Computing Science Group

Machine Learning and Genetic Regulatory Networks: A Review and a Roadmap

Christopher Fogelberg, Vasile Palade

CS-RR-08-04

Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford, OX1 3QD

Abstract

Genetic regulatory networks are large graphical structures and their inference is a central problem in bioinformatics. However, because of the paucity of the training data and its noisiness, machine learning is essential to good and tractable inference.

This literature review first surveys the relevant theoretical and empirical biochemistry. Next it describes the two types of GRN inference that are problems, the data which can be used for machine learning, and how different kinds of machine learning have been used in previous research. The survey concludes with an analysis of the field as a whole, some underlying methodological issues and a few possible areas for future research.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Inference of the network structure and parameters which cause the observed gene expressions and phenotypic states. This is the problem of *genetic regulatory network* (GRN) inference; how machine learning can be applied to it is the focus of this technical report. The technical report also surveys the underlying biology and the data which is available for inference.

Section 2 begins by summarising the biology which underlies the statistical and machine learning problems in the field of network inference. Following that, section 3 provides a brief overview of these problems.

Sections 4 and 5 discuss available data and describe existing approaches to *network inference*. Section 6 describes important and more general statistical concerns associated with the problem of inference, and section 7 provides a brief visual categorisation of the research in the field. Notable aspects of the problems described in section 3 are highlighted in section 8, which concludes the survey.

The key mathematical variables used in this survey are as follows. i, j and h refer to genes and to their *expression level*: which is intended will be clear from the context. k_i is i's fan-in factor, the number of genes it is regulated by. k is also used to refer to a gene's fan-out factor, the number of genes it regulates. When it is unclear what k refers to it is annotated k_{in} or k_{out}. ϕ_i are the external, non-genetic regulators of the gene i. N is the number of genes. Sometimes it is also used to refer to the set of all genes, or their expression levels. M is sometimes used to refer to the number and set of *samples*. A–C are used to refer to arbitrary sets, the size of the set, random variables and probability distributions. Arbitrary sets of functions are denoted by F. If a capital letter is a set, then the corresponding lower case letter is a member of that set. x, y and z are unit-less algebraic variables. Time is indexed using t, and prime (" ' ") is used to mean "later".

Other reviews of GRN include [23], [17] and [24]. However, many of these are dated. Those that are more current focus on presenting new research findings and do not summarising the field as a whole.

2 The Underlying Biology

This section clarifies the terms *regulatory network* and genetic regulatory network, and it briefly summarises the underlying cellular biology.

Regulatory network is a term used to describe causal interaction. Examples of regulatory networks include protein-protein networks, metabolic processes[77] and the inhibitory or excitatory relationship one gene may have on another[6]. This latter kind of network is known as a genetic regulatory network and they are the focus of this review. Historically, genetic interaction was not considered[7], but over the past few decades[51; 57] researchers have increasingly understood the importance of the regulatory relationships.

2.1 Network Structure and Macro Characteristics

This subsection discusses the known and hypothesised large scale characteristics and network structure of GRNs. Subsection 2.2 describes the single-gene level characteristics of a GRN. Intuitively, GRN can be understood to be *messily robust* as a consequence of their evolutionary history[108].

A GRN is a graph, the vertices of this graph are genes and the edges describe the regulatory relationships between genes. GRN may be modeled as either directed[102] or undirected[114] graphs, however the true underlying regulatory network is a directed graph. Recent[6] and historical[55] research shows that GRN are not just random directed graphs. [6] discusses other macro-statistical features of GRN.

2.1.1 The Out-degree (k_{out}) and In-degree (k_{in})

GRN network structure is neither random nor rigidly hierarchical. Instead, GRNs appear to be *scale free*. This means that the probability distribution for the out-degree follows a power law[6; 55]. I.e., the probability that a gene i regulates k other genes is $p(k) \approx k^{-\lambda}$, where λ is usually some value in the range 2–3. Kauffman's analysis of scale free Boolean networks shows that they behave as if they are on the cusp of being highly ordered and totally chaotic. It is claimed that being on this cusp contributes to the evolvability and adaptability of GRN. The in-degree follows an exponential distribution[6].

These distributions over k_{in} and k_{out} means that a number of assumptions have been made in previous research to simplify the problem and make it more tractable.

For example, assuming that an "average" gene is regulated by only 2 or 3 others and that it probably regulates a similar or slightly smaller number of genes itself is reasonable in most cases. In higher metazoa k_{in} is more likely to be in the range 4–8[63]. Crucially, this average is *not* a maximum. Many learning algorithms are exponential in k_{in}^{max}.

This means that models which strictly limit the number of regulatory genes k_{in} to some arbitrary constant (such as [102; 110]) may not be able to infer all networks. This compromises their explanatory value.

[110] "suggest[s] a sparse topology[,] because the maximal number of inputs (k_{in}) to a unit is $k_{in} \ll N$". In this work, except in the case of a very small and well known GRN of ten genes, k_{in}^{max} was restricted to 3. While such a tight constraint is true in the vast majority of cases, in a number of crucial situations it is not. For example, a gene which regulates a *module* may regulate hundreds of genes. Sparse topologies are inadequate in the general case.

2.1.2 Modules

Genes are organised into modules. A module is a group of genes which are functionally linked by their phenotypic effects. This phenotypic link may be temporal, or because they all contribute simultaneously to some complex activity[6]. Examples of phenotypic effects include protein folding, cell growth regulation[98], glycolysis metabolism[97], and the metabolism of amino acids[5]. As a consequence of natural selection, genes in the same module are often also physically proximate, and *co-regulated*, even *equi-regulated*. A gene which is in multiple modules is often regulated by different genes for each module[6; 53].

Furthermore, one or two genes may be the main regulators of all or most of the other genes in the module. It is crucial to take these "hub" genes into account to ensure a general and veridical model.

Genetic networks are enormously redundant. For example, the `Per1`, `Per2` and `Per3` genes are shared by a very wide range of species and help regulate circadian oscillations. Knocking out one or even two of them produces no detectable changes in the organism[57]. This redundancy is an expected consequence of evolution[6]. Hub genes create essential regulatory differences and drive the viability of the organism. Identifying them and their regulatory drivers in turn is a key step in the inference of large, multi-functional GRN. If the modular nature of GRN is not considered the result may lack robustness and biological meaning[98].

Known modules range in size from 10 or 15 to several hundred genes, and have no characteristic size[6]. A gene can also be a member of several modules. Methods which do not take this into account (e.g. [97]) may sacrifice robustness and biological meaning.

2.1.3 Motifs

This subsubsection discusses motifs. A motif is a sub-graph in a GRN which is repeated more times than would be expected if the network was randomly constructed, given the GRN's distributions over k[57]. For example, the feed-forward triangle shown in figure 1 is an atypically common motif in gene regulatory networks. This pattern frequently occurs with module regulatory genes, where one module regulatory gene may bind to another and then both may contribute to the module's regulation[5].

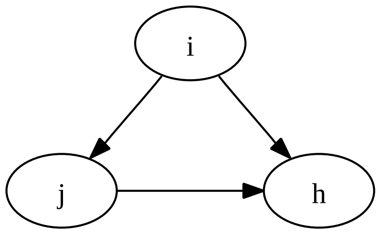


Figure 1: The feed-forward motif, as discussed in [79].

Like modules, motifs often overlap. In addition, they are highly conserved evolutionarily[16; 46]. This means that phylogenetics could provide useful insights in certain circumstances. However the converse is not necessarily true. Motifs like the feed-forward triangle may be the result of convergent evolution towards a robust genetic regulatory network. Their existence in two species does not necessarily denote a close phylogenetic relationship.

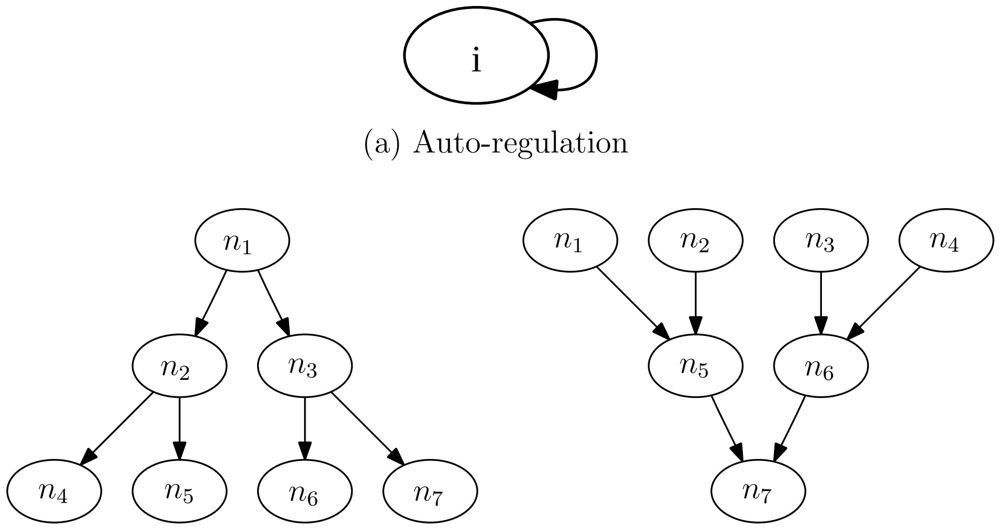
Another common motif is self-inhibition[22]. This is known as *auto-regulation*. Two more are the *cascade* and *convergent* motifs. Each of these three is illustrated in figure 2. The biases in network structures that motifs represent can be used to guide, describe and evaluate network inference.

2.2 Gene-Gene Interactions

Subsection 2.1 described the graph-level statistical properties of GRN. This subsection looks at the precise way in which two genes can interact: The gene i regulates the gene j, but how? Crudely, i may either *up-regulate* or *down-regulate* j. That is, if i regulates j then i is either *excitatory* or *inhibitory* with respect to j. Different formalisations model this *regulatory function* to different degrees of fidelity.

2.2.1 One-to-One Regulatory Functions

Regulatory links can have different strengths. i may be a crucial, very strong regulator of j, or it may be almost irrelevant. Imagine that i is the only gene



(a) Auto-regulation

(b) Cascade

(c) Convergence

Figure 2: The auto-regulatory, cascade and convergence motifs.

which regulates j[1]. Also consider non-genetic influences on j, denoted ϕ_j. In this situation, $j' = f_j(i, \phi_j)$. Inter-cellular signaling is modeled in [74] and is one example of ϕ. In many circumstances we can assume that $\frac{\delta f}{\delta \phi} = 0$. In that case $j = f_j(i)$. The nature of f_j is highly varied. It can be roughly linear, sigmoid or a piecewise threshold function[118].

It is also possible for one gene to both up-regulate and down-regulate another gene. For example, i might actively up-regulate j when i is low, but down-regulate it otherwise. However, the chemical process underlying this is not immediately clear, and in the models inferred in [91] a previously postulated case of this was not verified. In any case, this kind of especially complex relationship is not evolutionarily robust. For that reason it will be relatively rare.

Wider properties of the organism also have an influence on the kinds of regulatory functions that are present. For example, inhibitors are more common in prokaryotes than in eukaryotes[45].

[1]Not unknown, but relatively rare. There are two types of cell, prokaryotes and eukaryotes. All multi-cellular life is eukaryotic, most bacteria are prokaryotic. Most prokaryotic genes are regulated by 2–3 others[6], for eukaryotes $\overline{k}$ tends to be even greater[23]

Figure 3: The chromosomes of a healthy human. The chromosomes have each been condensed from a long thin strand into a highly ordered structure. Image sourced from [88].

2.2.2 Many-to-One Regulatory Functions

If a gene is regulated by more than one gene its regulatory function is usually much more complex. In particular, eukaryotic gene regulation can be enormously complex[26]. Consider the regulatory network shown in figure 2(c). The extent to which n_1 regulates n_5 may depend on how strongly n_1 is expressed *and* on the expression level of n_2.

This complexity arises because of the indirect, many-leveled and complex multi-stage process underlying gene regulation. This regulatory process is summarised in sources such as [17; 24; 118]. The relevant aspects are covered in this and the next subsubsection.

As [98] illustrates, a regulatory function may be a *conditional piecewise function*, i.e. i regulates h if and only if j is expressed strongly enough, however apart from this effect j has no effect on h. Alternatively, both i and j may up-regulate h in a non-additive manner.

However, some of the logically possibly regulatory relationships appear to be unlikely. For example, it appears that the *exclusive or* and *equivalent* relationships are biologically and statistically unlikely[67]. Furthermore, [55] suggests that many regulatory functions are *canalised.* A canalised[108] regulatory function is a function that is buffered and depends almost entirely on the expression level of just one other gene.

2.2.3 The Gene Transcription Process

Consider i's regulation of j. This process has been illustrated in figure 4, which is based on figures in [17] and [24]. The first regulatory step is the transcription of i's DNA into RNA. The creation of this fragment of RNA, known as *messenger RNA* (mRNA), is initiated when *promoters* (proteins that regulate transcription) bind ahead of the start site of the gene. The resulting fragment of RNA is the genetic inverse of the original DNA (i.e. an A in the DNA is transcribed as a T in the RNA). Following this the mRNA is translated into a protein.

The transcribed protein causes phenotypic effects, such as DNA repair or cellular signaling. In addition, some proteins act as promoters, and in this case the expression level of j is regulated when protein from i binds to promoter regions ahead of the start of j. Note that the term "motifs" is also used to refer to binding sites, to describe the kinds of promoters which can bind with and regulate a particular gene. For clarity, the term is not used in this way in this report. For example, we describe the work done in [92] as using prior knowledge of protein-binding site relationships, rather than as using prior knowledge of motifs.

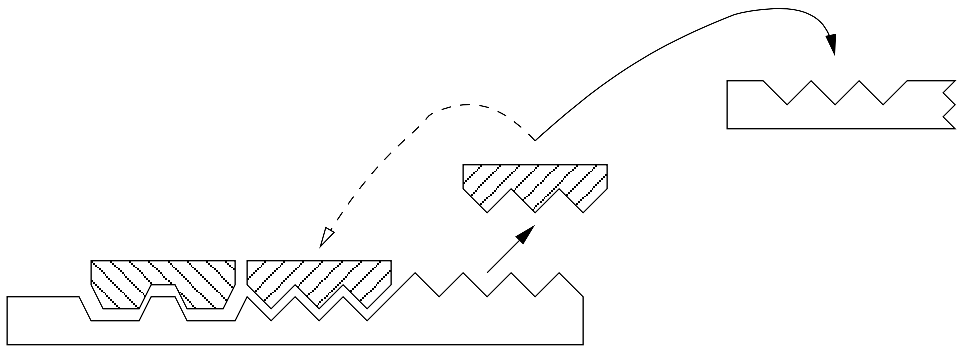


Figure 4: A protein eye's view of gene regulation. The gene displayed in the figure is regulated by itself and one other gene. We have omitted the mRNA $\longrightarrow$ protein translation step for the sake of clarity. A just-transcribed protein is shown in the figure as well. Right now it cannot bind to the gene which transcribed it, as that site is occupied. It may bind to the gene on the right hand side of the figure, or it may go on to have a more direct phenotypic effect.

To summarise: A GRN is a stochastic system of discrete components. However modeling a GRN in this way is not tractable. For that reason we do not consider stochastic systems in this survey. Instead, a continuous model of

GRN is used. In this model a GRN is a set of genes N and a set of functions F such that there is one function for each gene: $\forall n \in N$, $\exists f_n : f_n \in F$). Each of these functions would take all or a subset of N as parameters, and ϕ_n as well. Using this sort of model the important features of the regulatory relationships can be inferred and represented.

3 Outstanding Problems

There are two main types of problems which machine inference can be applied to for GRN: inferring epistasis and inferring whole-network structure and regulatory relationships. This section briefly discusses the inference of epistasis first.

An epistatic interaction is an interaction between two genes, often unexpected, which leads to changes in an organism. *Synthetic lethality* and *yeast two-hybrid* (Y2H) are examples of experiments which can help to reveal epistatic interactions. However because data gathered from these experiments is often very noisy and because a particular phenotypic change may not occur unless several genes have been perturbed, there is substantial scope for the use of machine learning techniques in the inference of epistasis. Two recent examples of this are [82; 94]. Synthetic lethality and other perturbations are discussed in more depth in subsection 4.2.

The second major type of problem is the inference of the underlying genetic regulatory network itself. Epistatic problems are an important subset of this more general inferential problem. Because researchers can only infer models of the underlying stochastic system, F can take on an enormous number of forms. The following section, section 5 describes some of these in detail, but they are introduced here in the context of the specific whole-network problems they address.

At the simplest level, F may be no more than a set of membership functions which group genes into clusters. In some work, the regulatory genes of those clusters have also been identified. Examples of clustering include [98] ,[47] and [41]. Such models can be inferred by clustering co-expressed or apparently co-regulated genes, although much more involved algorithms (e.g. [5]) incorporating other forms of data have also been used.

Such module-level analysis only gives an overview of the network, as one gene may be in several modules and may be regulated by several other genes, not all of which are necessarily module regulators[23]. However, the low-fidelity network that clustering provides can still answer important biological questions and suggest further, more detailed research.

At the other extreme, F may be a set of mathematical functions which relate concentrations of mRNA and other chemicals to gene expression levels. For example, "If the expression level of i rises above x then j is transcribed at the rate $y(i-\alpha x)$". When this level of detail is required, *ordinary differential equations*[9] are normally used, although partial differential equations[91] and logical networks [110; 111] have also been used. Accurate inference of such detailed $f \in F$ requires large, relatively noiseless M and can be computationally intractable for even moderate N.

Other approaches provide intermediately detailed models on intermediate numbers of genes. Interestingly, there appears to have been only one example[47; 114] of research which combines several models in a principled manner, although it is discussed theoretically by D'haeseleer et al. [23]. This is surprising as it is possible that several detailed models or one general and several detailed models could be combined to give a more complete, more detailed and more veridical network.

To summarise, the problems of epistatic and general network inference are distinct but related. What distinguishes them is the relationship between the output of the inferential process and the GRN model that is used. In epistatic problems the model is instrumental and does not need to accurately reflect the actual causal network, so long as it makes accurate predictions. However in the case of network inference the inferred network itself is the output. In this case, the inferred network should model the GRN as closely as possible.

The distinction between these two kinds of problem is frequently not made clear, and failing to do so makes some research reports more difficult to understand.

4 Available Data

To address bioinformatic network problems there are four types of data available. These are:

- Expression data
- Perturbation data
- Phylogenetic data
- Chemical and gene location data

This section goes through each of these and discusses their accessibility and utility. In some situations, multiple kinds of data are used, e.g. [5; 42; 121], however this usually makes the inference more time and space complex.

4.1 Expression Data

Expression data measures how active each $n \in N$ is. As transcription activity cannot be measured directly, the concentration of mRNA (which is ephemeral) is used as a proxy.

Because regulatory or phenotypic protein-protein interactions after transcription can consume some of the mRNA before it can regulate another gene[98] this may seem to be an inaccurate measure of gene activity[97]. Furthermore, a protein may bind to a promoter region but actually have no regulatory effect[42].

In addition, most genes are not involved in most cellular processes[53]. This means that many of the genes sampled may appear to vary randomly.

However, if the data set is comprehensive and we are just concerned with inference of the regulatory relationships these influences are not important. Sufficient data or targeted inference obviates the problem of irrelevant genes. Non-genetic, unmodeled influences are analogous to hidden intermediate variables in a *Bayesian network*[42] (BN, subsection 5.4) whose only parent is the regulatory gene and whose only child is the regulated gene. An influence like this does not distort the regulatory relationships or predictive accuracy of the model.

Tegner et al.'s successful inference of a network with known post-transcription regulation and protein-protein interactions using only (perturbed, subsection ref 4.2) gene expression data[110] provides evidence of this fact.

4.1.1 Types of Expression Data

There are two kinds of expression data, and both have been used for network inference.

Equilibrium expression levels in some environmental situation are the most easily obtained form of expression data, and *microarrays* are the most common way of gathering it. A microarray is a pre-prepared slide, divided into cells, one for each gene we are interested in. Each cell is individually coated with a chemical which fluoresces when it is mixed with the mRNA generated by just one of the genes. The brightness of each cell is used as a measurement of the level of mRNA and therefore of the gene's expression level. This measurement of brightness is relatively noisy. In addition, entire rows or columns are sometimes lost. As well as being *technically noisy*, microarrays represent a stochastic system with a continuous value. Thus they are also vulnerable to *biological noise*[87]: random stochastic fluctuations. However, the magnitude and impact of the noise is hotly debated. Recent research (2007) argues that it has been "gravely exaggerated"[58].

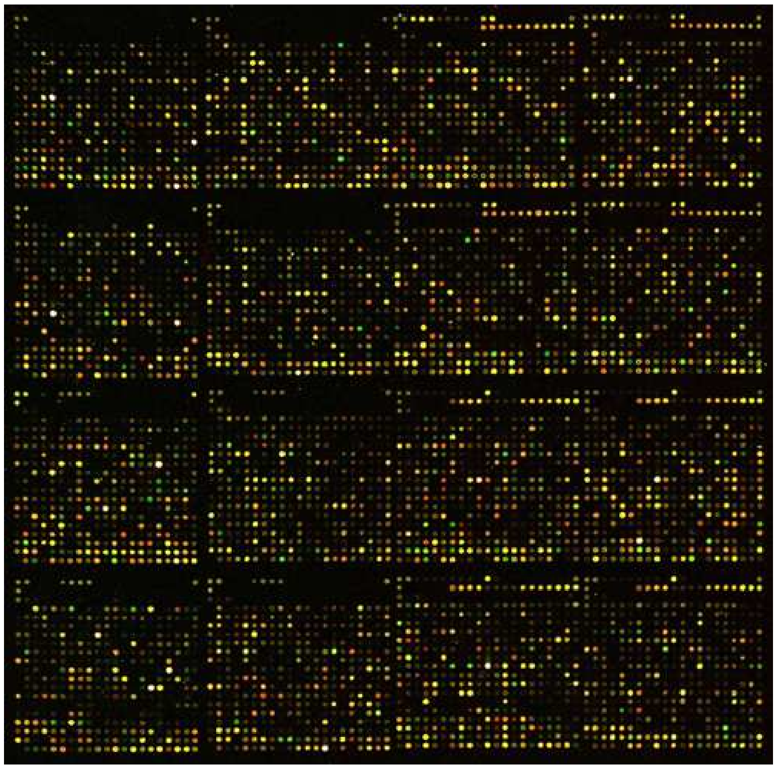


Figure 5: An example of a microarray. Each gene is represented by one dot, and the brightness of the dot is the mRNA concentration. Brightness can be difficult to measure accurately. Sometimes entire rows or columns may be corrupted as well. Image sourced from [50].

Some examples of research on inference which use this kind of data include [2; 14; 25; 48; 56; 66; 72; 83; 97–99; 101; 106; 107; 109; 113; 114; 127]. Wang et al.'s [119] work is particularly interesting as it describes how microarrays of the same gene collected in different situations can be combined into a single, larger data set.

Time series expression data is a sequence of gene expression data points gathered over the course of some phenotypic process, e.g. the cell division cycle[106]. The time-ordering places greater causal restrictions on the regulatory relationships than equilibrium data does. This means that fewer data points are necessary.

In addition, time series can allow more precise inference of the regulatory function between two genes. For this to be done reliably it is important that the data is as noise-free as possible[61].

Unfortunately, collecting this kind of data is difficult and it has only recently become available in the quantities necessary for research. [63] is one example of research which uses biological data. More commonly, the data is simulated. Examples of work that groups have done using time series data include [118], [91], [102] and [52; 104; 105; 123; 124]. Research on accurately simulating expression data includes [3; 8; 17; 87; 104].

Typically, expression data is normalised and preprocessed prior to being used. This is because all that matters in GRN inference is a a gene's level of expression relative to its own range.

New technologies[24] are promising to deliver more and cleaner expression data in the future.

4.2 Perturbation Data

Perturbation data is very similar to time series gene expression data. It consists of either the expression levels of the genes after they have reached a new equilibrium which takes into account the external artificial perturbation of one or more genes, or the time series of the genes as they come to this new equilibrium. Directly making known adjustments to some of the genes means that more information is included in the data.

This is because perturbation data includes a causal arrow. If we inhibit i and j increases then, loosely, we can infer that i inhibits j. In actual fact, i may not directly inhibit j; it may be that i and j have redundant functions and that j only increased because there was not enough i in the cell, rather than because i binds to j and inhibits it. As discussed in subsection 4.1, this kind of indirect regulatory effect is not important in an accurate GRN model. Effectively, i does regulate j, even if the actual mechanism is through some complicated chain of proxies. Exploitation of the causal arrow leads to much more efficient algorithms, e.g. [61].

Synthetic lethality[36; 68; 115] is also commonly used in biological research. In a synthetic lethality experiment each pair of genes is knocked out and the effect on the organism is measured and can be compared with the organisms growth when only one of the genes in each pair is knocked out. Although it is not practical to knock out all possible pairs (there are approximately 40 million such pairs in yeast), genes with known interaction or mutual redundancy can be selected using prior knowledge.

If the organism's survival or growth rate is unchanged in the pair-situation when compared with the single knockout situations then we can conclude that the gene's do not interact phenotypically in the environmental situation that the experiment tests. However the pair knockout organism's growth may be changed non-multiplicatively and substantially, compared to the effects of either of the single gene knockouts. The organism may even become completely inviable. Such a result indicates gene interaction.

Y2H data is gathered in a broadly similar manner.

Examples of experiments using perturbation data include Driscoll and Gardner [24], Kyoda et al. [61], Tegner et al. [110] and Gardner et al. [34].

4.3 Phylogenetic Data

Phylogenetics is the study of species' evolutionary relationships to each other. For example, the two species of chimpanzee, *Pan troglodytes* and *Pan paniscus*, are more closely related to each other than they are to humans, *Homo sapiens*.

To date, very little work has been carried out which directly uses phylogenetics and the principle of phylogenetic conservation[16; 46] to identify regulatory relationships *de novo*. This is because phylogenetic data is not sufficiently quantified or numerous enough. However, as subsection 4.4 discusses, this sort of information can be valuable in validating results obtained using other methods. As [59] notes, transcriptional promoters tend to evolve phylogenetically, and as research by Pritsker et al. illustrates, regulatory relationships in species of yeast are often conserved[92]. [26] reaches similar conclusions, arguing that the "evolution of gene regulation underpins many of the differences between species".

4.4 Chemical and Gene Location Data

As discussed in subsection 4.3, the utility of non-expression data comes from combining it with other forms of data. Most types of chemical and gene location data are useful like this. This subsection discusses these types of data. Because so many types of data can be used it is difficult to draw general conclusions. Instead we briefly discuss a number of recent approaches. This will give readers an idea of the kinds of inference which can be performed.

The first of these examples[121] presents a methodology and applies it to the yeast *Saccharomyces cerevisiae* so that protein-protein interactions (the *protein network*) could be inferred and understood in more depth than just synthetic lethality allowed.

Gene expression data was obtained from microarrays, as described in subsection 4.1. Information about which pairs of proteins interacted was obtained using several Y2H experiments, and phylogenetic profiles from 145 other species were used. In addition, the cell was divided up into 23 functional spatial components and the presence or absence of each protein in each compartment recorded using a 23 bit vector for each protein.

Statistical techniques were used to combine the data types, and the ability of standard unsupervised techniques to infer the network were compared with the results when certain known protein-protein interactions in yeast were fixed as definitely occurring in the protein network before inference began. Neither approach did well, and the number of false positives began to swamp the true positive results very quickly as the sensitivity was increased. However, the supervised approach was able to slow this rapid decrease in specificity.

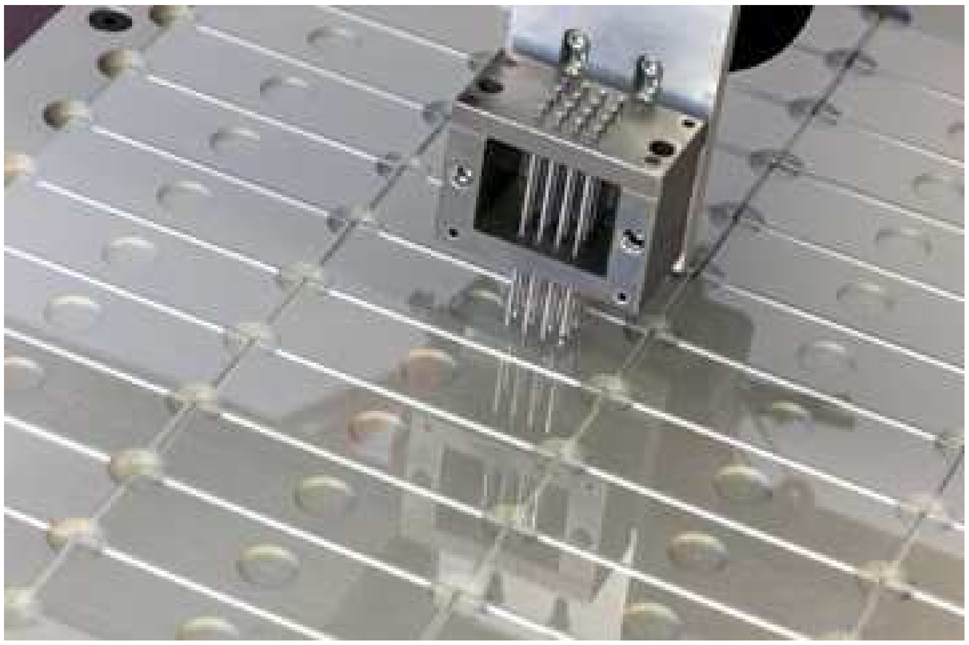


Figure 6: A microarray being constructed. Robotically controlled pins place the microscopic fragments on DNA onto the glass slide. Image sourced from [62].

A second example can be seen in the work done by [42]. This approach is broadly similar in its structure to that of [121], except that it is applied directly to the inference of the underlying GRN. The authors used a chromatin immunoprecipitation (ChIP) assay to discover some genes in *S. cerevisiae* which regulated one another. Then they compared the BNs inferred when this prior data was available and when it was not.

The final results presented were calculated using an approximate integration of the posterior distribution over graphs. Although the ChIP assays can be noisy, the authors forced the regulatory relationships indicated by these to be present in the graphs they searched over. However, they do discuss altering the prior probability of edges suggested by assays or other sources of information instead, so that the search is not incorrectly constrained.

A third example is described in [41]. Here, Harbison et al. combined microarrays of the entire genome and phylogenetic insights from four related species of yeast (*Saccharomyces*). Given 203 known regulatory genes and their transcription factors, the researchers were able to discover the genes that these factors acted as regulators for. This research is another example of the type of approach developed in [121].

These examples illustrate two key features of research which use chemical or phylogenetic data and, more generally, research which combines multiple kinds of data. Firstly, research using multiple kinds of data is normally based on gene expression data, and uses the other kinds of data (formally or informally) to provide a prior of some kind.

Secondly, although some research[117; 121] has been done on how multiple types of data can be combined and used in a general and principled manner, almost all examples of this kind of research aims to answer specific questions about a specific species (that is already well known, e.g. yeast) in a limited number of situations, rather than to develop more general methods.

Extracting as much information as you can when doing model inference is very important. As [97] points out, one way of doing this is by using many types of data and more of it. However, from a Computer Science research perspective this fact doesn't suggest just using more data. Instead it suggests developing better approaches and taking advantage of techniques such as multi-classifiers[93] and fuzzy set theory[11] to maximise the information extracted.

5 Approaches to GRN Inference

Having discussed the underlying biology, the bioinformatic problems which result and the data that can be used to approach these problems, this section reviews different types of approach to network inference.

Subsection 5.1 discusses gene clustering. Clustering is done in an effort to reveal the modular structure of the network; sometimes it can also be used as a preprocessor or to discover the major regulatory genes of each module. Following that, subsections 5.2 and 5.3 discuss discuss approaches based on logical networks and differential equations. Subsection 5.3 also describes other functional approaches to the problem of network inference. Finally, Bayesian networks are evaluated as a way of approaching the problem of GRN inference in subsection 5.4. Although a number of other kinds of approach exist, including models which directly reflect the stochastic nature of the underlying true GRN[17], the approaches we summarise are broad enough that we can also usefully discuss common issues of efficiency and tractability in subsection 6.1.

5.1 Clustering

As described in section 2, genes are functionally organised into a modular regulatory structure[5; 47]. Clustering[120] can reveal this modular structure, with applications in data preprocessing and also to guide future biological experiments.

This subsection discusses distance measures and clustering methods first. Following that it gives a number of examples which use clustering, including the use of clustering to preprocess data. Subsubsection 5.1.5 summarises *biclustering*.

5.1.1 Overview

A clustering algorithm is made up of two elements: the *method*, and the *distance measure*. The distance measure is how the similarity (difference) of any two data points is calculated, and the method determines how data points are grouped into clusters based on their similarity to (difference from) each other. Any distance measure can be used with any method.

Primarily, clustering algorithms have been used on expression data, just as expression data has been the source of training data for most other approaches to GRN inference. This is because expression data is usually the only form of data available in the quantities necessary to support reliable machine inference. However data from a range of other approaches, such as ChIP assays[98], is increasingly available in these quantities.

The data is often normalised based on its range, mean values and/or standard deviation. Non-linear transformations[23] can also be used to help spread the data out in such a way that the clusters are more distinguishable.

5.1.2 Distance Measures

The simplest useful distance measure is Euclidean distance. The Euclidean distance between the genes i and j for which there are M pairs of expression levels is shown in equation 1. The *Manhattan distance*[60], also known as the taxicab distance, is similar to the Euclidean distance.

$$d_{ij} = d_{ji} = \sqrt{\sum_{m=1}^{M} (i_m - j_m)^2} \tag{1}$$

Euclidean distance is symmetric. I.e. $d_{ij} = d_{ji}$. Not all distance measures are symmetric. Typically, a gene's distance from a cluster is its average distance from each member of the cluster and the clustering method aims to minimise each gene's average distance. Other ways of measuring the distance between a cluster and a sample include the median, maximum and minimum distance from a member of the cluster.

Mutual information (MI) is closely related to the Shannon entropy[100]. It is a measurement of how much you learn about i when you know j. Equally, it is a measure of how much uncertainty there is between i and j. Just as with Euclidean distance, mutual information is symmetric. Mutual information is expressed in terms of probability, and its calculation for continuous expression levels (and assuming some distribution over the expression levels) is shown in equation 2.

$$I(X;Y) = \int_Y \int_X p_{xy}(x,y) \log\left(\frac{p_{xy}(x,y)}{p_x(x)p_y(y)}\right) dx\, dy \tag{2}$$

When using MI the clustering method minimises the uncertainty by grouping together genes which are strongly informative about each others' expression levels. Consider the genes i and j. If i and j are highly correlated a good mutual information clustering clusters them together. This is because if you already know i and then discover j you do not learn much new information.

As a final example, we summarise the Mahalanobis distance[18; 73]. The Mahalanobis distance addresses a weakness in Euclidean distance and some other measures. To understand the weakness it addresses it is important to distinguish between the real module or cluster underlying the gene expression, and the apparent cluster which an algorithm infers. [20] has a more in depth discussion of this distinction, however it is essential to remember that the clustering algorithm does not necessarily reveal true or functional clusters in the data.

Imagine that we are using the Euclidean distance and that the samples we have of genes in the underlying "real" clusters C and D are biased samples of those clusters. Assume that the method is clustering the gene h, and that h is truly in D. However, because the samples of C and D are biased it will be clustered into C. Having been clustered into C it will bias future genes towards C even more. Because microarrays are done on genes and phenotypic situations of known interest this bias is possible and may be common.

In summary, h is clustered incorrectly because the distance measure did not consider the covariance or spread of the cluster. Therefore, information about the probable size of the true cluster was not used. The Mahalanobis distance measure takes this bias into account. Therefore it may be more likely to provide a truer grouping of the genes into clusters than other measures which do not consider this factor.

5.1.3 Clustering Methods

With our analysis of distance measures in mind, this subsubsection discusses some clustering methods. Clustering methods can be divided into two types: *partitional* and *hierarchical* methods. Partitional methods work by assigning the samples to clusters whose existence is independent of their members. Hierarchical methods work by agglomerating (bottom-up) or dividing (top-down) clusters to create a tree-like hierarchy of clusters known as a *dendogram*. This subsubsection focuses on partitional methods. The following subsubsection describes several examples of GRN inference carried out using partitional and hierarchical clustering and summarises clustering. C is used to refer to the number of clusters or the set of clusters.

Algorithm 1: The C-means clustering method.

```
Input:
$N \times M$, $N$ genes, each with $M$ examples
$d$, a distance measure
Output:
$C$ clusters of genes
begin
    Generate $C$ random points in the $M$-dimensional data space
    repeat
        foreach $n \in N$ do $n \rightarrow c : \arg\min_{c \in C} d(n, c)$
        Set each $c = \overline{n \in c}$
    until No $n$ changes to another $c$
end
```

C-means clustering[1; 71] (see figure 7) is a relatively simple partitional method with several variants. One of these variants is presented in figure 1.

With a sensible[2] distance measure the c-means method is guaranteed to find a local minimum in the variation between the members of each cluster. The method typically converges very quickly.

However it functions best on hyper-spherical clusters which are well separated and it usually does not return an optimal result. For that reason the algorithm is frequently run several times and the most compact result is used.

[2]Transitive and symmetric.

Running the algorithm several times does not address the greatest weakness though, which is that the number of clusters, C, must be specified *a priori*. While a search over the number of clusters could be performed this would be inefficient since the method is not guaranteed to return the optimal result each time it is run.

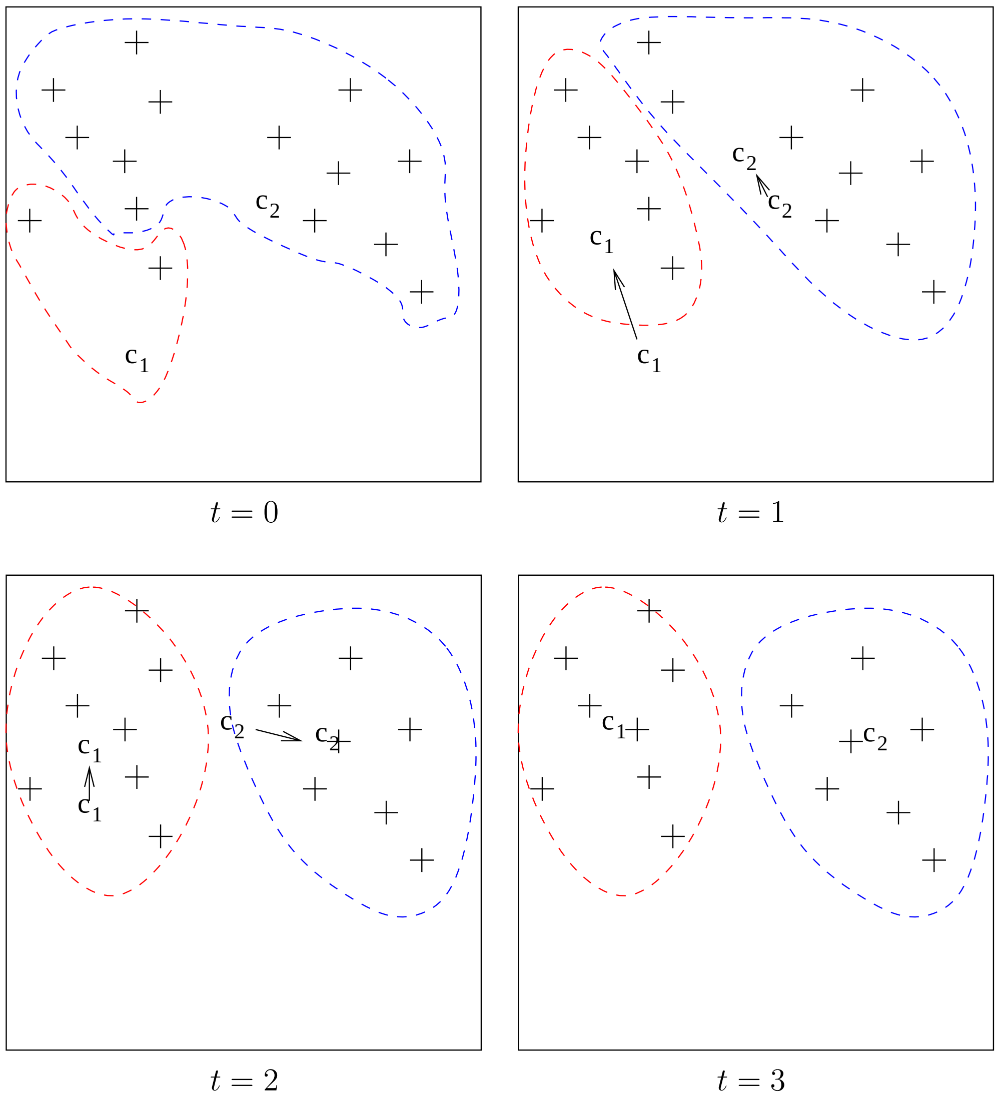


Figure 7: An example of C-means clustering. Each iteration of the method is shown. Cluster centres are denoted by c_i and their movement from iteration to iteration shown by directed arrows, except for $t = 3$ where old c_i are not shown due to overlap. Membership in each cluster is shown by dashed lines.

A *self-organising maps*[39, ch. 7] (SOM) is superficially very different to a C-means set of clusters. SOMs are a kind of one-layer feed forward neural network, where the neurons are organised into a lattice. Each neuron is an output neuron and associated with every neuron is an initially random vector of the same dimension as the inputs.

After each input to be classified is presented to the neural network the most similar neuron and neurons near it are adjusted so that their vector is more like the input vector. After many epochs the network will have stabilised into several groups of neurons, each of which form a blurrily defined cluster. Alternately, each output neuron can be interpreted as representing its own cluster, one similar to those of nearby neighbouring neurons.

Equation 3 presents a simplified competitive learning rule which does not consider a neuron's neighbours.

$$\Delta w_{ij} = \begin{cases} \alpha(x_i - w_{ij}) & \text{if neuron } j \text{ is most like the input} \\ 0 & \text{if neuron } j \text{ is not most like the input} \end{cases} \tag{3}$$

Just as with C-means clustering, SOM need the number and dimensionality (usually 2, but any $d \in \mathbb{I}$ is valid) of the output neurons to be specified in advance. The data is fit to this configuration. Prior specification of the number of clusters or neurons is a common weakness of both methods.

The requirement of this prior specification represents a weakness in both of these methods.

Another kind of clustering method that is applicable to GRN inference is *fuzzy clustering*. Fuzzy clustering is a description of a range of distance measures and methods which assign genes *fuzzy membership* to clusters. This means that any one gene can be a partial member of several clusters, which is biologically accurate[6]. Fuzzy methods are also comparatively robust in the face of noisy data.

Similarly, clustering methods which find hypergraphs[40; 81] allow any one gene to belong to more than one cluster. In a hypergraph an edge can connect to any number of vertices. This is analogous to a cluster.

Formally, fuzzy (and discrete) methods that allow a gene to have cluster membership greater than one, i.e. $\sum_j u_{ij} > 1$, create *covers*[72] over the data, and don't just *partition* it[23]. This use of the term partition is easily confused with the way a clustering method can be described as partitional. In this latter case it describes how clusters are found, in the former it describes data point membership in the clusters.

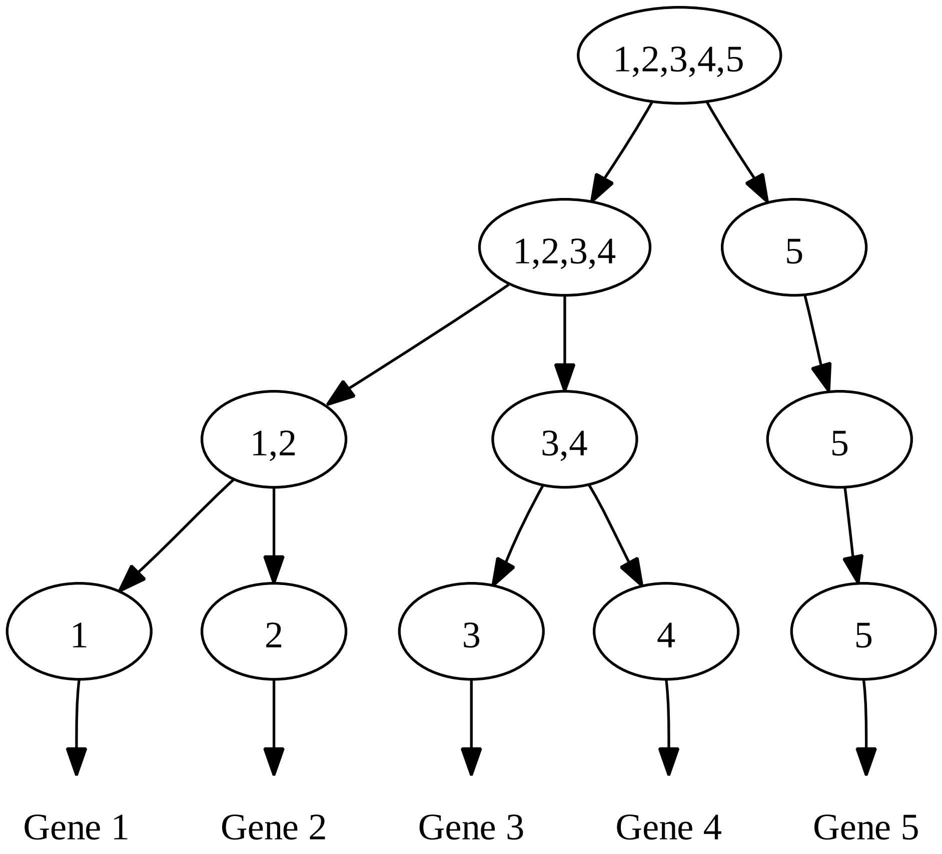


Figure 8: An example of a divisive hierarchical clustering method.

5.1.4 Previous Research

Clustering gene expression data from microarrays has been surveyed in several recent papers (e.g. [2; 23; 25; 99; 127] and others). Zhou et al. [127] compares a range of algorithms and focuses on combining two different distance measures (e.g. mutual information and fuzzy similarity or Euclidean distance and mutual information) into one overall distance measure. Azuaje [2] describes some freely available clustering software packages and introduces the SOTA algorithm. SOTA is a hierarchical clustering algorithm which determines C based on a validity threshold.

Zhou et al. [127] also suggests searching over cluster assignments using techniques such as *simulated annealing*. Simulated annealing is described in subsubsection 5.4.2. Initial membership of each gene in each fuzzy cluster is randomly assigned. As a part of the search process the fuzzy membership is swapped in the same way that discrete cluster memberships would be swapped while searching. This idea could be easily generalised to fuzzy covers, although the efficiency of such an algorithm is unclear.

The GRAM algorithm[5] is a clustering algorithm designed for clustering gene expression data. Although it requires candidate regulatory genes to be specified in advance, it still has a number of interesting characteristics. In particular, it combines protein-binding and gene expression data and it uses this non-expression data not just as a prior.

GRAM clusters genes based on the statistical likelihood of shared promoters first. A *core* for each cluster is calculated, based on the expression profiles of the genes with shared promoters. Other genes which match this expression profile relatively closely can be incorporated into the cluster, even if the evidence from the promoters was not strong enough for the gene to be incorporated in the first round. The algorithm finds a cover and not a partition of the genes, which is biologically more plausible.

[97] describes another clustering algorithm which uses a list of candidate regulators specified in advance to cluster genes into modules.

This algorithm has a number of strengths and weaknesses when compared with [5]. Unlike GRAM, Segal et al.'s algorithm only finds a partition of the data and not a cover. In addition, like Bar-Joseph et al.'s [5]'s algorithm, the number of clusters is effectively specified in advance by the use of candidate regulators. However the algorithm in [97] only needs expression data and can infer more complex combinatorial (Boolean AND/OR) regulatory relationships between regulators and genes in modules. Furthermore its predictions have been empirically confirmed and it is expected that the algorithm can be easily transfered to other metazoa with greater $\overline{k_{in}}$[97].

Because expression data is the only data which is used the algorithm is vulnerable to omissions in the data. If there are omissions then regulators may be inferred as a part of the module and members of the module may appear to be the regulator.

Another weakness is that a gene which is actually the regulator but which is not in the list of candidate regulators can not be selected. In that situation it is likely that a regulator of this gene will be selected as the regulator of the module. This is a general problem with inference using noisy and possibly contradictory data, especially if the search is artificially constrained by hard boundaries like a prior list of regulators.

Horimoto and Toh's [47] algorithm is very different from Segal et al.'s [97] and Bar-Joseph et al.'s [5]. Like theirs, it depends only on gene expression data. However, as opposed to using the Euclidean distance between the expression levels, the algorithm uses the Euclidean distance between the Pearson correlations of the genes. This means that when calculating the similarity of two genes the distance measure also incorporates the similarity of others as well. In this respect it echoes the Mahalanobis distance[73].

Furthermore the algorithm is hierarchical, not partitional, and determines the number of clusters more automatically than either of those described in [97] and [5]. Although it does rely on an arbitrary constant value x to determine what variance in expression profile between members of a cluster is acceptable (this is similar to SOTA), the use of this arbitrary constant is more defensible than the specification of an arbitrary C beforehand.

The algorithm works by clustering the genes based on their variance, until all genes are in one super-cluster. The super-cluster is recursively broken up into sub-clusters, and any cluster of genes whose variance is less than x is not broken up any more. Thus the algorithm finds the minimum number of clusters with variance less than x, given the initial hierarchical clustering.

In addition the calculated clusters were not the final output but were used to guide the regulatory network inference described in [114]. This research shows how clustering and module inference can be made a more integral part of the wider project of detailed GRN inference and not just used to create an overview of the main modules and their primary regulators.

Multi-stage inference ([23; 47; 114] can make principled inference over larger numbers of genes tractable. Although the underlying network is directed (as described in subsection 2.2) and may have very complex regulatory relationships these factors are conditionally independent of the graphical structure and do not need to be considered simultaneously.

Horimoto and Toh also found that nearly 20% of the gene pairs in a set of 2467 genes were Pearson-correlated at a 1% significance level. This emphasises the modular nature of GRN.

Mascioli et al.'s [75] algorithm is very interesting. It is a hierarchical algorithm, and clusters are created according to some validity criterion such as the maximum allowable average distance from the cluster centre. The validity criterion is smoothly changed, and this means that every cluster has a *lifetime*: the magnitude of the validity criterion from the point the cluster is created to the point that it splits into sub-clusters. The dendogram is cut so that the longest lived clusters are the final result.

This idea is very powerful and selects C automatically. However a cluster's lifetime may also depend on samples not in the cluster, and this is not necessarily appropriate if intra-cluster similarity is more important.

Eisen et al. [25] describes a data set and a simple hierarchical clustering algorithm that has reportedly been very commonly used by biologists[53, p. 1374].

Depending on the level of noise, [99] suggests not clustering the genes which are the most distant outliers and leaving them as singletons. Shamir and Sharan also discuss the difference between distance measures of the intra-cluster homogeneity (similarity) and inter-cluster heterogeneity (distance). A similar description of this distinction can be found in [53, p. 1383] as well.

Many clustering algorithms are fragile in the face of missing data. Troyanskaya et al. [116] suggests a number of methods which estimate missing values. The best of these appears to be k-nearest neighbour, with $6 \leq k \lessapprox 10$ to 20, although smaller k are more appropriate if the average cluster size is smaller. There does not appear to be any more recent research on clustering data sets with missing values.

Jiang et al. [53] moots using the network inferred using the clusters to guide the clustering itself. Such an approach is (loosely) analogous to the EM algorithm and may be asymptotically optimal.

Selection of the right clustering algorithm remains a challenging and demanding task, dependent on the data being used and the precise nature of any future inference.

5.1.5 Previous Biclustering Research

Biclustering is also known as co-clustering and direct clustering. It involves finding clusters of genes and clusters of samples at the same time. The gene expression data which is being clustered can be conceptualised as a matrix ($N \times M$) with one row for each gene and one column for each sample. In the previous clustering research that has been described, rows were clustered together. It is also possible for the columns to be clustered together. However, in a bicluster the rows (genes) and columns (samples) are simultaneously clustered, so that each bicluster is a row and column submatrice of the data. This means that a bicluster may represent a subset of the genes which are coregulated in some phenotypic situations. Such a model generalises naturally to a cover in which each gene can be in more than one cluster. This kind of biclustering cover algorithm is described in [101] and [109].

An early example of biclustering is [113]. [72] is a recent survey of the field. Madeira and Oliveira's [72] article also tabulates and compares many biclustering algorithms.

Biclusters can be submatrices in which (excluding noise) the rows are identical, the columns are identical, the rows and columns are identical, or there may be more complex but constant relationships between members of the submatrix. Depending on the purpose of the biclustering the user may be interested in only some kinds of bicluster.

Cluster homogeneity can be calculated for each 2×2 sub-submatrix in the submatrix that defines the bicluster. This may be another efficient and principled technique for dealing with missing values.

In general, optimal biclustering is an NP-hard problem[122]. In a limited number of cases, exhaustive enumeration is possible. In other cases, heuristics such as divide-and-conquer or a greedy search may be used to speed up the search[72].

5.2 Logical Networks

This subsection describes research which infer Boolean or other logical networks as a representation of a GRN. Boolean networks were first described by Kauffman[54] and have been a natural first approach to modeling regulatory networks. Prior to discussing examples of GRN inference carried out using Boolean networks we define them.

5.2.1 Overview

A Boolean network is a graph, $G = \langle N, F \rangle$. N is a set of nodes (also known as variables) which represent the genes and are either off (0) or on (1). There is no intermediate level of gene expression. F is a set of functions, one per gene. Each $f_i \in F$ has the form $i' = f(n_1, \ldots, n_i, \ldots, n_N)$ and allows the value of i to be deterministically calculated based on the current value of the set of genes N. Gene are updated simultaneously and synchronously. Figure 9 is an example of a Boolean network.

Each function $f_i \in f$ is a boolean function. For example: $i' = f_i(N) = (\neg i \vee i) \wedge j \wedge h)$. The expression level of each gene is updated by the application of its function.

An important distinction is drawn in [63] about the difference between *essential* and *fictitious* genes for each function $f \in F$. A gene j is fictitious with respect to the gene i iff $f_i(\ldots, n_{j-1}, 1, n_{j+1}, \ldots) = f_i(\ldots, n_{j-1}, 0, n_{j+1}, \ldots)$ and essential otherwise.

For example, i is a fictitious gene with respect to itself in figure 9. Only the essential genes matter when determining i. Because genetic regulatory networks are approximately sparse (i.e. $\overline{k_{in}} \ll N$) the number of essential genes for i, denoted E_i, is much less than N.

Recent research[10; 12] investigates the theoretical properties of *fuzzy logic networks* (FLN) and infers biological regulatory networks using time series expression data. FLN are a generalisation of Boolean networks to fuzzy logic.

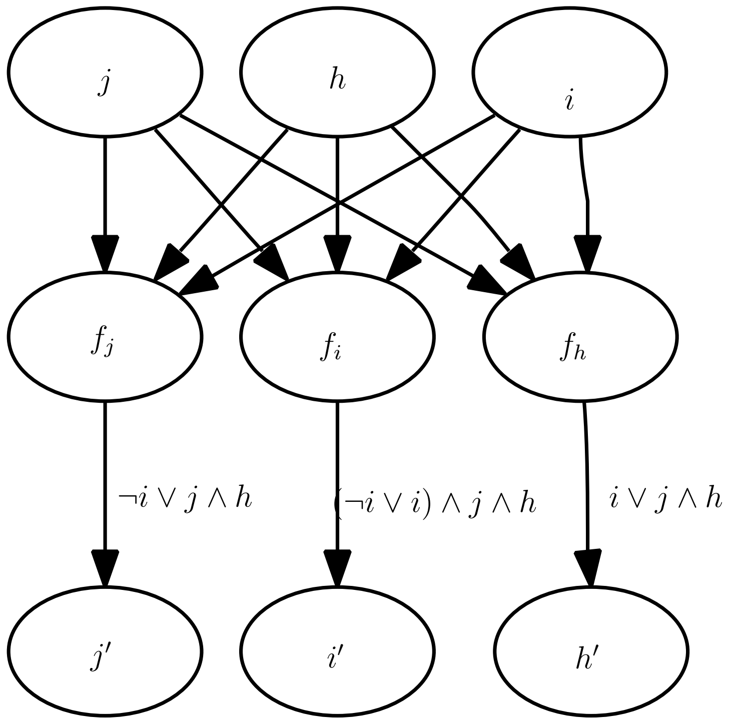


Figure 9: A Boolean network. For clarity each $f \in F$ has been made into a node. n and n' are connected via these function nodes. Normally the functions are implicit in the edges amongst N.

5.2.2 Previous Research

This subsubsection considers the REVEAL algorithm first. It was developed by Liang, Fuhrman and Somogyi and is described in [67].

It uses transition pairs, i.e. pairs of gene expression levels. These pairs need to come from a time series database. The algorithm uses mutual information (described in subsection 5.1) and the regulatory relationships for each gene are inferred iteratively.

The mutual information for each i' given each j is calculated first and the regulatory function for i is determined. If transitions for some genes are still incorrectly predicted by these functions then the mutual information for each i' given every pair of genes j, h is calculated and the process is iterated with increasing tuple size until the best regulatory function for each gene given up to k_{in} others is determined. The algorithm rapidly becomes intractable for large k_{in}; however, approximately 100 samples were all that were necessary when $N = 50$ and the number of possible states was therefore $2^{50} \approx 10^{15}$. As the algorithm was only tested on simulated data it is difficult to evaluate its performance.

Silvescu and Honavar's methodology[102] could be considered an extension of the the work done by Liang et al.. Like [67], the methodology relies on time series data, but it has been extended to use *temporal Boolean networks*. This methodology was developed to deal with delays in the regulatory network. Such a delay may come about for any number of reasons, including missing intermediary genes and spatial or biochemical delays between transcription and regulation. An example of a temporal Boolean network has been presented in figure 10.

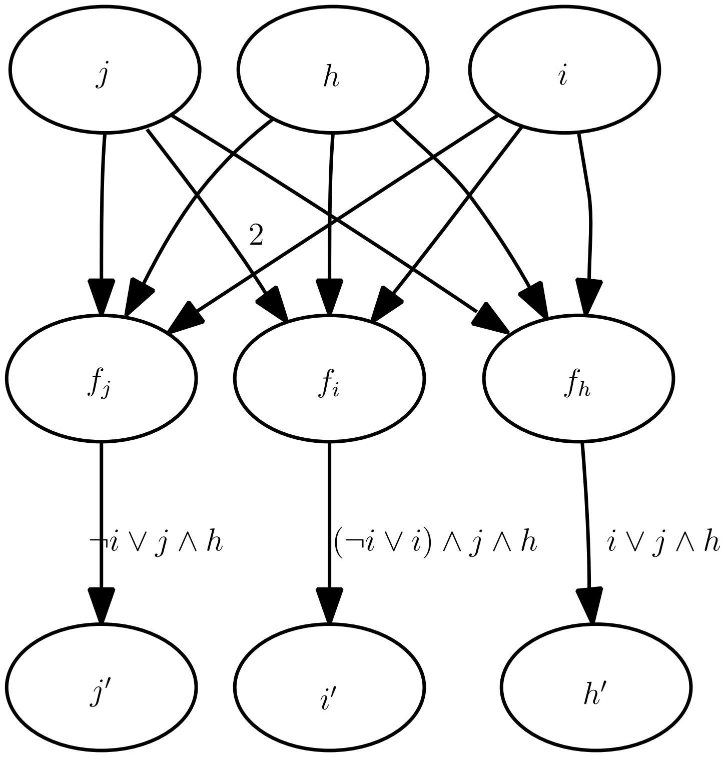


Figure 10: A temporal Boolean network. Presentation is as in figure 9, but delays are shown in brackets between genes and functions. The default delay if no annotation is present is assumed to be 0.

A temporal Boolean network is very similar to a normal Boolean network except that the functions $f \in F$ can refer to past gene expression levels. Rather than depending just on N_t to infer N_{t+1}, parameters to f_i can be annotated with an integer temporal delay.

For example: $i' = f_i(G) = (\neg i_0 \vee i_0) \wedge j_2 \wedge h_0)$, where i_0 refers to the value of i at time t and more generally i_x refers to the value of i at time $t - x$. Inference of temporal Boolean networks can be achieved by reformulating the problem as one of decision tree inference and inferring a decision tree for each gene. However, like [67] the method has only been applied to simulated data. Consequently, it's difficult to evaluate its performance.

The final example of Boolean network inference which we consider is described in [63]. Here, the authors considered how to take into account the often contradictory and inconsistent results which are obtained from microarray data. The aim of the approach is to find not just one function f_i but a set of functions F_i for each gene i. Each member of F_i may predict the wrong value for i based on the values of N. In those situations though other $f \in F_i$ may predict correctly.

Lähdesmäki et al. developed a methodology which would find all functions which made less than ϵ errors on the available data. Like the work carried out in [67] and [102] this research used time series data, but instead of simulating the data the research used data on 799 genes regulated by the cellular cycle. The search for all consistent or all *best-fit* functions could not be done efficiently and the regulatory functions of only five genes were identified.

As these examples have shown, Boolean networks have a number of disadvantages. Compared to the underlying biology, they create such a simple model that it can only give a broad overview of the regulatory network. In addition, despite a simple model, the algorithms are usually intractable. Typically, they are polynomial or worse in N and exponential in k_{in}^{max}. Furthermore, as k_{in}^{max} increases you need greater quantities of data to avoid overfitting.

However, the apparent reliance on time series data is deceptive, as algorithms which perform inference from steady state data are also possible. Time series data is expected to become more available with time. The simplicity of the functional representation is a strength as well as a weakness. Although the f have very low fidelity it means that they are more robust in the face of noisy data. Attractor basin analysis of Boolean networks can help provide a better understanding of the stability and causes of equilibrium gene expression levels. Such equilibria are often representative of particular cellular phenotypes[17].

Finally and briefly, this subsubsection briefly discusses a general logical formalism that relaxes some of the constraints facing Boolean networks. This formalism was developed by Thomas and others[111] in the 1990's and is summarised in [17].

Rather than allowing each gene to have only two possible values (0 or 1),

Table 1: A transition table. Such a table represents F for some GRN by enumerating all possible the result of all 2^N network states. This table represents the graph shown in figure 9
.

i	j	h	i'	j'	h'
1	1	1	1	1	1
1	1	0	0	0	0
1	0	1	0	0	1
1	0	0	0	0	0
0	1	1	1	1	1
0	1	0	0	0	0
0	0	1	0	1	0
0	0	0	0	0	0

each gene can have as many linearly ordered values ($\sigma_i^{(1)} < \sigma_i^{(2)} < \ldots < \sigma_i^{(k)}$) as there are genes it regulates. Just as in temporal Boolean networks[102], edges of the network are labeled with an integer. An edge from i to f_j and labeled with +2 means that i excites j when i is expressed at its second or higher level. Conversely a value of -2 means that i inhibits j whenever it is expressed at its second or higher level.

Although this approach has proven useful on several small regulatory networks[17], and it is clear that it can identify several kinds of regulatory function, it also serves as another example of the problems with tractability which plague GRN inference.

5.3 Differential Equations and Other Formalisms

Approaches based on differential equations predict very detailed regulatory functions. This is both a strength and a weakness. It is a strength because the resulting model is more complete. It is a weakness because it increases the complexity of the inference and there may not be enough data to reliably infer such a detailed model. In addition, the existence of a link and the precise nature of the regulatory function are two inferential steps and the regulatory function can be easily inferred given knowledge of the link.

Because there are so many different ways of doing this kind of high-fidelity inference this subsection just presents a number of examples, as in subsection 4.4.

The NIR (Network Inference via multiple Regression) algorithm is summarised in [24; 34]. It uses gene perturbations to infer ordinary differential equations (ODEs).

The rate of change for each gene is assumed to fit to an equation of the form $\frac{\delta \vec{N}}{\delta t} = A\vec{N} \cdot \vec{U}$, where $\vec{N}$ is a vector of the current gene expression values, $\vec{U}$ are the perturbations and A is a matrix representing the regulatory relationships. A is found using multiple regression on the results of a number of perturbations and the method has been applied to networks containing approximately 20 genes. The authors are working to extend their method to infer the functional network that makes up the *E. coli* DNA damage response and repair mechanisms[34].

D'haeseleer and Furhman[22] used time series data to infer sets of ODEs, one per gene. Because the underlying processes are discrete and the data is noisy, these ODEs are only approximations. This is acknowledged by the authors.

To address the problem of noise and to increase the quantity of data which could be used for network inference, the log expression levels were cubicly interpolated. More importantly, data from different tissue types and phenotypic situations was combined. This meant that the expression data gave a more complete picture of the network.

Furthermore, the research also modeled the influence of external factors by injecting kainic acid into the developing central nervous systems. Kainic acid is *"a glutamatergic agonist which causes seizures, localised cell death, and [which] severely disrupts the normal gene expression patterns"*[22, p2]. The robustness of the inferred networks was checked by artificially constructing new data sets through random Gaussian perturbations of the expression levels.

Kyoda et al. [61] uses perturbations of steady state data more directly in the network inference and provides another example of network model inference as ODE inference.

Their algorithm uses a modified version of the Floyd-Warshall algorithm[27] to infer the most parsimonious network, one ODE for each gene. Although noise and redundancy may make this incorrect it is the most correct network which can be inferred with the training data. Different inferred networks can fit the data equally well, because GRN are cyclic[61].

Kyoda et al.'s method is substantially more efficient than many others (it is $O(N^3)$) and is not bound by arbitrary k_{in}^{max}. This is a consequence of the fact that it uses perturbation data, which is much more informative than expression data, as discussed in subsection 4.2.

Tegner et al. [110] develop a similar algorithm which uses perturbations to infer ODEs. This methodology was successfully used to infer linear and non-linear networks, even though the inferred model is linear. This is because non-linear functions can appear to be linear if the perturbations are small enough. Perturbations were actively selected based on which would be most informative given past inference and perturbations. This is a different procedure than the standard, which is to generate the data and then infer from it sequentially. Actively selecting perturbations minimises the number of experiments necessary.

The effect of changes in k_{in}^{max} and N on algorithmic efficiency and the sample complexity[80] were described. It was found that $O(k_{in}^{max} \cdot \epsilon/\Delta \log N)$ experiments were necessary, where ϵ is the measurement error and Δ is the magnitude of the change in the average expression level of all genes. Perturbing multiple genes at once significantly reduced the number of experiments. This is because if only one gene is perturbed then very few genes show a change in their expression level.

Toh and Horimoto[114] used *graphical Gaussian models* (GGM) to find conditional dependencies between the clusters of genes found when the hierarchical clustering algorithm described in [47] was applied. As the clustering eliminated linear dependencies amongst the genes the resulting conditional dependencies were assumed to indicate inter-module regulation and not just modular co-expression.

Dependencies amongst the 34 clusters were found using steady state gene expression data, and the results compared with the BN inference in [33]. As in [33], because the raw results are conditional dependencies, they are undirected. Regulatory direction was manually annotated using other sources of information.

5.4 Bayesian Approaches

Bayesian probability is the application of logical reasoning about probabilities to model inference. It is based on the argument in table 2.

Table 2: The justification of Bayesian statistics. ',' denotes conjunction, Capitalised letters are random variables. Specifically, D denotes data and H denotes hypotheses.

$$\begin{array}{lll} & p(A,B) = p(A|B)p(B) = p(B|A)p(B) & \textit{by definition} \\ \therefore & p(D,H) = p(D|H)p(H) = p(H|D)p(D) & \textit{by substitution} \\ \therefore & p(H|D) = \frac{p(D|H)p(H)}{p(D)} & \square \end{array}$$

This means that we can determine the probability of a hypothesis (given the data) if we can work out the probability of the data given the hypothesis (easy), the probability of the data given all possible hypotheses (harder, but certainly possible in principle) and given the prior possibility of the hypothesis (easy). Although the idea of a hypothesis having a probability may seem counterintuitive, the Cox axioms[70] have shown that equating uncertainty with knowledge is valid.

5.4.1 Bayesian Networks

The idea that explanatory hypotheses can have probabilities in the same way that random variables do means that causal explanations are factorisations of a joint distribution. For example:

- the grass is wet either because it rained or because the sprinkler was on

entails the factorised joint distribution

$$p(W, R, S) = p(R)p(S)p(W|R, S)$$

This structure is formally represented as $G = \langle \eta, \theta \rangle$. η represents the structure of the graph, one variable for each gene and an unannotated edge to it from each gene it is regulated by. θ represents the parameters to the graph: the conditional probability distributions of each gene. A gene which is conditionally independent of all other genes must still have a distribution $p_i \in \theta$. Such a distribution is called a prior distribution.

The *lack* of an edge from i to j entails that, given j's *Markov blanket*, i and j are independent. This conditional independence can be expressed notationally as: $p(j|i, mb(j)) = p(j|mb(j))$, where $mb(j)$ denotes the Markov blanket of j. The Markov blanket[70] consists of a variable's parents, children and children's parents as defined by the edges in and out of the variables. See figure 11 for an example.

Using this definition, we describe some of the properties of Bayesian statistics and BN. Following that we discuss the ways in which BNs have been used as models for GRN inference and some of the issues which have arisen.

There are three basic properties of a BN which need to be considered immediately. The first of these is the fact that a BN must be an *acyclic* graph[24]. This is a problem because (as described in section 2) autoregulation and negative feedback are common in GRN[112]. The reason why BN must be acyclic is that a cyclic BN cannot be factorised.

Consider the BN shown in figure 12. The value of A depends on the value of B, the value of B depends on the value of C, and the value of C depends

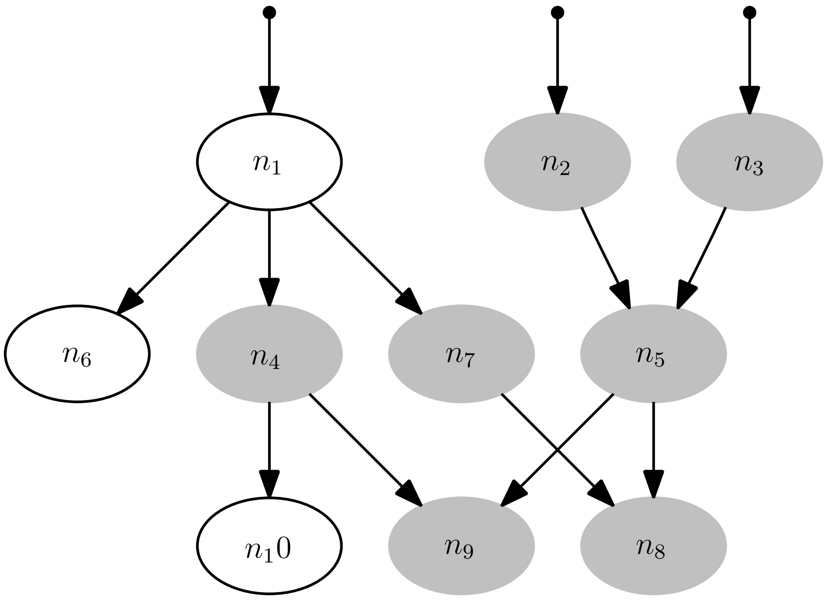


Figure 11: A Bayesian network and Markov blanket. Genes in the Markov blanket of n_5 are shown with a grey background. Priors for $n_{1..3}$ are denoted by incoming parentless edges.

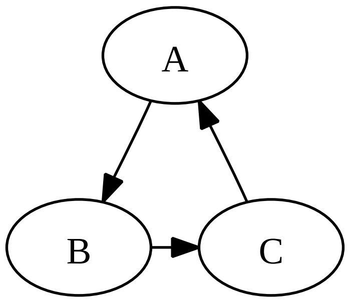


Figure 12: A cyclic Bayesian network. Impossible to factorise.

on the value of A. Equation 4 shows what happens when we try to show the factorisation of $p(A, B, C)$ implied by this graph, recursively expanding the distributions of the parents (pa) of each variable.

$$\begin{aligned} p(A,B,C) &= p(A|pa(A))p(pa(A)) \\ &= p(A|B)p(B|pa(B))p(pa(B)) \\ &= p(A|B)p(B|C)p(C|pa(C))p(pa(C)) \\ &= p(A|B)p(B|C)p(C|A)p(A|pa(A))p(pa(A)) \\ & \qquad \text{And so on...} \end{aligned} \tag{4}$$

The second property is that a BN is a different kind of model than an ODE or Boolean model. A BN does not provide a definite (if possibly incorrect) description of the functional regulatory relationships between genes. Instead it provides a probabilistic indication in the form of θ.

Such a model does not claim that if i is high then j will be low. Rather the model makes the claim that if $i \in High$ then there is a 46% chance that $j \in Low$, a 23% chance that $j \in Med$ and a 31% chance that $j \in High$. Such a model is more difficult to interpret but accurately reflects uncertainty in the data.

A third important property of Bayesian networks stems from their underlying statistical nature. Recall that $p(A|B)p(B) = p(B|A)p(A)$. This means that if that you only have observations of i and j it is impossible to infer whether i regulates j or j regulates i. The output of a Bayesian inference algorithm is not a single BN. Rather, they infer an arbitrary member of an *equivalence class*. Each network in an equivalence class makes the same predictions on the training data as all other networks in that equivalence class. In this way Bayesian networks are very similar to the GGM inferred in [114].

However, one of the reasons BNs are so widely used is that in certain situations you can infer causal relationships. By using perturbations, direction can be added to conditional dependencies. The formal details of this are described in [90]. Judea Pearl has done substantial work in this area and is considered by many to be the leading expert on causal inference.

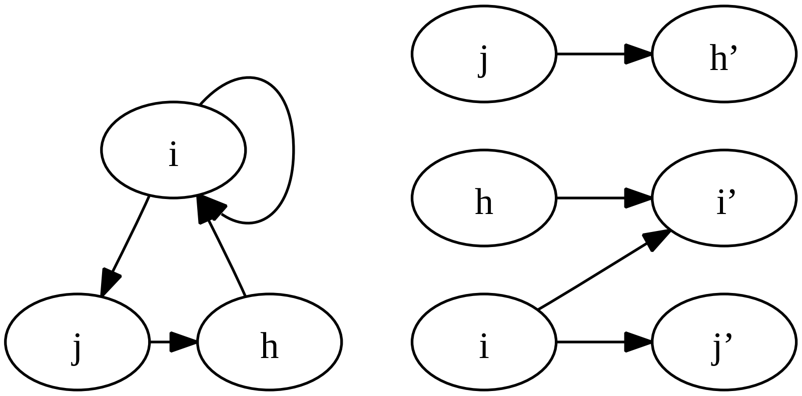
In summary, BNs are simultaneously attractive (causal inference and a principled way of handling noise and uncertainty) and unattractive (acyclic only). Although work has been carried out with some success despite the acyclic requirement (e.g. [33; 52]) there is an elegant and effective solution to it: *dynamic Bayesian networks*[31; 85].

Friedman et al. [31] also report a number of advantages that dynamic Bayesian networks (DBNs) have over alternate representations such as Kalman filters and *hidden Markov models* (HMM).

A DBN is a Bayesian network which has been temporally "unrolled". Typically we view variables as entities whose value changes over time. If we view them as constant, as they are in HMM, then we should represent i at t and i at $t+1$ with two different variables, say i_t and i_{t+1}.

If we assume that conditional dependencies cannot point backwards or "sideways" in time this means that the graph must be acyclic, even if i auto-regulates. A visual illustration of this is provided in figure 13.

Because we assume that the conditional dependencies are constant over time the BN also only needs to be unrolled for one time step. As Friedman et al. [31] point out, this also assumes that the prior distribution over the network is the same as the temporal distribution. Although this is not necessarily the case it is a reasonable assumption to make when inferring a stationary regulatory network.



(a) A cyclic BN (b) An equivalent, acyclic, DBN

Figure 13: A cyclic BN and an equivalent, acyclic, DBN. The prior network[31] is not shown in this diagram.

Many other approaches, such as D'haeseleer and Fuhrman's linear model[22], are special cases of DBN. This means that these other approaches can be no more effective than DBN and may be more efficient only by making assumptions which reduce their generality or accuracy some of the time.

5.4.2 Learning Bayesian Networks

The problem of learning a Bayesian network can be divided into two subproblems. The simpler problem is learning θ, the conditional distributions of the BN given its structure. This can be done with either full or incomplete training data.

The second and much more difficult problem is inference of the graph's structure, η, as well as of θ. This can also be done with either full or incomplete training data.

Bayesian network inference is a large research field, and comprehensively summarising it here is impossible. For such a summary we refer interested readers to [44; 84] and [31; 38]. This subsubsection focuses on just a few algorithms. It considers the case of θ inference first, before showing how many of the same algorithms can be applied to structural inference.

θ Inference

The simplest and most intuitive way of performing θ inference is just to count up and categorise the examples. This is a valid approach in the case of complete data. The result of such a count is a *maximum likelihood* (ML) estimate. The desired result is the *maximum a posteriori* (MAP), which incorporates any prior information. When the prior is uniform then ML = MAP. A uniform prior is common as it also maximises the informativeness of the data.

One advantage of BNs is that the likelihood of the graph is *decomposable*: the likelihood of any particular set of gene expression levels is just the product of each gene's likelihood. For pragmatic reasons the sums of log-likelihoods are often used. Decomposability is an advantage for online learning (it allows conditional distributions to be efficiently updated). It also makes calculation of the distributions from data fast in the case of offline learning.

To avoid zero-entries in the conditional probability distributions *pseudo-counts* are often used. Pseudocounts were invented by Laplace[65] for the sunrise problem[3] and they can be considered an *ad hoc* adjustment of the prior distribution. These are invented data values, normally 1 for each entry in each conditional distribution. The effect of these is to remove 0 values (and 1 values) from the conditional probability distributions. This is important because $p(i|j) = 0$ (or $p(i|j) = 1$) implies certainty, which is impossible and inferentially invalid with finite data.

Although pseudocounts are invalid if there is missing data, we speculate that if the available data is nearly complete then using pseudocounts could be accurate enough.

If the data is too incomplete for counts of the data to be used to estimate the ML θ there is a range of algorithms which can be used. These include greedy hill climbing with random restarts[95], the EM algorithm[19], simulated annealing[78] and *Markov Chain Monte Carlo*[43] (MCMC).

Each of these is a search method which takes a possible solution and iteratively improves it. Given infinite time each is guaranteed to return an optimal result. More practically, although still inefficient in comparison to the use of counts, each of the algorithms can also be expected to find near-

[3] *Viz:* What is the probability that the sun will rise tomorrow?

optimal results.

The independence and decomposability of the conditional distributions is a crucial element in the efficiency of the algorithm, as it makes calculation of L, the likelihood, much faster.

Hill Climbing with Random Restarts

The hill climbing with random restarts algorithm is summarised in algorithm figure 2. When this algorithm is being used the gradient of the current solution needs to be numerically or analytically calculated. With sufficient restarts hill climbing with random restarts finds the maximum likelihood parameters, θ_{ML}.

Algorithm 2: The hill climbing algorithm with random restarts. L denotes the likelihood function and s_θ is used to denote the slope at some θ. δ is a value used to indicate how far a candidate solution should be from the current solution. Typically this value is dynamically shrunk when s_θ is steep.

```
Input:
$N \times M$, the data to model
Output:
$\theta_{best}$, the best solution found
begin
    $\theta_{best} \longleftarrow -\infty$
    while search time remains do
        $\theta_{curr} \longleftarrow$ random solution
        if $L(\theta_{curr}) > L(\theta_{best})$ then $\theta_{best} \longleftarrow \theta_{curr}$
        $s_\theta \longleftarrow grad(\theta_{curr})$
        $\theta_{cand} \longleftarrow \theta_{curr} + \delta s_\theta$
        while $L(\theta_{cand}) > L(\theta_{curr})$ do
            $\theta_{curr} \longleftarrow \theta_{cand}$
            if $L(\theta_{curr}) > L(\theta_{best})$ then $\theta_{best} \longleftarrow \theta_{curr}$
            $s_\theta \longleftarrow grad(\theta_{curr})$
            $\theta_{cand} \longleftarrow \theta_{curr} + \delta s_\theta$
    return $\theta_{best}$
end
```

The EM Algorithm

Another algorithm is the expectation maximisation (EM) algorithm[19]. From an initial guess of θ_0 and the observed gene expression levels we can calculate N_i, the conditional probability distribution of the gene expression levels.

Holding these distributions constant and assuming that they are correct, θ_{i+1} is set so that N_i is the maximum likelihood gene expression levels.

This process is repeated and θ will converge on a local maxima. Random restarts or using simulated annealing (described next) to help determine θ_{i+1} means that the algorithm can also find θ_{ML}.

Compared to hill climbing, the real strength of the EM algorithm lies in the way in which it tractably incorporates the likely values of missing data into its search. This is particularly important for accuracy if several inter-related variables are simultaneously missing. The EM algorithm is outlined in algorithm figure 3.

Algorithm 3: The EM algorithm[19]. p_N is a function that returns the conditional distribution of the genes, N_i, given the current parameters θ and the observed training data $N \times M$. ML_N is a function that returns the θ which maximises the likelihood of some state of the genes, N. θ_{best} is the best set of parameters the search has found. If simulated annealing or random restarts are used this will be θ_{ML}.

```
Input:
$N \times M$, the data to use in the inference
$\eta$, the edges of the graph $G$
Output:
$G = \langle \eta, \theta_{best} \rangle$, the maximum likelihood BN given $\eta$.
begin
    $\theta_0 \longleftarrow$ initial guess, e.g. based on counts from noisy data
    repeat
        $N_{i+1} \longleftarrow p_N(\theta_i, N \times M)$
        $\theta_{i+1} \longleftarrow ML_N(N_{i+1})$
    until $p(\theta_i) = p(\theta_{i-1})$
    return $\theta_{best}$
end
```

Simulated Annealing

Simulated annealing [78] is a generalised Monte Carlo method which was inspired by the process of annealing metal. A Monte Carlo method is an iterative algorithm which is non-deterministic in its iterations. Metal is annealed by heating it to a very high temperature and then slowly cooling it. The resulting metal has a maximally strong structure.

Simulated annealing (SA) is very similar to hill climbing, except that the direction to travel in ($grad(\theta)$ in greedy hill climbing) and δ are sampled from a probability distribution for each transition. Transitions to better states are always accepted, whilst transitions to worse states are accepted with a probability p, which is lower for transitions to much worse states, and which decreases from transition to transition.

In this way simulated annealing is likely to explore a much wider part of the search space at the early stage of the search, but eventually it optimises greedily as hill climbing does[29]. If proposed transitions are compared to current positions based on their likelihood, simulated annealing finds θ_{ML}. If a prior is included in the calculations and the algorithm is changed to optimise for the posterior rather than the likelihood then it finds the MAP solution (θ_{MAP}) instead.

η Inference
The three search algorithms just discussed are naturally applicable to θ inference, but each of them can be used for η inference as well. For example, the EM algorithm has been generalised by Friedman *et al.*[30; 31]. Although we will not describe *structural EM* (SEM) in detail, it is very similar to standard EM. The main difference is in the ML step. As well as changing θ in this step so that the current N_i is maximally likely we also use the distribution N_i to search over the η-space.

Integrating over the Posterior
One common weakness of these algorithms is that they all find a single solution, either the ML or MAP. The MAP is normally considered to be a better solution than the ML, but it is better still to integrate over the posterior distribution[42]. This is because the MAP solution may be only one of several equi-probable solutions. Ignoring these other solutions when calculating a result means that there is a greater chance it will be in error.

Analytically integrating the posterior distribution is usually impossible. As a result we must do so numerically. Let γ be a sample from the posterior distribution of θ, let Γ be the result of the integration and let $N \times M$ be the data. A common way of integrating over a probability distribution is equation 5.

$$\Gamma = \frac{1}{N} \sum_{i=1}^{N} \gamma \sim p(\theta | N \times M, \eta) \tag{5}$$

To draw samples from the posterior distribution it must either be explicitly calculated and then sampled from, or it must be implicitly calculated and sampled from. Markov chain Monte Carlo (MCMC) algorithms[69] are the most common way of doing this, and they do it implicitly. The authors know of no general methods for explicitly calculating and sampling from an arbitrary posterior distribution.

Markov Chain Monte Carlo Algorithms
Common MCMC algorithms include the Metropolis-Hastings algorithm[43] and Gibbs samplings[35]. Other, much more intricate algorithms (such as Hybrid Monte Carlo[86]) have also been developed.

An MCMC algorithm works as follows. Starting from an initial state, an MCMC algorithm explores the space of possible solutions through a series of transitions selected from a probability distribution. This is very similar to simulated annealing, which is a generalised Monte Carlo method.

When using either MCMC or SA, each transition must have the *Markov property*, so that they make up a *Markov chain*. The probability of a transition which has the *Markov property* is independent of all previous states. I.e. $p(\gamma_{t+1}|\gamma_t) = p(\gamma_{t+1}|\gamma_t, \gamma_{t-1}, \ldots)$. *Higher order Markov chains* can also be defined. An s'th order Markov chain is independent of all states before γ_{t-s}. Markov chains are a type of Bayesian network and are very similar to dynamic Bayesian networks.

Because of the wide range of MCMC algorithms it is difficult to give an informative list of invariant properties. The next subsubsubsection describes the Metropolis-Hastings algorithm and we use that as an example instead.

Metropolis-Hastings MCMC
Assume we have a solution γ_t and that we can draw another solution, γ_{t+1}, using a proposal distribution q. Assume also that we can calculate the posterior probability of any solution γ (this is usually much easier than drawing samples from the posterior). The proposed transition to γ_{t+1} is accepted if and only if the *acceptance function* in equation 6 is true.

$$u < \frac{p(\gamma_{t+1})q(\gamma_t|\gamma_{t+1})}{p(\gamma_t)q(\gamma_{t+1}|\gamma_t)}, \text{ where } u \sim U(0,1) \tag{6}$$

If γ_{t+1} is a better solution than γ_t then the RHS of the equation will be greater than 1 and so the transition will always be accepted. Other MCMC methods are similar. Often they only differ in how they reject or accept samples or in how they generate them. For example, Metropolis-Hastings explores the search space completely at random, with a bias towards high probability regions. Other methods such as Hybrid Monte Carlo[86] tend to take larger steps and explore low probability areas more.

Many MCMC methods (such as Metropolis-Hastings and Gibbs sampling) work most efficiently if there are no extreme probabilities in the conditional distributions. If there are then convergence to the posterior may take longer[38]. The normal distribution $N(\theta, \sigma^2)$ is frequently used as the proposal distribution q. It is easy to sample from and can be worked with analytically.

So far we have not shown how MCMC can be used to integrate over the posterior. In fact, it would seem that it cannot and that MCMC just finds the MAP and then probabilistically wanders away from it.

However the probability of an MCMC algorithm making a particular transition is constant from iteration to iteration, and the acceptance function guarantees that the sequence of states converges to the posterior distribution over time. This means MCMC can be used to take samples from the posterior, by giving it time to converge and by leaving a large enough number of proposed transitions between successive samples to ensure that $x_t \perp\!\!\!\perp x_{t+1}$.

The number of transitions needed to converge depends on the cragginess and dimensionality of the search space. Considering 10^5 proposed transitions is usually sufficient to "burn in" to the posterior, and having 10^4 proposed transitions between samples usually means that they are independent. Convergence and independence can be checked by re-running the MCMC algorithm. If the results are significantly different from run-to-run it indicates that one or both of the conditions was not met.

Smarter MCMC algorithms can reduce the number of transitions that are needed. In addition (and this is another key advantage of MCMC algorithms) as the dimensionality of the problem grows the number of samples that needs to be taken is constant[70]. Although it may take more transitions to meet the convergence and independence criteria the impact of the *curse of dimensionality*[21; 24] on MCMC is limited in comparison to the impact on most search algorithms.

5.4.3 Scoring Bayesian Networks

One important aspect of structural inference is the comparison of two graphs so that the better one can be chosen. Without this there is no way of searching over graphs.

In this subsection G refers to the set of all graphs and $G' = \langle \eta', \theta' \rangle \in G$ refers to one particular graph. η, η', θ and θ' are defined similarly here.

The simplest scoring measure is the marginalised likelihood of the data, given the graph: $p(N \times M|G')p(G')$. This scoring measure is decomposable, as shown in equation 7. The log-sum is often used for pragmatic reasons.

However, the marginalised likelihood may overfit the data. This is because any edge which improves the fit will be added, and so the measure tends to make the graph too dense.

$$p(N \times M|G') = \prod_{m \in M} \prod_{n \in N} p(n \times m|N_n \times m, G') \tag{7}$$

A complexity penalty can be introduced if graphs are scored by their posterior probability, the *Bayesian Scoring Metric*[123]. Equation 8:

$$\begin{aligned} BSM(G', N \times M) &= \log p(G'|N \times M) \\ &= \log p(N \times M|G') + \log p(G') - \log p(N \times M) \end{aligned} \tag{8}$$

Because calculating this requires specifying a prior distribution over the G-space, overly complex graphs can be automatically penalised. This can be done by using a non-uniform prior, or one can just rely on the fact that more complex graphs have more free parameters in θ. Because $p(\sum_{\theta|\eta} \theta) = 1$, the probability of any particular θ given a complex η will be less. In effect, the probability distribution for θ is "spread out" over a larger space. [28] considers the implications of this in more detail.

For multinomial BN this scoring measure is commonly referred to as the BDe (*Bayesian Dirichlet equivalent*). The marginal probability of the data, $p(N \times M)$, is the probability of the data over all possible graphs. Because there are so many possible graphs this usually needs to be calculated using a Monte Carlo search method. The expansion of the marginal probability is shown in equation 9.

$$\begin{aligned} p(N \times M) &= \sum_{G' \in G} p(N \times M, G') \\ &= \sum_{G' \in G} p(N \times M|G')p(G') \\ &= \sum_{\eta' \in \eta} \sum_{\theta' \in \theta} p(N \times M|\theta', \eta')p(\theta'|\eta') \end{aligned} \tag{9}$$

Because Monte Carlo methods are time consuming, the posterior is often approximated using a measure called the Bayesian Information Criterion[96] (BIC). As [123] outlines, the BIC is an asymptotic approximation to the posterior and is faster than the BDe to calculate. However the BDe substantially outperforms the BIC when the training data is limited[124]. This is because the BIC over penalises complexity relative to the BDe when training data is limited[44]. Training data for GRN inference is typically very limited.

The BIC is shown in equation 10, where θ' is the maximum likelihood θ, given $N \times M$ and η', and $|\theta'|$ is the number of free parameters in θ'. If greater accuracy was required the BIC could be extended to integrate over all θ.

$$\log p(N \times M|\eta') \approx BIC(N \times M, G) = \log p(N \times M|\eta', \theta') - \frac{|\theta'|}{2}\log N \quad (10)$$

A number of other measures exist as well (e.g. the *minimum description length*[37; 64], the Local Criterion[44], the *Bayesian Nonparametric heteroscedastic Regression Criteria* (BNRC)[48; 49] and application of computational learning theory[15]). As with the BIC and the BDe, each of these tries to balance a better fitting graph with complexity controls.

No scoring method is appropriate in all situations. In addition, identifying different network structures may be crucial in order to get a representative sample of the posterior[44]. The use of only one scoring metric is unlikely to achieve this. Multi-classifiers[93] which use a range of scoring measures and search methods may address this problem.

5.4.4 Previous Research

The most comprehensive program of research into GRN inference using Bayesian networks to date has been carried out at Duke University by Jarvis, Smith, Yu, Hartemink and others[52; 104; 105; 123; 124]. This subsubsection also discusses some of the work done by Nir Friedman and others[31; 48; 85], who used DBNs for GRN inference.

The Duke project developed methods to understand songbird singing. GRN inference makes up only one part of the project. Another important part is the neural activity in different regions of the songbird brain. This is an epistatic problem which relates to temporal behaviour, and so time series gene expression data was necessary. This data also needs to be synchronised with neural activity in the songbird brain. As there is no way to collect this data with current technology the methodology uses data simulated by `BrainSim` and `GeneSim`, a pair of applications developed by Smith, Jarvis and others[104; 123; 124].

Several birds, each with the same regulatory networks and gene expression levels were simulated. `BrainSim` and `GeneSim` modeled the variables (genes, regions of the brain, singing/not singing) in 1 minute increments. Excluding regulation, expression levels were degraded back towards to a constitutive level each time step.

Between 40–90% of the genes simulated were "distractors" and varied their expression levels according to a normal distribution. The number of genes simulated differed from experiment to experiment and was in the range 20–100 ([124] and [104], respectively). It was claimed that the simulated data showed realistic regulatory time lags, and also between gene expression and neural activity[104].

Data for inference was sampled from the simulated data every 5 minutes. This was based on the underlying processes. It takes approximately 5 minutes for a gene to be transcribed, for the mRNA to be translated into a protein and for the protein to get back to the nucleus to regulate other genes[104].

The data was preprocessed before being used. The extracted expression and activity levels were continuous and a range of normalised discretisations were trialled. Discretisation involves a careful balance between tractability (a smaller number of values is always be more tractable) and accuracy.

In [124] 2, 3 and 4-bin discretisations were trialled. In addition discrete versus fuzzy (*hard* versus *soft*) discretisations were also evaluated. 3-bin hard discretisation performed best. The authors of the paper hypothesise that this performed better than 4-bin discretisation because the latter spread the data too thinly across the conditional distributions. 4-bin quartile-based hard discretisation was used in [104]. We think this was because more data was simulated for this earlier experiment, but are not certain.

There was no analysis of the differences between hard and soft discretisations. This is surprising. On information theoretic grounds and counter to their findings, we would expect the soft (fuzzy, continuous) approach to do better than the hard[48] ([103]).

Data interpolation was used to maximise training data informativeness. In particular, [124] found that linearly interpolating 5 data points between each pair of samples drawn gave significantly better recovery and reduced the number of false positive results.

[124] developed and made use of the *influence score*. This was applied to each pair of connected nodes in the highest scoring graph. If i regulates j then $-1 < I_{ij} < 1$, where the sign indicates the excitatory or inhibitory nature of the regulation and the magnitude gives some indication of the strength of the regulatory effect. This is important as it makes a joint distribution more comprehensible in biological terms.

The work done at Duke university has a number of weaknesses. The first, discussed in [52], is that the search algorithms and scoring measures (described in subsubsection 5.4.3) had problems finding convergent motifs. Unless there were more than 5000 data points only one regulator for each gene was found. It is also very difficult to evaluate the accuracy of the *NetworkInference* algorithm on real data.

However the algorithm does reliably infer regulatory cycles and cascade motifs. Topological aspects of the algorithm's effectiveness are discussed in more depth in [105]. Topological factors are also mentioned in [38].

Work done by Friedman et al. [31] has extended the BIC and BDe measures so that they can score graphs in the case of complete data. In addition they have extended SEM so that it works with incomplete data. This makes it an important alternative to MCMC when doing structural GRN inference. This methodology can also be applied to non-biological problems, such as predicting agent behaviour.

Friedman and others have proposed a *sparse candidate* algorithm for general BN inference[32]. This algorithm is optimised for problems in which there are either a great many variables or a great many examples. By using cues such as the mutual information of two variables the possible regulators can be restricted to some subset ($cand$) of size $|k|$ for each gene, where $k_{in} \ll N$. These restrictions on which genes are possible regulators are then used to guide network inference. Only a gene $i \in cand_j$ may be a regulator of j. Following each iteration the candidate sets are recalculated and the algorithm repeats until convergence.

Murphy and Mian[85] discuss DBNs with continuous θ. So far our discussion has only considered multinomial BN. In multinomial BN, each variable has one of some finite number of states. We have briefly discussed ways of discretising continuous data.

Alternately, each variable can be continuous and have a continuous conditional probability distribution. Its value is determined by drawing a sample from this distribution or by integrating over it. Continuous representations maximise the amount of information that can be extracted from the data, although they are also vulnerable to noise.

A common continuous distribution is a Gaussian, because it can be solved analytically. Although an arbitrary distribution could, in theory, be used and sampled from using MCMC, issues of tractability make this infeasible. These issues are discussed in more detail in [85].

There is some debate about the relative tractability of discrete and continuous Bayesian networks. For example, Jarvis et al. [52, p974] explicitly claim that continuous BNs are intractable and that discretising the data is essential. However, Murphy and Mian [85, 5.1] assert that exact inference in densely connected discrete BN is computationally intractable and must be approximated. Other authors explain the complexity of *hybrid Bayesian networks*, which combine continuous and multinomial variables[4; 89]. In any case, Bayesian inference, regardless of the nature of θ, is NP-complete[13].

A range of search algorithms have been used in this research, and there is no consensus. For example, [42] concluded that simulated annealing found better graphs than either greedy hill climbing or Metropolis-Hastings MCMC. On the other hand, [124] argued that greedy hill climbing with restarts was the most efficient and effective search method over graphs. Others, such as [48], used non-parametric regression. Because each algorithm is better for subtly different problems this variation is unsurprising, and [38, sections 3.2.1, 4.2.2] has some suggestions on how to select an algorithm.

6 Statistical and Computational Considerations

As has been described above, the data which can be used for GRN inference is noisy and often has missing values. Furthermore, microbiological systems are intrinsically stochastic. Although stochastic models exist[17] these are intractable for almost all epistatic and GRN inference problems.

This section discusses the problems of tractability (subsection 6.1) for continuous models. It also describes a particular statistical problem with GRN inference from microarrays (subsection 6.2).

6.1 Efficiency and Tractability

Although it is very rarely explicitly mentioned, careful reading of the research described in subsections 5.2–5.4 reveals that almost all of the algorithms described above must strongly limit N, the number of genes. Most must also strongly limit k_{in}^{max}, the maximum fan-in factor of any one gene.

k_{in}^{max}-unbounded network inference is almost always[4] $O(N^k) = O(N^N)$ or worse. This is true even of the simplest representations, such as Boolean networks[63]. Bayesian network inference is NP-hard, due to cycles in the undirected graph[13; 110]. Furthermore, DBN inference is even harder than BN inference[85].

The magnitude of the efficiency problem becomes clear when we consider the number of genes in *S. cerevisiae* (approximately 6,265) and compare it to the size of the inferred networks. See table 3 for examples. What this table illustrates is that improvements in hardware technology over the last 9 years have not led to improvement in either N or k_{in}^{max}. This is because the algorithmic complexity grows more quickly than our technology improves.

The two exceptions to this trend are informative. The clustering work carried out by Toh and Horimoto[114] illustrates one way in which large numbers of genes can be considered.

An alternate approach can be seen in [61]. By using perturbation data and assuming that changes are a result of the perturbations, a polynomial time algorithm independent of k_{in}^{max} based on Warshall's algorithm (as described in subsection 5.3) has been developed.

Although not tested on $N > 100$ the method nonetheless shows good recovery of data for $\overline{k_{in}} \leq 5$. These works show how preprocessing and the full use of data can maximise the efficiency and performance of the algorithm.

[4][61] was better, but it used an interactive style of learning that is not practical with current biochemical technology.

Table 3: GRN algorithmic efficiency against the number of genes N. Most of these results also require $k_{in}^{max} \leq 3$. [63] found all explanatory Boolean functions with $\epsilon \lesssim 5$ for the genes it solved for.

Research Citations	Maximum N
[67] (1998)	50
[74] (1998)	Unspecifiedly "small"
[118] (2001)	$\approx$ 100
[110] (2003)	$\approx$ 10–40
[63] (2003)	5
[104; 124] (2002–2004)	$\approx$ 20–100
[9] (2004)	100, also $\overline{k_{in}} = 10$
[24] (2006)	$\approx$ 20

A similar approach which also uses perturbation data and with $N = 100$, $\overline{k_{in}} = 10$ is discussed in [9].

Much of the time though perturbation data is unavailable. It is surprising that there appears to have been no research done on combining separately inferred graphs together, and that there is so little research which uses clustering to reduce the dimensionality of the problems. This would help address the problems of dimensionality.

6.2 Microarrays and Inference

This subsection very briefly summarises a problem with inference from microarrays and analyses the consequences. [14] is the original presentation.

The problem is as follows:

- Expression levels obtained from microarrays are the summed expression levels of $10^3 < x < 10^6$ cells.

- Except in limited circumstances, summed conditional dependencies can be different from individual conditional dependencies.

The summed conditional dependencies are identical to the conditional dependencies of the summands when the regulatory graph is singly connected (i.e. $i \longleftarrow j \longleftarrow h$, but not $i \longrightarrow j \longleftarrow h$). Similarly, if the noise is Gaussian and the regulatory relationships are all linear then the factorisation of the sum is identical with the factorisation of the summands.

Since neither of these conditions hold in GRN, the authors of [14] suggest that the apparently successful results of machine learning using sums of gene expression levels may be coincidental.

However, since the article was published substantially many more results have been biologically verified. While this does not indicate that Chu et al. are wrong, it does suggest that the conditional dependencies of the sum and the summands are nearly the same for GRN.

This in turn suggests that the extra uncertainty entailed by the loss of this guarantee has no substantial impact on the inference which can be carried out. It is important to remember the extent to which noise in the data already blurs any existing conditional dependencies, and the extent to which inferential methods are able to succeed despite this.

7 A Rough Map of the Field

This section presents a rough visual categorisation of GRN research. We have excluded results which just cluster the genes into modules but which do not predict general regulatory relationships.

Table 4: A visual categorisation of GRN research. Columns denote kinds of data (sometimes simulated) which can be used and rows denote types of model. The GRN inference in [92] was secondary, and that [82; 94] discuss epistatic inference. [98] also cites some work which uses phylogenetic data and ChIP assays. [42] and [41] used equilibrium microarray data as well.

	ODE etc.	Boolean	BN	Neural
Time series	[22; 91; 119]	[63; 67; 102]	[33; 52; 85; 126]	[118]
Equilib.	[97; 114]		[48]	
Perturb.	[9; 24; 34; 61; 110]	[125]		[82; 94]
Phylo.	[92]			
Chem./Loc.	[41]		[42]	

8 Conclusion and Directions

Research in GRN inference and epistatic analysis spans an enormous range of methods, fields, technologies and approaches. New technologies are being continuously developed and methods from other fields are being used in new ways. Nonetheless the problem of network inference remains open, and there appear to be several fruitful avenues for new research. These include:

- Incorporating cluster information into more detailed GRN inference.
- Combining separately learnt networks. Bayesian networks seem to be an ideal representation to use in this case. There does not appear to be any research on this issue to date. Considerations include:
  - What do we do in the case of disagreement between two or more inferred models?
  - How do we combine different inferred regulatory functions from two or more models?
  - What does agreement between two models mean for the posterior distribution?
  - What does it mean if two models agree but a third disagrees?
  - Is there a principled way of combining models inferred using different data sets?

  In related research, inferred networks have been compared with networks manually assembled from the primary literature [45].
- Gathering even more information from perturbation data, as [61] has done.
- Using multi-classifiers[93] to maximise the value of the data or to combine of several independently learnt networks.
- Incorporation of fuzzy techniques and clustering to address noise and the curse of dimensionality.

References

[1] David Arthur and Sergei Vassilvitskii. k-means++: The advantages of careful seeding. Technical Report 2006-13, Stanford University, 2006.

[2] Francisco Azuaje. Clustering-based approaches to discovering and visualing microarray data patterns. *Briefings in Bioinformatics*, 4(1): 31–42, March 2003.

[3] Yoganand Balagurunathan, Naisyin Wang, Edward R. Dougherty, Danh Nguyen, Michael L. Bittner, Jeffrey Trent, and Raymond Carroll. Noise factor analysis for cDNA microarrays. *Journal of Biomedical Optics*, 9(4):663–678, July/August 2004.

[4] Jim F. Baldwin and Enza Di Tomaso. Inference and learning in fuzzy Bayesian networks. In *FUZZ'03: The 12th IEEE International Conference on Fuzzy Systems*, volume 1, pages 630–635, May 2003.

[5] Z. Bar-Joseph, G. K. Gerber, T. I. Lee, N. J. Rinaldi, J. Y. Yoo, F. Robert, D. B. Gordon, E. Fraenkel, T. S. Jaakkola, R. A. Young, and D. K. Gifford. Computational discovery of gene modules and regulatory networks. *Nature Biotechnology*, 21(11):1337–1342, November 2003. ISSN 1087-0156. doi: 10.1038/nbt890. URL `http://www.nature.com/nbt/journal/v21/n11/abs/nbt890.html`.

[6] Albert-Laszlo Barabasi and Zoltan N. Oltvai. Network biology: Understanding the cell's functional organisation. *Nature Review Genetics*, 5(2):101–113, February 2004. doi: 10.1038/nrg1272. URL `http://www.nature.com/nrg/journal/v5/n2/abs/nrg1272.html`.

[7] G. W. Beadle and E. L. Tatum. Genetic control of biochemical reactions in neurosporta. In *Proceedings of the National Academy of Sciences, USA*, volume 27, pages 499–506, USA, 1941.

[8] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999. URL `citeseer.ist.psu.edu/ben-dor99clustering.html`.

[9] D. Di Bernardo, T. S. Gardner, and J. J. Collins. Robust identification of large genetic networks. *Pacific Symposium on Biocomputing*, pages 486–497, 2004.

[10] Y. Cao, P. Wang, and A. Tokuta. Reverse engineering of NK boolean network and its extensions — fuzzy logic network (FLN). *New Mathematics and Natural Computation*, 3(1):68–87, 2007.

[11] Yingjun Cao. *Fuzzy Logic Network Theory with Applications to Gene Regulatory Systems*. PhD thesis, Department of Electrical and Computer Engineering, Duke University, 2006.

[12] Yingjun Cao, Lingchu Yu, Alade Tokuta, and Paul P. Wang. S. pombe gene regulatory network inference using the fuzzy logic network. *New Mathematics and Natural Computation*, 2006? Full bibliographic information for this entry could not be obtained. Publication date is uncertain.

[13] David M. Chickering. Learning Bayesian networks is NP-Complete. In D. Fisher and H. J. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

[14] Tianjiao Chu, Clark Glymour, Richard Scheines, and Peter Spirtes. A statistical problem for inference to regulatory structure from associations of gene expression measurements with microarrays. *Bioinformatics*, 19(9):1147–1152, 2003.

[15] Ira Cohen, Nicu Sebe, Fabio G. Cozman, Marcelo C. Cirelo, and Thomas S. Huang. Learning Bayesian network classifiers for facial expression recognition using both labeled and unlabeled data. *cvpr*, 01:595–601, 2003. ISSN 1063-6919. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2003.1211408.

[16] G. C. Conant and A. Wagner. Convergent evolution of gene circuits. *Nature Genetics*, 34(3):264–266, 2003.

[17] Hidde de Jong. Modeling and simulation of genetic regulatory systems: A literature review. *Journal of Computational Biology*, 9(1):67–103, 2002.

[18] A. R. de Leon and K. C. Carriere. A generalized Mahalanobis distance for mixed data. *Journal of Multivariate Analysis*, 92(1):174–185, January 2005. doi: 10.1016/j.jmva.2003.08.006.

[19] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[20] Daniel C. Dennett. Real patterns. *Journal of Philosophy*, 88:27–51, 1991.

[21] Patrik D'haeseleer. *Resconstructing Gene Networks from Large Scale Gene Expression Data*. PhD thesis, University of New Mexico, Albuquerque, New Mexico, December 2000.

[22] Patrik D'haeseleer and S. Fuhrman. Gene network inference using a linear, additive regulation model. URL `citeseer.ist.psu.edu/286456.html`. Submitted to Bioinformatics, 1999.

[23] Patrik D'haeseleer, Shoudan Liang, and Roland Somogyi. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, 18(8):707–726, 2000. URL `citeseer.ist.psu.edu/article/dhaeseleer00genetic.html`.

[24] Michael E. Driscoll and Timothy S. Gardner. Identification and control of gene networks in living organisms via supervised and unsupervised learning. *Journal of Process Control*, 16(3):303–311, March 2006. doi: 10.1016/j.jprocont.2005.06.010. URL `http://dx.doi.org/10.1016/j.jprocont.2005.06.010`.

[25] Michael B. Eisen, Paul T. Spellman, Patrick O. Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Procedings of the National Academy of Sciences USA*, 95(25):14863–14868, December 1998. ISSN 0027-8424. doi: 10.1073/pnas.95.25.14863. URL `http://dx.doi.org/10.1073%2Fpnas.95.25.14863`.

[26] Peter C. FitzGerald, David Sturgill, Andrey Shyakhtenko, and Brian Oliverand Charles Vinson. Comparative genomics of drosophila and human core promoters. *Genome Biology*, 7:R53+, July 2006. ISSN 1465-6906. doi: 10.1186/gb-2006-7-7-r53. URL `http://genomebiology.com/2006/7/7/R53`.

[27] Robert W. Floyd. Algorithm 97: Shortest path. *Communications of the ACM*, 5(6):345, 1962. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/367766.368168.

[28] Christopher Fogelberg. Bayesian reasoning and Occam's razor. Position paper for discussion. Available from `http://www.syntilect.com/cgf/pubs:bayespp`, 2007.

[29] Christopher Fogelberg and Mengjie Zhang. Linear genetic programming for multi-class object classification. In Shichao Zhang

and Ray Jarvis, editors, *AI 2005: Advances in Artificial Intelligence: Proceedings of the 18th Australian Joint Conference on Artificial Intelligence, Lecture Notes in Computer Science, Vol. 3809.*, volume 3809 of *LNAI*, pages 369–379, Sydney, Australia, December 2005. Springer Verlag. ISBN 3-540-30462-2. Available from `http://www.syntilect.com/cgf/pubs:aaai2005`.

[30] Nir Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the 14th International Conference on Machine Learning*, pages 125–133. Morgan Kaufmann, 1997. URL `citeseer.ist.psu.edu/friedman97learning.html`.

[31] Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, volume 14, pages 139–147, San Francisco, CA, 1998. Morgan Kaufmann. URL `citeseer.ist.psu.edu/friedman98learning.html`.

[32] Nir Friedman, Iftach Nachman, and Dana Pe'er. Learning Bayesian network structure from massive datasets: The "sparse candidate" algorithm. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 206–215, San Francisco, CA, 1999. Morgan Kaufmann. URL `http://citeseer.ist.psu.edu/218089.html`.

[33] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using Bayesian networks to analyze expression data. *Journal of Computational Biology*, 7(3):601–620, 2000.

[34] Timothy S. Gardner, Skip Shimer, and James J. Collins. Inferring microbial genetic networks. *ASM News*, 70(3):121–126, 2004.

[35] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–742, 1984.

[36] G. Giaever, A. M. Chu, L. Ni, C. Connelly, L. Riles, S. Vronneau, S. Dow, A. Lucau-Danila, K. Anderson, B André, A. P. Arkin, A. Astromoff, M. El-Bakkoury, R. Bangham, R. Benito, S. Brachat, S. Campanaro, M. Curtiss, K. Davis, A. Deutschbauer, K. Entian, P. Flaherty, F. Foury, D. J. Garfinkel, M. Gerstein, D. Gotte, U. Güldener, J. H. Hegemann, S. Hempel, Z. Herman, D. F. Jaramillo, D. E. Kelly, S. L. Kelly, P. Kötter, D. LaBonte, D. C. Lamb, N. Lan, H. Liang, H. Liao

andL. Liu, C. Luo, M. Lussier, R. Mao, P. Menard, S. L. Ooi, J. L. Revuelta, C. J. Roberts, M. Rose, P. Ross-Macdonald, B. Scherens, G. Schimmack, B. Shafer, D. D. Shoemaker, S. Sookhai-Mahadeo, R. K. Storms, J. N. Strathern, G. Valle, M. Voet, G. Volckaert, C. Wang, T. R. Ward, J. Wilhelmy, E. A. Winzeler, Y. Yang, G. Yen, E. Youngman, K. Yu, H. Bussey, J. D. Boeke, M. Snyder, P. Philippsen, R. W. Davis, and M. Johnston. Functional profiling of the Saccharomyces cerevisiae genome. *Nature*, 418(6896):387–91, 2002.

[37] Peter Grünwald. The minimum description length principle and non-deductive inference. In Peter Flach, editor, *Proceedings of the IJCAI Workshop on Abduction and Induction in AI, Japan*, 1997.

[38] H. Guo and W. Hsu. A survey of algorithms for real-time Bayesian network inference. In *Joint AAAI-02/KDD-02/UAI-02 workshop on Real-Time Decision Support and Diagnosis Systems*, 2002. URL `citeseer.ist.psu.edu/guo02survey.html`.

[39] Kevin Gurney. *An Introduction to Neural Networks*. Taylor & Francis, Inc., Bristol, PA, USA, 1997. ISBN 1857286731.

[40] Eui-Hong Han, George Karypis, Vipin Kumar, and Bamshad Mobasher. Clustering based on association rule hypergraphs. In *Research Issues on Data Mining and Knowledge Discovery*, page TODO, 1997. URL `citeseer.ist.psu.edu/article/han97clustering.html`.

[41] Christopher T. Harbison, Benjamin D. Gordon, Tong I. Lee, Nicola J. Rinaldi, Kenzie D. MacIsaac, Timothy W. Danford, Nancy M. Hannett, Jean-Bosco Tagne, David B. Reynolds, Jane Yoo, Ezra G. Jennings, Julia Zeitlinger, Dmitry K. Pokholok, Manolis Kellis, Alex P. Rolfe, Ken T. Takusagawa, Eric S. Lander, David K. Gifford, Ernest Fraenkel, and Richard A. Young. Transcriptional regulatory code of a eukaryotic genome. *Nature*, 431(7004):99–104, 2004. doi: 10.1038/nature02800. URL `http://dx.doi.org/10.1038/nature02800`.

[42] A. J. Hartemink, D. K. Gifford, T. S. Jaakkola, and R. A. Young. Combining location and expression data for principled discovery of genetic regulatory network models. *Pacific Symposium on Biocomputing*, pages 437–449, 2002.

[43] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[44] David Heckerman. A tutorial on learning with Bayesian networks. Technical report, Microsoft Research, Redmond, Washington, 1995. URL `http://citeseer.ist.psu.edu/41127.html`.

[45] Markus J. Herrgard, Markus W. Covert, and Bernhard o. Palsson. Reconciling gene expression data with known genome-scale regulatory network structures. *Genome Research*, 13(11):2423–2434, 2003. doi: 10.1101/gr.1330003. URL `http://www.genome.org/cgi/content/abstract/13/11/2423`.

[46] Veronica F. Hinman, Albert T. Nguyen, Andrew R. Cameron, and Eric H. Davidson. Developmental gene regulatory network architecture across 500 million years of echinoderm evolution. *Proceedings of the National Academcy of Sciences, USA*, 100(23):13356–13361, November 2003. URL `http://www.pnas.org/cgi/content/full/100/23/13356`.

[47] Katsuhisa Horimoto and Hiroyuki Toh. Statistical estimation of cluster boundaries in gene expression profile data. *Bioinformatics*, 17(12): 1143–1151, 2001.

[48] Seiya Imoto, Takao Goto, and Satoru Miyano. Estimation of genetic networks and functional structures between genes by using Bayesian networks and nonparametric regression. In *Pacific Symposium on Biocomputing*, volume 7, pages 175–186, 2002.

[49] Seiya Imoto, SunYong Kim, Takao Goto, Sachiyo Aburatani, Kousuke Tashiro, Satoru Kuhara, and Satoru Miyano. Bayesian network and nonparametric heteroscedastic regression for nonlinear modeling of genetic network. *Journal of Bioinformatics and Computational Biology*, 1(2):231–252, 2003.

[50] Karolinksa Institutet. Main website, CMM Core Microarray Facility, 2008. URL `http://ki.se`. Accessed 28th March, 2008.

[51] F. Jacob and J. Monod. Genetic regulatory mechanisms in the synthesis of proteins. *Journal of Molecular Biology*, 3:318–356, 1961.

[52] E. D. Jarvis, V. A. Smith, K. Wada, M. V. Rivas, M. McElroy, T. V. Smulders, P. Carninci, Y. Hayashizaki, F. Dietrich, X. Wu, P McConnell, J Yu, P. P. Wang, A. J. Hartemink, and S. Lin. A framework for integrating the songbird brain. *Journal of Computational Physiology A*, 188:961–80, December 2002.

[53] Daxin Jiang, Chun Tang, and Aidong Zhang. Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering.*, 16(11):1370–1386, 2004. ISSN 1041-4347. doi: http://dx.doi.org/10.1109/TKDE.2004.68.

[54] S. A. Kauffman. *The Origins of Order: Self-Organization and Selection in Evolution.* Oxford University Press, 1993.

[55] Stuart A. Kauffman. Antichaos and adaptation. *Scientific American*, 265(2):78–84, August 1991. ISSN 0036-8733.

[56] Sunyong Kim, Seiya Imoto, and Satoru Miyano. Dynamic Bayesian network and nonparametric regression for nonlinear modeling of gene networks from time series gene expression data. *Biosystems*, 75(1-3):57–65, July 2004. doi: 10.1016/j.biosystems.2004.03.004. URL `http://dx.doi.org/10.1016/j.biosystems.2004.03.004`.

[57] H. Kitano. Computational systems biology. *Nature*, 420(6912):206–210, November 2002. ISSN 0028-0836. doi: 10.1038/nature01254. URL `http://dx.doi.org/10.1038/nature01254`.

[58] Lev Klebanov and Andrei Yakovlev. How high is the level of technical noise in microarray data? *Biology Direct*, 2:9+, April 2007. ISSN 1745-6150. doi: 10.1186/1745-6150-2-9. URL `http://dx.doi.org/10.1186/1745-6150-2-9`.

[59] M. A. Koch, B. Weisshaar, J. Kroymann, B. Haubold, and T. Mitchell-Olds. Comparative genomics and regulatory evolution: conservation and function of the chs and apetala3 promoters. *Molecular Biology and Evolution*, 18(10):1882–1891, October 2001. ISSN 0737-4038. URL `http://mbe.oxfordjournals.org/cgi/content/full/18/10/1882`.

[60] Eugene F. Krause. *Taxicab Geometry.* Dover Publications, 1987.

[61] Koji M. Kyoda, Mineo Morohashi, Shuichi Onami, and Hiroaki Kitano. A gene network inference method from continuous-value gene expression data of wild-type and mutants. *Genome Informatics*, 11:196–204, 2000.

[62] Lawrence Berkeley Laboratory. Berkeley Lab Currents, April 21, 2000, 2008. URL `http://www.lbl.gov`. Accessed 28th March, 2008.

[63] Harri Lähdesmäki, Ilya Shmulevich, and Olli Yli-Harja. On learning gene regulatory networks under the Boolean network model. *Machine Learning*, 52(1–2):147–167, 2003. ISSN 0885-6125.

[64] Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. In *Computational Intelligence*, volume 10, pages 269–293, 1994.

[65] Pierre-Simon Laplace. *Essai philosophique sur les probabilits.* Mme. Ve. Courcier, 1814.

[66] Phillip P. Le, Amit Bahl, and Lyle H. Ungar. Using prior knowledge to improve genetic network reconstruction from microarray data. *In Silico Biology*, 4, 2004.

[67] Shoudan Liang, S. Fuhrman, and Roland Somogyi. REVEAL: a general reverse enginerring algorithm for inference of genetic network architectures. In *Pacific Symposium on Biocomputing*, pages 18–29, 1998.

[68] Pek Y. Lum, Christopher D. Armour, Sergey B. Stepaniants, Guy Cavet, Maria K. Wolf, Scott J. Butler, Jerald C. Hinshaw, Philippe Garnier, Glenn D. Prestwich, and Amy Leonardson. Discovering modes of action for therapeutic compounds using a genome-wide screen of yeast heterozygotes. *Cell*, 116(1):121–137, January 2004. doi: 10.1016/S0092-8674(03)01035-3. URL `http://dx.doi.org/10.1016/S0092-8674(03)01035-3`.

[69] David J. C. MacKay. Introduction to Monte Carlo methods. In M. I. Jordan, editor, *Learning in Graphical Models*, NATO Science Series, pages 175–204. Kluwer, 1998.

[70] David J. C. MacKay. *Information Theory, Inference, and Learning Algorithms.* Cambridge University Press, 2003. URL `http://www.cambridge.org/0521642981`. Available from `http://www.inference.phy.cam.ac.uk/mackay/itila/`.

[71] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297. University of California Press, 1967.

[72] Sara C. Madeira and Arlindo L. Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 1(1):24–45, 2004. doi: 10.1109/TCBB.2004.2. URL `http://dx.doi.org/10.1109/TCBB.2004.2`.

[73] P. C. Mahalanobis. On the generalised distance in statistics. In *Proceedings of the National Institute of Science of India 12*, pages 49–55, 1936.

[74] G. Marnellos and E. Mjolsness. A gene network approach to modeling early neurogenesis in drosophila. In *Pacific Symposium on Biocomputing*, volume 3, pages 30–41, 1998.

[75] Fabio Massimo Frattale Mascioli, Antonello Rizzi, Massimo Panella, and Giuseppe Martinelli. Scale-based approach to hierarchical fuzzy clustering. *Signal Processing*, 80(6):1001–1016, 2000. ISSN 0165-1684. doi: http://dx.doi.org/10.1016/S0165-1684(00)00016-5.

[76] Luke McCrohon. AI Battle Tanks: Learning Through Destruction. URL `http://www.mcs.vuw.ac.nz/ mccrohluke/battletanksposter.pdf`. VUW BSc(Hons) Poster Presentation, 2006.

[77] D. C. McShan, M. Updadhayaya, and Imran Shah. Symbolic inference of xenobiotic metabolism. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Tiffany A. Jung, and Teri E. Klein, editors, *Pacific Symposium on Biocomputing*, pages 545–556. World Scientific, 2004. ISBN 981-238-598-3.

[78] N. A. Metropolis, M. Rosenbluth, A. Teller, and E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1956.

[79] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, October 2002. ISSN 1095-9203. doi: 10.1126/science.298.5594.824. URL `http://www.sciencemag.org/cgi/content/abstract/298/5594/824`.

[80] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[81] E. Mjolsness, R. Castano, and A. Gray. Multi-parent clustering algorithms from stochastic grammar data models. Technical Report JPL-ICTR-99-5, JPL, 1999.

[82] Alison A. Motsinger, Stephen L. Lee, George Mellick, and Marylyn D. Ritchie. GPNN: Power studies and applications of a neural network method for detecting gene-gene interactions in studies of human disease. *BMC Bioinformatics*, 7:39, 2006. URL `http://dx.doi.org/10.1186/1471-2105-7-39`.

[83] T. M. Murali and Simon Kasif. Extracting conserved gene expression motifs from gene expression data. In *Pacific Symposium on Biocomputing*, pages 77–88, 2003.

[84] K. Murphy. Learning Bayes net structure from sparse data sets. Technical report, Comp. Sci. Div., UC Berkeley, 2001. URL `http://citeseer.ist.psu.edu/murphy01learning.html`.

[85] K. Murphy and S. Mian. Modelling gene expression data using dynamic Bayesian networks. Technical report, Computer Science Division, University of California, Berkeley, CA, 1999.

[86] R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993. URL `citeseer.ist.psu.edu/neal93probabilistic.html`.

[87] Matti Nykter, Tommi Aho, Miika Ahdesmäki, Pekka Ruusuvuori, Antti Lehmussola, and Olli Yli-Harja. Simulation of microarray data with realistic characteristics. *Bioinformatics*, 7:349, July 2006.

[88] University of Utah. Learn Genetics, 2008. URL `http://learn.genetics.utah.edu`. Accessed 28th March, 2008.

[89] Heping Pan and Lin Liu. Fuzzy Bayesian networks - a general formalism for representation, inference and learning with hybrid Bayesian networks. *IJPRAI*, 14(7):941–962, 2000.

[90] Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82 (4):669–709, 1995. URL `citeseer.ist.psu.edu/55450.html`.

[91] Theodore J. Perkins, Joannes Jaeger, John Reinitz, and Leon Glass. Reverse engineering the gap gene network of drosophila melanogaster. *PLoS Computational Biology*, 2(5):e51+, May 2006. doi: 10%2E1371%2Fjournal%2Epcbi%2E0020051. URL `http://dx.doi.org/10%2E1371%2Fjournal%2Epcbi%2E0020051`.

[92] Moshe Pritsker, Yir-Chung Liu, Michael A. Beer, and Saeed Tavazoie. Whole-genome discovery of transcription factor binding sites by network-level conservation. *Genome Research*, 14 (1):99–108, January 2004. doi: 10.1101/gr.1739204. URL `http://dx.doi.org/10.1101/gr.1739204`.

[93] Romesh Ranawana and Vasile Palade. Multi-classifier systems: Review and a roadmap for developers. *International Journal of Hybrid Intelligent Systems*, 3(1):35–61, 2006. ISSN 1448-5869.

[94] Marylyn D. Ritchie, Bill C. White, Joel S. Parker, Lance W. Hahn, and Jason H. Moore. Optimization of neural network architecture using genetic programming improves detection and modeling of gene-gene interactions in studies of human diseases. *BMC Bioinformatics*, 4:28, 2003.

[95] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (2nd Edition).* Prentice Hall, December 2002. ISBN 0137903952.

[96] Gideo Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[97] E. Segal, M. Shapira, A. Regev, Dana Pe'er, David Botstein, D. Koller, and Nir Friedman. Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics*, 34(2):166–176, June 2003. ISSN 1061-4036. doi: 10.1038/ng1165. URL `http://www.nature.com/ng/journal/v34/n2/abs/ng1165.html`.

[98] E. Segal, Nir Friedman, N. Kaminski, A. Regev, and D. Koller. From signatures to models: Understanding cancer using microarrays. *Nature Genetics*, 37:S38–S45, June 2005. By invitation.

[99] Ron Shamir and Roded Sharan. *Current Topics in Computational Biology*, chapter Algorithmic approaches to clustering gene expression data, pages 269–300. MIT press, Cambridge, Massachusetts, 2002. (T. Jiang, T. Smith, Y. Xu, and M. Q. Zhang, eds).

[100] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 & 623–656, July & October 1948.

[101] Qizheng Sheng, Yves Moreau, and Bart De Moor. Biclustering microarray data by Gibbs sampling. *Bioinformatics*, 19:ii196–ii205, 2003.

[102] A. Silvescu and V. Honavar. Temporal Boolean network models of genetic networks and their inference from gene expression time series. *Complex Systems*, 13:54–70, 2001.

[103] D. S. Sivia. *Data Analysis: A Bayesian Tutorial.* Clarendon Press, Oxford, 1996.

[104] V. A. Smith, E. D. Jarvis, and A. J. Hartemink. Evaluating functional network inference using simulations of complex biological systems. *Bioinformatics*, 18:S216–S224, 2002.

[105] V. Anne Smith, Erich D. Jarvis, and Alexander J. Hartemink. Influence of network topology and data collection on network inference. In *Pacific Symposium on Biocomputing*, pages 164–175, 2003.

[106] Paul T. Spellman, Gavin Sherlock, Michael Q. Zhang, Vishwanath R. Iyer, Kirk Anders, Michael B. Eisen, Patrick O. Brown, David Botstein, and Bruce Futcher. Comprehensive identification of cell cycle-regulated genes of the yeast Saccharomyces cerevisiae by microarray hybridization. *Molecular Biology of the Cell*, 9(12):3273–3297, December 1998. ISSN 1059-1524. URL `http://www.molbiolcell.org/cgi/content/abstract/9/12/3273`.

[107] Peter Spirtes, Clark Glymour, Richard Scheines, Stuart Kauffman, Valerio Aimale, and Frank Wimberly. Constructing Bayesian network models of gene expression networks from microarray data. In *Proceedings of the Atlantic Symposium on Computational Biology, Genome Information Systems and Technology*, 2000.

[108] Kim Sterelny and Paul E. Griffiths. *Sex and Death : An Introduction to Philosophy of Biology (Science and Its Conceptual Foundations series)*. University Of Chicago Press, June 1999. ISBN 0226773043.

[109] Chun Tang, Li Zhang, Aidong Zhang, and M. Ramanathan. Interrelated two-way clustering: An unsupervised approach for gene expression data analysis. *Proceedings of the IEEE 2nd International Symposium on Bioinformatics and Bioengineering Conference, 2001*, pages 41–48, 4–6 November 2001. doi: 10.1109/BIBE.2001.974410.

[110] J. Tegner, M. K. Yeung, J. Hasty, and J. J. Collins. Reverse engineering gene networks: integrating genetic perturbations with dynamical modeling. *Proceedings of the National Academy of Sciences, USA*, 100(10): 5944–5949, May 2003. ISSN 0027-8424. doi: 10.1073/pnas.0933416100. URL `http://www.pnas.org/cgi/content/abstract/100/10/5944`.

[111] Rene Thomas. Regulatory networks seen as asynchronous automata: A logical description. *Journal of Theoretical Biology*, 153:1–23, 1991.

[112] Rene Thomas. Laws for the dynamics of regulatory networks. *International Journal of Developmental Biology*, 42:479–485, 1998.

[113] Robert Tibshirani, Trevor Hastie, Mike Eisen, Doug Ross, David Botstein, and Pat Brown. Clustering methods for the analysis of DNA microarray data. Technical report, Stanford University, October 1999.

[114] Hiroyuki Toh and Katsuhisa Horimoto. Inference of a genetic network by a combined approach of cluster analysis and graphical Gaussian modeling. *Bioinformatics*, 18(2):287–297, 2002.

[115] A. H. Tong, M. Evangelista, A. B. Parsons, H. Xu, G. D. Bader, N. Pagé, M. Robinson, S. Raghibizadeh, C. W. Hogue, H. Bussey, B. Andrews, M. Tyers, and C. Boone. Systematic genetic analysis with ordered arrays of yeast deletion mutants. *Science*, 294(5550):2364–2368, December 2001. ISSN 0036-8075. doi: 10.1126/science.1065810. URL `http://dx.doi.org/10.1126/science.1065810`.

[116] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, June 2001. ISSN 1367-4803.

[117] Jean-Philippe Vert and Yoshihiro Yamanishi. Supervised graph inference. In Lawrence K. Saul, Yair Weiss, and Léon Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1433–1440. MIT Press, Cambridge, MA, 2005.

[118] Jiri Vohradskỳ. Neural network model of gene expression. *FASEB Journal*, 15:846–854, 2001.

[119] Yong Wang, Trupti Joshi, Xiang-Sun Zhang, Dong Xu, and Luonan Chen. Inferring gene regulatory networks from multiple microarray datasets. *Bioinformatics*, 22(19):2413–2420, 2006. doi: 10.1093/bioinformatics/btl396.

[120] Rui Xu and Donald Wunsch II. Survey of clustering algorithms. *IEEE Transactions on Neural Networks*, 16(3):645–678, May 2005.

[121] Y. Yamanishi, J.-P. Vert, and M. Kanehisa. Protein network inference from multiple genomic data: a supervised approach. *Bioinformatics*, 20(1):363–370, 2004. ISSN 1367-4803. doi: http://dx.doi.org/10.1093/bioinformatics/bth910.

[122] E. Yang, P. T. Foteinou, K. R. King, M. L. Yarmush, and I. P. Androulakis. A novel non-overlapping bi-clustering algorithm for network generation using living cell array data. *Bioinformatics*, 23(17):2306–2313, 2007. doi: 10.1093/bioinformatics/btm335.

[123] J. Yu, V. A. Smith, P. P. Wang, A. J. Hartemink, , and E. D. Jarvis. Using Bayesian network inference algorithms to recover molecular genetic

regulatory networks. In *International Conference on Systems Biology (ICSB02)*, December 2002.

[124] Jing Yu, V. Anne Smith, Paul P. Wang, Alexander J. Hartemink, and Erich D. Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20 (18):3594–3603, 2004.

[125] Chiou H. Yuh, Hamid Bolouri, and Eric H. Davidson. Genomic cis-regulatory logic: Experimental and computational analysis of a sea urchin gene. *Science*, 279:1896–1902, 1998.

[126] Yu Zhang, Zhingdong Deng, Hongshan Jiang, and Peifa Jia. Dynamic Bayesian network (DBN) with structure expectation maximization (SEM) for modeling of gene network from time series gene expression data. In Hamid R. Arabnia and Homayoun Valafar, editors, *BIOCOMP*, pages 41–47. CSREA Press, 2006. ISBN 1-60132-002-7.

[127] Xiaobo Zhou, Xiaodong Wang, Edward R. Dougherty, Daniel Russ, and Edward Suh. Gene clustering based on clusterwide mutual information. *Journal of Computational Biology*, 11(1):147–161, 2004.