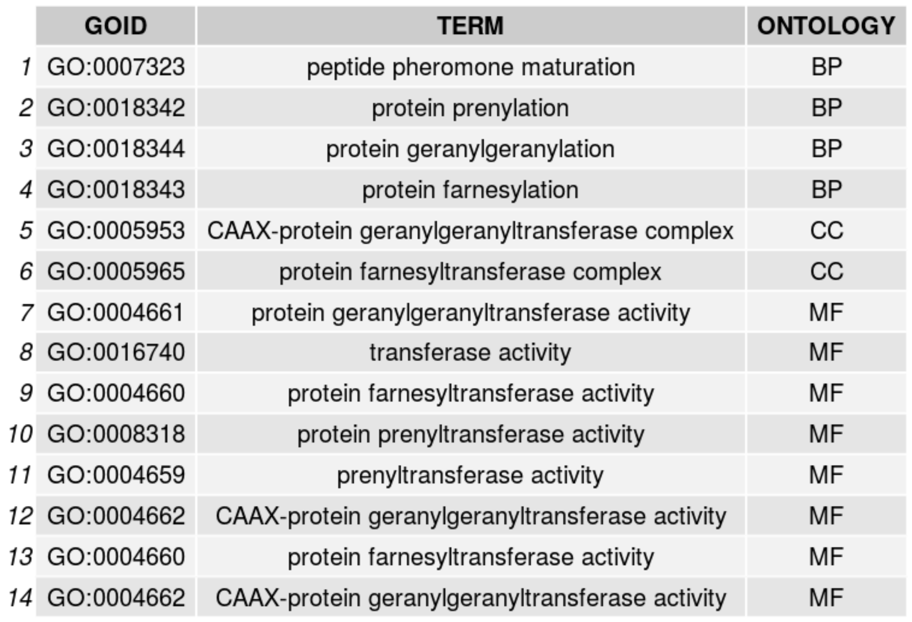


Figure 5.1: Visualisation of simple generated network and resulting map. Nodes are coloured in the network by the neuron in the map that they are assigned to.



	GOID	TERM	ONTOLOGY
1	GO:0007323	peptide pheromone maturation	BP
2	GO:0018342	protein prenylation	BP
3	GO:0018344	protein geranylgeranylation	BP
4	GO:0018343	protein farnesylation	BP
5	GO:0005953	CAAX-protein geranylgeranyltransferase complex	CC
6	GO:0005965	protein farnesyltransferase complex	CC
7	GO:0004661	protein geranylgeranyltransferase activity	MF
8	GO:0016740	transferase activity	MF
9	GO:0004660	protein farnesyltransferase activity	MF
10	GO:0008318	protein prenyltransferase activity	MF
11	GO:0004659	prenyltransferase activity	MF
12	GO:0004662	CAAX-protein geranylgeranyltransferase activity	MF
13	GO:0004660	protein farnesyltransferase activity	MF
14	GO:0004662	CAAX-protein geranylgeranyltransferase activity	MF

Figure 5.2: All GO terms annotated to the gene with OPF identifier YKL019W in the org.Sc.sgd.db annotation database. YKL019W is automatically annotated with the parents of all these terms due to the true path rule of GO.

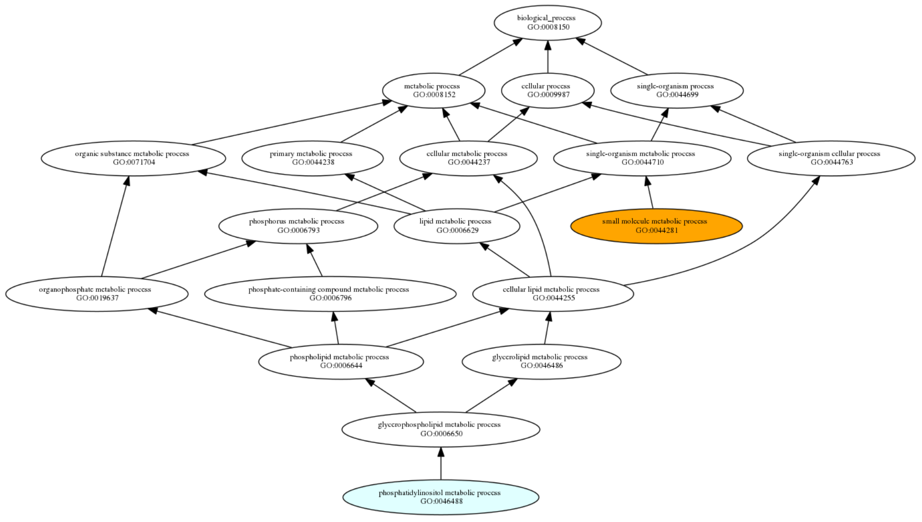

biological_process GO:0008150
metabolic process GO:0008152
cellular process GO:0009987
single-organism process GO:0044699
organic substance metabolic process GO:0071704
primary metabolic process GO:0044238
cellular metabolic process GO:0044237
single-organism metabolic process GO:0044710
single-organism cellular process GO:0044763
phosphorus metabolic process GO:0006793
lipid metabolic process GO:0006629
small molecule metabolic process GO:0044281
organophosphate metabolic process GO:0019637
phosphate-containing compound metabolic process GO:0006796
cellular lipid metabolic process GO:0044255
phospholipid metabolic process GO:0006644
glycerolipid metabolic process GO:0046486
glycerophospholipid metabolic process GO:0006650
phosphatidylinositol metabolic process GO:0046488

Figure 5.3: Relative positions of the terms GO:0044281 *small molecule metabolic process* and GO:0046488 *phosphatidylinositol metabolic process* in the Biological Process (BP) ontology. Graph generated using the web tool G-SESAME [35].

Network name	Number of Nodes	Number of Edges
Uetz Screen	263	292
Y2H Union	1647	2682

Table 5.1: Topology information for the two Saccharomyces Cerevisiae networks. Datasets available from [36, 37].

5.2 Experiments on Saccharomyces Cerevisiae dataset

Two Saccharomyces Cerevisiae (budding yeast) network data sets were used. Table 5.1 details the size of the networks.

After the networks are partitioned into communities by GHSOM, the set of enriched GO terms (p-value < 0.01) for each community was computed, using GO annotations from the org.Sc.sgd.db annotation database. The p-value was computed from a 2x2 contingency table and corresponded to the probability of the observed distribution of genes labelled with a particular GO term occurring from a random sample of genes drawn from the population. Figure 5.5 gives an example of one such table.

Two semantic similarity measures found commonly in the literature were used in this work. Resnik and Wang.

5.2.1 Resnik

The Resnik measure was first used to compute the semantic similarity of words [40]. It involves computing the 'frequency' of a term as

$$p(t) = \frac{|\{t, \text{children of } t\}|}{N} \tag{5.1}$$

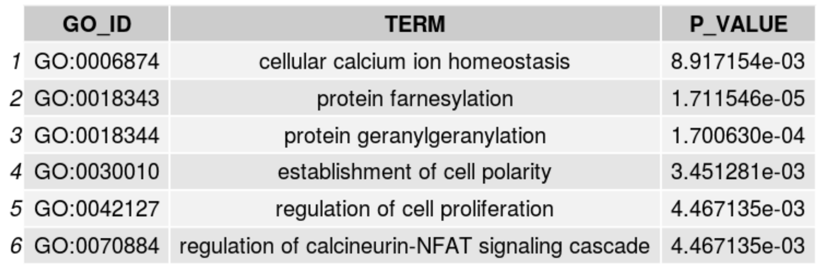
which is used to assign a measure of information content (IC) to each term in the ontology:

$$\text{IC}(t) = -log(p(t)) \tag{5.2}$$

This ensures that terms with few children contain more information.

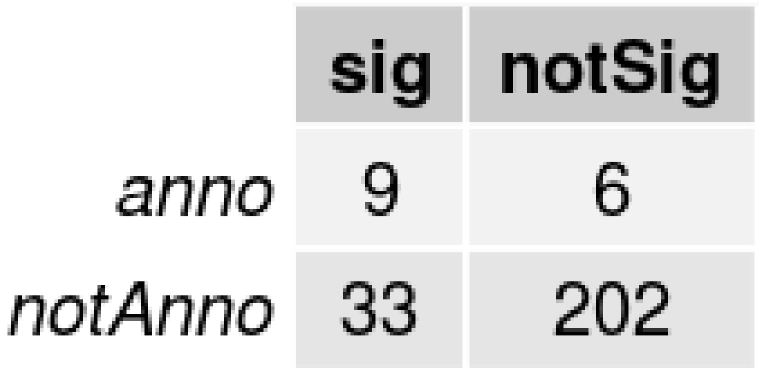
The Resnik measure assigns similarity equal to the maximum information content of a common parent of both terms in in the ontology. Terms that are more related will have common parents further down the tree and these parents will have a higher IC. Sevilla et al. found that "Resnik's measure correlates well with gene expression" [41].

$$\text{sim}_{RESNIK}(t_1, t_2) = \max_{i \in \{\text{parents of } t_1\} \cap \{\text{parents of } t_2\}} (\text{IC}(i)) \tag{5.3}$$



	GO_ID	TERM	P_VALUE
1	GO:0006874	cellular calcium ion homeostasis	8.917154e-03
2	GO:0018343	protein farnesylation	1.711546e-05
3	GO:0018344	protein geranylgeranylation	1.700630e-04
4	GO:0030010	establishment of cell polarity	3.451281e-03
5	GO:0042127	regulation of cell proliferation	4.467135e-03
6	GO:0070884	regulation of calcineurin-NFAT signaling cascade	4.467135e-03

Figure 5.4: Enriched GO terms (p-value < 0.01) in the BP ontology and their respective p-values for the genes with ORF identifiers: YKL019W, YGL155W, KL159C, YCR063W, YBR247C, YDL090C, YOL135C. Table generated using the R package GOSim [38] which is built upon topGO [39].



	sig	notSig
anno	9	6
notAnno	33	202

Figure 5.5: An example contingency table for the go term: GO:0006914 *autophagy* (Fisher's exact test p-value=0.000111024375808973).

To calculate the similarity of two gene sets, the Best Match Average (BMA) method is used. One first finds the enriched GO terms and then computes the pairwise similarity between each term in both sets. The best match for each row and column is summed and divided by the number of rows and columns.

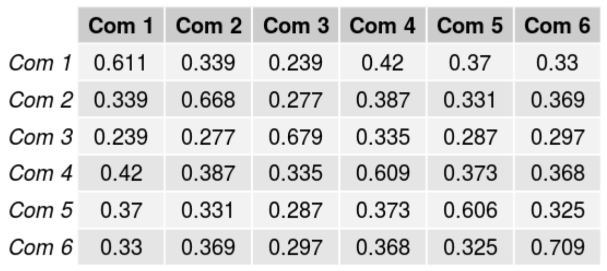
$$\mathrm{sim}_{BMA}(G_1, G_2) = \frac{\sum_{i=1}^{m} \max_{1 \le j \le m} sim(G_{1i}, G_{2j}) + \sum_{j=1}^{n} \max_{1 \le i \le n} sim(G_{1j}, G_{21})}{m+n} \tag{5.4}$$

5.2.2 Wang

The Wang measure is based on the topology of the GO ontology itself [42]. One must first assign weightings w_e to each relation in the ontology. (I use $w_e = 0.8$ for 'is a") and then compute the S-value of every term t in the DAG T_A consisting of the original term A and all of its ancestors, as

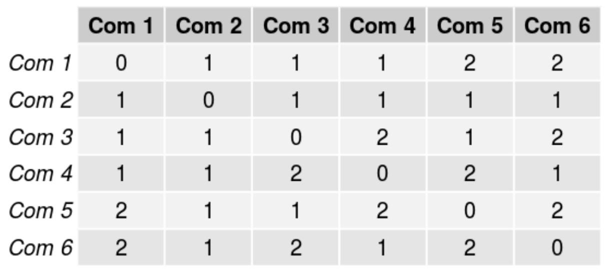
$$S_A(t) = \begin{cases} 1, & \text{if } t = A \\ \max\{w_e * S_A(t') | t' \in \text{children of } t\}, & \text{if } t \neq A \end{cases} \tag{5.5}$$

Then we can compute the semantic value of term A as



	Com 1	Com 2	Com 3	Com 4	Com 5	Com 6
Com 1	0.611	0.339	0.239	0.42	0.37	0.33
Com 2	0.339	0.668	0.277	0.387	0.331	0.369
Com 3	0.239	0.277	0.679	0.335	0.287	0.297
Com 4	0.42	0.387	0.335	0.609	0.373	0.368
Com 5	0.37	0.331	0.287	0.373	0.606	0.325
Com 6	0.33	0.369	0.297	0.368	0.325	0.709

Figure 5.6: Example similarities between the six communities discovered by GH-SOM in the Uetz screen network in the MF ontology using the Resnik similarity measure.



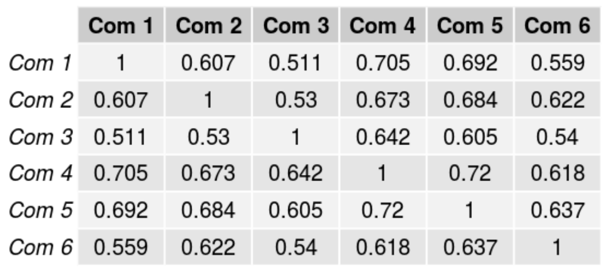
	Com 1	Com 2	Com 3	Com 4	Com 5	Com 6
Com 1	0	1	1	1	2	2
Com 2	1	0	1	1	1	1
Com 3	1	1	0	2	1	2
Com 4	1	1	2	0	2	1
Com 5	2	1	1	2	0	2
Com 6	2	1	2	1	2	0

Figure 5.7: Distances of neurons on the map.

$$\mathrm{SV}(A) = \sum_{t \in T_A} S_A(t) \tag{5.6}$$

Finally, the similarity of two terms is

$$sim_{WANG}(A, B) = \frac{\sum_{t \in T_A \cap T_B} S_A(t) + S_B(t)}{\mathrm{SV}(A) + \mathrm{SV}(B)} \tag{5.7}$$



	Com 1	Com 2	Com 3	Com 4	Com 5	Com 6
Com 1	1	0.607	0.511	0.705	0.692	0.559
Com 2	0.607	1	0.53	0.673	0.684	0.622
Com 3	0.511	0.53	1	0.642	0.605	0.54
Com 4	0.705	0.673	0.642	1	0.72	0.618
Com 5	0.692	0.684	0.605	0.72	1	0.637
Com 6	0.559	0.622	0.54	0.618	0.637	1

Figure 5.8: Example similarities between the six communities discovered by GH-SOM in the Uetz screen network in the MF ontology using the Wang similarity measure.

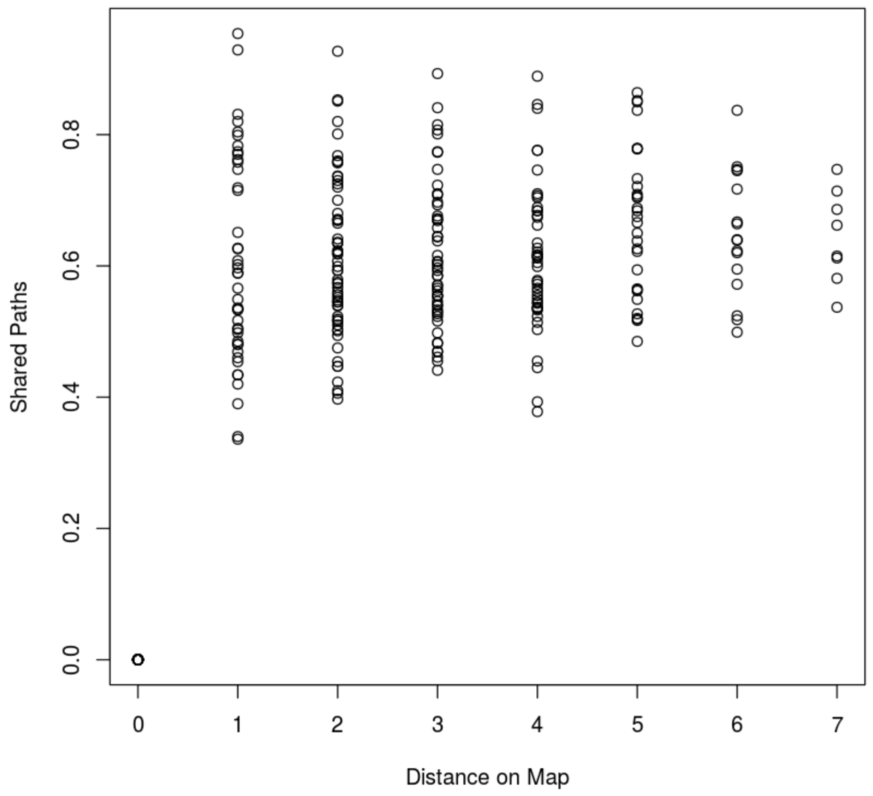


Figure 5.9: Plot of semantic similarity against distance of map using the Wang measure on the Y2H Union dataset.

Chapter 6

Discussion & Future Plans

The aim of my 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. 4.1). Both [5] and [10] 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 [43] cites [13] who identified that MDS can allow for clusters and communities within a graph to be more easily identifiable even before a cover is found.

The semantic similarity measures have not yet shown any correlation between community distance on the map (see fig. 5.9 for an example) and I would like to spend a little more time investigating this. I would also like to investigate the similarity of neighbouring communities in social networks.

Another area I would like to investigate before completing this work is to embed the genes based on semantic similarity with each other (rather than shortest path distance on the network) and explore the clusters discovered by GHSOM in this case. Since the embeddings are not based on topology, there is not guarantee that neighbouring nodes on the graph will be embedded together and so the clusters may not represent true communities but functional clusters.

At the end of my work on GHSOM, I would like to continue working on topographic feature selection, but using a semi-restricted Boltzmann machine, with interlayer links on the hidden layer to uncover the relationships between the learned features. I have taken some time to implement a working RBM that runs efficiently on the GPU as a starting point for my experimentation with this. In general, I would like to build some generative model to infer the underlying features of the network, and use this model to uncover more powerful features than features based on topology alone.

Bibliography

[1] Albert-László Barabási. Scale-free networks: a decade and beyond. *science*, 325(5939):412–413, 2009.

[2] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] Geoffrey E Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[5] 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.

[6] 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.

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

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

[9] 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.

[10] 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.

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

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

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

[14] 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.

[15] 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.

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

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

[18] 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.

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

[20] 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.

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

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

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

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

[25] 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.

[26] Michael T Schaub, Jean-Charles Delvenne, Martin Rosvall, and Renaud Lambiotte. The many facets of community detection in complex networks. *arXiv preprint arXiv:1611.07769*, 2016.

[27] 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.

[28] Michael Dittenbach, Andreas Rauber, and Dieter Merkl. Uncovering hierarchical structure in data using the growing hierarchical self-organizing map. *Neurocomputing*, 48(1):199–216, 2002.

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

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

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

[32] 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.

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

[34] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler, J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al. Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.

[35] Zhidian Du, Lin Li, Chin-Fu Chen, S Yu Philip, and James Z Wang. G-sesame: web tools for go-term-based gene similarity analysis and knowledge discovery. *Nucleic acids research*, page gkp463, 2009.

[36] Uetz screen dataset. `http://interactome.dfci.harvard.edu/S_cerevisiae/download/Uetz_screen.txt`.

[37] Y2h union dataset. `http://interactome.dfci.harvard.edu/S_cerevisiae/download/Y2H_union.txt`.

[38] Holger Fröhlich, Nora Speer, Annemarie Poustka, and Tim Beißbarth. Gosim–an r-package for computation of information theoretic go similarities between terms and gene products. *BMC bioinformatics*, 8(1):166, 2007.

[39] Adrian Alexa and Jorg Rahnenfuhrer. topgo: enrichment analysis for gene ontology. *R package version*, 2(0), 2010.

[40] Philip Resnik et al. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *J. Artif. Intell. Res.(JAIR)*, 11:95–130, 1999.

[41] Jose L Sevilla, Victor Segura, Adam Podhorski, Elizabeth Guruceaga, Jose M Mato, Luis A Martinez-Cruz, Fernando J Corrales, and Angel Rubio. Correlation between gene expression and go semantic similarity. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB)*, 2(4):330–338, 2005.

[42] James Z Wang, Zhidian Du, Rapeeporn Payattakool, S Yu Philip, and Chin-Fu Chen. A new method to measure the semantic similarity of go terms. *Bioinformatics*, 23(10):1274–1281, 2007.

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