Programming Research Group

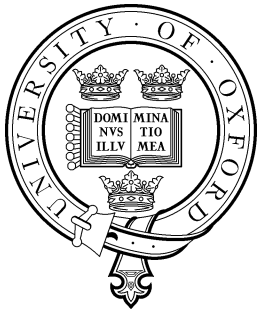
FUZZY BAYESIAN NETWORKS FOR NETWORK INFERENCE

(Transfer Report)

Christopher Fogelberg

Submitted in partial fulfillment of the requirements for
the degree of Doctor of Philosophy, Computer Science.

September 16, 2008



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

Abstract

Causal inference is a challenging and important problem in a wide range of scientific fields. Understanding how factors interact is essential in theoretical research, experimental studies and applied research and development.

Causal networks are a particular kind of causal inference problem, where a known number of factors interact with each other in an unknown way. Causal networks range in size from very small (5–10 factors) to very large (10^4). Although rigorous causal inference is possible in some situations it is much more common to infer a dependency network. This network model of the factors can then be used to guide further direct experimentation which verifies or falsifies suggested causal relationships.

A wide range of techniques have been used to infer causal and network models, including ordinary differential equations, fuzzy and logical networks, neural networks and Bayesian networks. Generally, all of these techniques are intractable on large networks (i.e. networks with more than 100 factors).

In our research we aim to develop techniques which can be used to infer detailed draft models of large causal networks. The techniques will include fuzzy Bayesian networks. Fuzzy Bayesian networks are Bayesian networks that have variables which are simultaneously uncertain and fuzzy. Thus the associated theoretical questions surrounding fuzzy probability are important in our research.

This report introduces the problem of causal network inference. It then describes the relevant aspects of a bioinformatic problem domain and previous research. We present a proposed approach to the problem of inferring large causal networks, some bioinformatic data sets that can be used to evaluate and validate this approach, and discuss the goals and contributions. Appendices present our relevant published and current work in the field so far.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Causal inference is one of the most challenging problems facing statisticians, computer scientists and philosophers in an increasingly complex and diverse world.

This chapter introduces the research program. Section 1.1 describes and conceptualises *causal networks*. Inference of them is difficult and valuable. Section 1.2 summarises the goals of the research program. Section 1.3 concludes the chapter by describing the structure of this transfer report.

1.1 The Problem

Causal inference is a common problem in physical and social science, and a range of techniques have been developed to address it. We are particularly concerned with inferring *large causal networks*. A causal network is *large* if it has too many vertices for a Bayesian network (subsection 3.3) model to be tractably inferred. This is approximately 100 vertices. Causal networks are formally described in subsection 1.1.1. Existing methods are unsuitable for large causal networks. Subsection 1.1.2 describes a bioinformatic problem domain where large causal networks are common.

1.1.1 Causal Networks

In a causal network there is a finite set of *factors*, V, and a finite set of causal *edges*, E. An edge from $v_j \rightarrow v_i$ indicates that v_j is causally potent with respect to v_i.

Each factor $v_i \in V$ is also annotated with a function, θ_i. This function specifies the functional relationship between v_i and its *parents*, the factors which are causally potent for v_i. Collectively, V and E make up η, and a causal network can be represented as a directed annotated graph: $G = \langle \eta, \theta \rangle$.

Causal inference is a special kind of *network inference*. Network inference involves finding a model of η and θ. Typically, V is known but E and θ are not. Pearl [78] is a leading expert in causal inference.

Causal inference unambiguously identifies causal relationships between components, based on subnetwork structure in a directed model. However, even if the data is noiseless, causal inference cannot be performed in every situation.

For this reason, we will focus on general network inference. Such inference finds an arbitrary member of the best-fitting *equivalence class* of models. An equivalence class is the set of models which make identical predictions on the training data.

Although they are in the same equivalence class, two members may differ over the direction of some edges. They may also "skip a generation" and directly relate a *grandparent* component to the *child* instead of relating the child to its parent; they can be different in other ways as well.

Despite these differences, identifying a model which is from the best-fitting equivalence class is sufficient. This is because such a model can be used to guide more direct experimentation which tests and verifies possible causal relationships. This is often easier and more fruitful than trying to directly infer causal relationships from a network model.

Network inference is easier than causal inference, however it is still extremely difficult. Traditional statistical techniques often give the most accurate answer. However this answer will be contingent on their assumptions (e.g. Gaussian distributions) and such techniques are intractable for networks of any size. This means that the heuristic approximations introduced by machine learning techniques are necessary.

When the data being used to infer the network is corrupted by noise or partially *covered*, previous research has found traditional statistical techniques to be even more intractable and machine learning is even more essential. Covered data is missing data.

1.1.2 Examples of Causal Networks in Biology

Genetic regulatory networks (GRN) are one kind of causal network. In a GRN, $v \in V$ are the genes in an organism, and E and θ are the regulatory relationships amongst those genes. As well as giving you brown hair or blue eyes, genes also regulate each others activity level. Historically, regulatory relationships were not considered[6], but they have become increasingly central in modern research[50; 56]. Other biological networks, such as protein-protein networks and metabolic networks[71] also exist, as do non-biological networks. In our program we focus on applying the theoretical research to GRN inference and reserve other networks for future investigation.

Biological network inference can assist research into an organism's *ontogeny* (i.e. it's developmental cycle, from birth to death), suggest genetic engineering and guide pharmaceutical research.

1.2 Goals

The central goal of our research program is to tractably infer *detailed drafts* of large causal networks when the data from these networks is noisy and partially covered. The decision to infer this *class* of model will be justified in chapter 4, when we analyse the previous research and open questions in the field.

A model of a causal network is *detailed* when it may specify a unique regulatory function for each factor. An *undetailed* model is one which assumes some factors have identical causal relationships. For example, clustering the factors and then inferring a model over the clusters creates an undetailed model.

A model is a *draft* if it uses simplifying assumptions which may not hold in the underlying network. Examples of draft assumptions are fixing an arbitrary maximum degree in the network structure, or assuming some linearity in the regulatory relationships. A *final* model does not make any such assumptions. Figure 1.1 shows the relationship amongst these model classes.

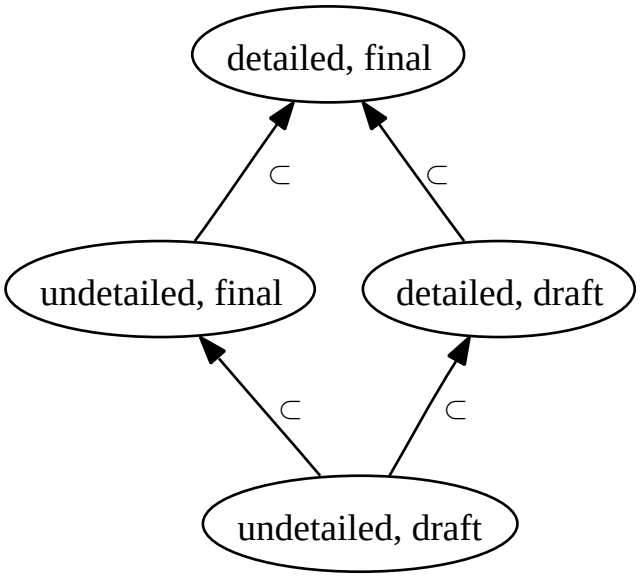


Figure 1.1: Classes of model. This figure partially orders different classes of model. If all of the models which can be represented by class C can also be represented in class C', then $C \subseteq C'$. If $C' \nsubseteq C$ then $C \subset C'$.

Thus, the techniques that we will develop must meet two key criteria. Firstly, to infer large networks, they must be more tractable than many modern bioinformatic techniques[1]. These current techniques have a different goal and are intractable on networks with more than 15–20 genes.

Secondly, the most tractable traditional machine learning techniques do not infer detailed models. For example, clustering does not, although it has been successfully applied to group genes into causally related clusters (e.g.[39; 48; 88]). This means that new techniques must be created or new ways of using existing techniques must be found that tractably preserves their detailed nature.

We expect to achieve these goals by combining *fuzzy*[8; 108; 110; 112] and *Bayesian network*[45] techniques in a novel way: *fuzzy Bayesian networks* (FBN). Along with theoretical analysis of fuzziness, probability and fuzzy probability, this will be the first major theoretical contribution of this DPhil and supports the inference of detailed models.

We will also develop an integrated sequence of machine learning techniques ("a pipeline") which take advantage of the characteristics of FBN to perform tractable detailed draft inference of large causal networks. An example of a pipeline that has been successfully used in previous machine learning research uses gradient descent to fine tune the best solutions found by evolutionary algorithms[94; 95; 118; 119]. Using a pipeline will also maximise the value of the noisy and partially covered data.

1.3 Structure

The remainder of this transfer report is structured as follows:

§ 2 introduces GRN in more detail and summarises the relevant biological characteristics.

§ 3 introduces the machine learning techniques that will be used in this research.

§ 4 describes previous research done on inference of large GRN, using the techniques described in chapter 3. It will also analyse the gaps in this research.

§ 5 proposes a research program, summarises research already done and motivates the particular techniques that are being developed.

§ 6 describes the GRN data sets that will be used in this research and summarises `GreenSim`, a GRN simulator we have written.

[1]For example, [53] is used on 5-, 9- and 20-gene networks, and [68] inferred a 10-variable network

§ 7 summarises the contribution and objectives of this research in the context of previous research. It also includes a research plan and contingencies.

Following these chapters, seven appendices are included:

A describes work done each month to date.

B is the full text of a paper presented in CIMA at ECAI. It presents our formalisation of FBN and outlines belief propagation in them.

C is the long paper version of our submission to the 2008 Oxford University Computing Laboratory DPhil Student Conference. It provides further explanation of FBN and gives more in-depth examples of belief propagation in FBN.

D presents our initial and in-progress work on the conceptual foundations of *fuzzy probability*. This work is some way from completion but represents an important theoretical aspect of our research on large causal network inference.

E is the full text of the technical report describing the `GreenSim` GRN simulator.

F describes our literature review[27], which we have published as a technical report and submitted as a book chapter[29]. The full text is not included in the transfer report due to its length (68 pages). Instructions for obtaining it are provided in this appendix.

G is the abstract of a position paper which will be further prepared for submission to a machine learning journal as a letter. It identifies an argument which is commonly misinterpreted and represents our wider appreciation of issues in the field of machine learning. Instructions for obtaining the full text are provided in this appendix.

Chapter 2

Genetic Regulatory Network Biology

This chapter summarises the key biological characteristics of genetic regulatory networks. References include our technical report, [27], and a range of other publications, including [16; 21; 22]. Readers are referred to these publications for more detail. For example, the biological details of gene regulation and transcription are described in [27, section 2].

Section 2.1 of this transfer report describes the macro-structural characteristics of GRN, and section 2.2 explains the common characteristics of individual regulatory relationships. Section 2.3 summarises the previous two sections, considers prior knowledge and argues that an organism's genome can be considered *informationally complete*.

2.1 Macro-Structural Network Characteristics

As described in chapter 1, a GRN can be modeled as a directed graph. When this kind of model is used (as opposed to, for example, ordinary differential equations[7; 9; 22; 34; 39; 79; 80; 111]) several features emerge. These include edge distributions (subsection 2.1.1), modules (subsection 2.1.2), and motifs (subsection 2.1.3).

2.1.1 Edge Distributions

A directed graph has a pair of edge distributions, one which specifies the probability distribution over the number of outgoing edges (the *out-degree*) of each vertex, and one which specifies the probability distribution over the number of incoming edges (the *in-degree*) of each vertex.

As a result of their evolutionary origins, GRN are not random graphs with uniform edge distributions, nor are they rigidly hierarchical (and consequently fragile) with, for example, Gaussian edge distributions.

Instead, GRN are *scale free*. In particular, biological research indicates that the out-degree (k_{out}) follows a power law distribution across a wide range of organisms[5; 55].

On the other hand, the in-degree (k_{in}) is exponentially distributed[5] and some of the differences between *prokaryotic* and *eukaryotic* cells are visible in this distribution. This is because prokaryotes tend to have exponential distribution parameter $\mu \approx 3$ while eukaryotes have $4 \leq \mu \leq 8$[61].

Crucially, the value of μ does not impose any limit on the maximum in-degree, $max(k_{in})$. This means that techniques which strictly limit k_{in} to some arbitrary constant (e.g. [92; 103]) compromise their explanatory value and may not be able to infer some networks.

2.1.2 Modules

A module is a group of genes which are functionally related through their phenotypic effects. As a consequence of the edge distributions, evolutionary selection (e.g. convergence[100]) and biological processes (e.g. gene duplication[84]), genes which are in the same module often share regulators and even regulatory functions. A gene which is in multiple modules is often regulated by different genes for each module[5; 52].

Modules have no characteristic size, and modules as small as 15 genes or as large as several hundred have been identified[5]. Like k_{in}, techniques which do not take this range into account (e.g. [87]) will not be able to infer some networks.

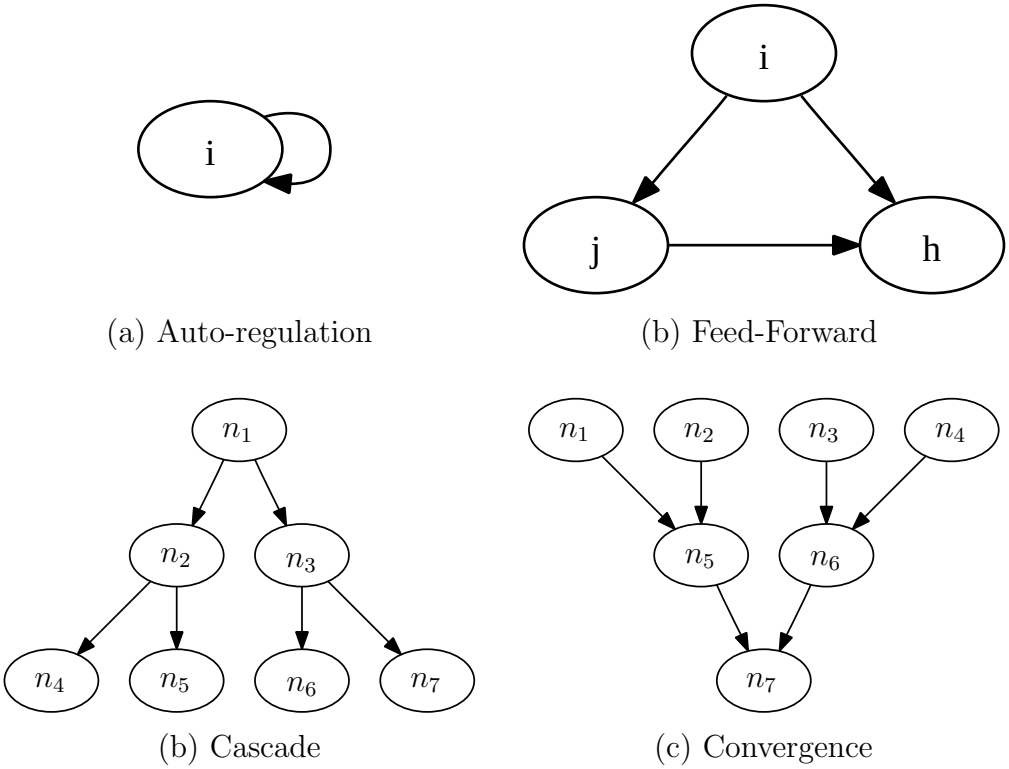
2.1.3 Motifs

Motifs are described in extensive detail in [27, subsubsection 2.1.3]; this subsection of the transfer report summarises them.

A motif is a subgraph pattern which is repeated more times in a GRN than would be expected if a graph with its edge distributions were randomly connected[56]. Four motifs have been identified: the *auto-regulatory motif*[20], the *feed-forward motif*[73], the *convergence motif*, and the *cascade motif*.

An auto-regulatory motif is where a gene regulates itself. A feed-forward motif is a triangular pattern of three genes, i, j and h. If these three genes form a feed-forward motif then it means that i regulates both j and h and that j also regulates h[4]. In the cascade motif one gene is the root of an expanding tree of regulatory relationships. Conversely, in the convergence motif, the distal regulatory effect of

a large number of genes converges on just one. Figure 2.1 is sourced from [27] and provides visual examples of GRN motifs.



(a) Auto-regulation (b) Feed-Forward

(b) Cascade (c) Convergence

Figure 2.1: Network motifs in genetic regulatory networks. The auto-regulatory, feed-forward, cascade and convergence motifs.

2.2 Micro-Structural Network Characteristics

Gene regulatory relationships can be very complex, and they are summarised in more detail in [27, subsubsection 2.2.3] and [16; 22; 109]. One reason they are complex is because the gene transcription and regulatory process has many stages, and regulation may be affected in complex stochastic and non-linear ways at each of these stages. Further, the cell type may also bias the range of regulatory functions in subtle ways. For example, inhibitory relationships are more common in prokaryotes than in eukaryotes[46]. This could be seen (loosely) as an example of "downwards causation"[76].

In general, a gene can up-regulate (*excite*) or down-regulate (*inhibit*) another gene, and it can do this weakly, strongly, or at any other level. The functional

nature of the regulatory relationship also varies widely. It may be linear, sigmoid or logically piecewise and phenotypically dependent[109]. In general, gene regulation is non-linear, although previous research[20] has linearly approximated it with some degree of success.

2.3 Summary and Discussion

In general, we have assumed that gene expression data is informationally complete, i.e. sufficient to infer an accurate model of the GRN. Although post-transcriptional regulation may affect the expression data, as can various forms of noise (see chapter 6), these factors are analogous to hidden variables in a *Bayesian network*[40]. While explicitly considering them may reduce the functional complexity of the regulatory relationships, this benefit would be counter-balanced by the increased size and complexity of the model[45, p. 29].

Tegner et al.'s [103] successful inference of a network with known post-transcription regulation and protein-protein interactions using noisy gene expression data provides evidence that expression data is informationally complete.

Note that, we have not considered the inclusion of prior knowledge when doing GRN inference. This will be clear when we describe our planned research in chapter 5. Although prior biological knowledge has been used in some past research (e.g. [4; 9; 40; 80; 116]) we do not consider it. This is because the focus of our research is on developing machine learning techniques for general inference of large causal networks. Different prior knowledge could be incorporated in different ways into different types of causal networks. Also using the somewhat naive approaches used in past bioinformatic research would not be a general contribution to computer science. Extending and generalising these approaches would be a complete program of research in its own right.

Chapter 3

Machine Learning for Causal Networks

Three areas of research in machine learning are central to our proposed course of research, which is causal inference in large networks. This chapter provides the relevant background on fuzzy theory (section 3.1), clustering (section 3.2) and Bayesian networks (section 3.3). Readers are referred to [27] for detail on other areas of research relevant to causal inference.

3.1 Fuzzy Theory

The variables in machine learning are usually either intrinsically continuous or intrinsically discrete. The temperature is an example of an intrinsically continuous variable. Whether Alice could decode Bob's message is discrete: either she did, or she didn't.

Understanding what a particular value means or how important it is usually needs a lot of background knowledge. A temperature of 60 °C is extremely hot for a summer's day in England, but cold for a turbine engine.

A common way of making a variable which has a continuous value more comprehensible is to discretise it. For example, the temperature today may be discretised into `hot`, `tepid` and `cool` according to a classification function. This is known as a *hard discretisation*.

One problem with hard discretisations is that they discard information, and values which are near the borders are often misrepresented. For example, all temperatures over 30 degrees may be discretised as `hot`, but this discretisation does not distinguish between 30.1 °C and 60 degrees.

Fuzzy theory was developed by Zadeh [117] in 1965 as a way of addressing this representational problem. Just as a continuous variable can be represented by a

classification function and the associated set of hard discretisations, a continuous variable can also be represented by a set of *fuzzy sets*, and each fuzzy set has a *membership function* associated with it. In our ECAI paper we refer to a fuzzy set as a *fuzzy value* (FV). This is explained and justified in our in-progress work on fuzzy probability (appendix D).

Given such a representation of the variable, a particular value for that variable is represented by its values *degree of membership* (denoted μ, $\mu \in [0, 1]$) in each fuzzy set. This is calculated according to each fuzzy set's membership function.

For example, rather than saying that 30.1 °C is `hot`, we could *fuzzify* the continuous value as $[hot_{0.6}, tepid_{0.4}, cool_{0.0}]$, where the subscripts denote μ in each fuzzy set. Typically $\sum \mu = 1$. Such a fuzzification concisely expresses the fact that 30.1 °C is quite hot, but not unreasonably so.

Although fuzzy variable states may increase the complexity of a machine learning problem relative to discrete values, this is counter-balanced by their greater comprehensibility to people and some robustness in the face of noisy data.

Fuzzy values are particularly appropriate for cases where a variable cannot be expressed as just one continuous value or a single discrete value. For example, a child with two English grandparents, one German grandparent and a French grandparent is not just English or 0.5 English. Instead they are $[eng_{0.5}, ger_{0.25}, fre_{0.25}]$. This is often the case when a variable represents an aggregate or average, as temperature and genealogy do. Readers are referred to appendix D for a more detailed discussion.

Because of the way it naturally and comprehensibly represents many real world phenomena, fuzzy theory has found wide use in artificial intelligence. Its uses include clustering[8; 19; 59; 69; 108; 112], production system rule inference[101] and causal network inference[10–12].

3.2 Clustering

A clustering algorithm is made up of a *method* and a *distance measure*. One use is clustering for feature selection, which divides the *features* that specify each *sample* into a smaller number of representative *clusters*.

Subsection 3.2.1 describes distance measures and subsection 3.2.2 describes some types of clustering methods. Subsection 3.2.3 concludes by discussing *covers* and *biclusters*, special types of clustering method. Although there has been a lot of research on fuzzy clustering (e.g. [66] and above references) we do not discuss it in this transfer report, to reduce its length. Unpublished reports[25; 26] with more detail on fuzzy clustering are available upon request, and it is also discussed in our technical report[27].

3.2.1 Distance Measures

A distance measure is used to describe how similar (equivalently, different) two samples are. A clustering method uses these distance measurements to group the features into clusters.

The *Euclidean distance* (equation 3.1) and derivatives of it such as the *Manhattan distance*[58] and *Mahalanobis distance*[17; 67] are the most commonly used distance measures.

$$d_{ij} = d_{ji} = \sqrt{\sum_{m=1}^{M} (i_m - j_m)^2} \tag{3.1}$$

A feature's distance from a cluster may be calculated by its maximum, minimum, mean or median distance from a member of the cluster.

Mutual information is a a distance measure which uses the *Shannon entropy*[90] or a variant of it to measure the similarity of two features. If two features are very similar then, when you know one, learning the other will not give you much new information, and the two features will be clustered together.

3.2.2 Types of Method

A clustering method can be either *supervised* or *unsupervised*, and it can either *partitionally* or *hierarchically* cluster the data.

A supervised method uses some fixed number of *fingerprints*, which are specified before the method is executed. A fingerprint forms the nucleus of each cluster, and features are clustered relative to them. An unsupervised method does not use fingerprints, and the gene clusters emerge solely from the data. The number of clusters is not necessarily explicitly defined in advance. For this reason unsupervised methods can be more useful when the characteristics of the data are not well understood; supervised clustering may give better results or find the clusters more quickly.

A hierarchical clustering method (e.g. *SOTA*[3], or Horimoto and Toh's [48]) can be either *agglomerative* or *divisive*. In a divisive method all features are initially grouped into one cluster, then the least similar features are iteratively broken off into independent clusters. This results in a *dendogram*, a tree of clusters. The final set of clusters can be determined using an inter- or intra-cluster validity criterion on the tree. An agglomerative method is very similar but builds the dendogram from the bottom up, rather than from the top down.

In a partitional method (e.g. *C-means clustering*[2; 65]) each feature is placed into a cluster independently of the other features. Following this the cluster centres are updated and the process is iterated until cluster location and membership

converges. Most partitional methods are supervised; most hierarchical methods are unsupervised.

3.2.3 Covers and Biclusters

In a classic clustering algorithm, a feature is placed into one and only one cluster. Likewise, while a fuzzy clustering algorithm may give a single feature partial membership in several clusters, the feature's membership across all clusters sums to 1.

However, if a a feature can be a member of more than one cluster then the result is a cover. This is a different use of the term "cover". It is also used to describe missing data points (subsection 1.1.1). Research on clustering covers includes [4] and [91; 102]. Biclustering will be discussed shortly, [91; 102] are also examples of biclustering algorithms.

In certain situations a cover may have advantages over a cluster. This is particularly true when a single feature may have total membership in more than one group, as is sometimes the case in biological organisms (subsection 2.1.2).

For example, the gene X may regulate embryonic development in response to external chemical gradients. It may also manage parts of the *cell development cycle*[99] by regulating genes which control cell division and contribute to other processes as well. If X is forced into just one cluster then any model inferred over the clusters will be incomplete.

Another relatively modern technique is biclustering. A recent survey of biclustering for bioinformatics is [66]. Biclustering can be used on any data set in which either the features can be grouped across samples, or the samples can be grouped across features. Rather than clustering by either just the samples or just the features, a biclustering simultaneously selects a subset of the samples and a subset of the features within those samples. The subsets are selected so that the features in the selected samples are close to each other, and so that the samples are also close to each other with respect to the selected features. In general, finding an optimal biclustering is NP-hard[114].

3.3 Bayesian Networks

A Bayesian network (BN) is a factorisation of a *joint probability distribution*. A joint probability distribution describes the uncertain state of a finite set of variables, where a variable represents a factor, like the expression level of one gene.

Like a GRN, a BN can be formally represented as a directed graph, $G = \langle \eta, \theta \rangle$, where η is the set of variables (factors) and the edges (conditional dependencies)

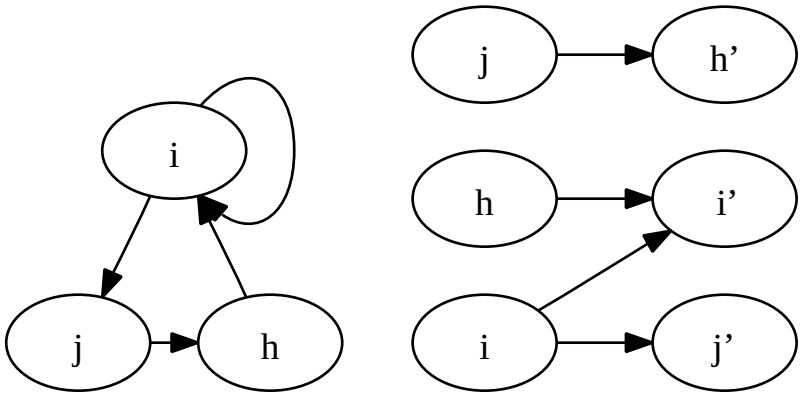
amongst them, and θ is the set of probability distributions, one for each variable. Each $\theta_i \in \theta$ is conditional on the parents of the variable i.

3.3.1 Dynamic Bayesian Networks

As explained in [27, subsubsection 5.4.1], a BN must be acyclic. However, many causal networks are mutual. That is, A affects B, but B also affects A.

Consider a *hidden Markov model*[81] (HMM). Such a model can concisely represent an arbitrary number of time steps because it can be *unfolded*.

By representing the same underlying factor at time t and time $t + 1$ with two different variables, cycles can be avoided. Because the distributions are stationary, the model only needs to be unfolded one step, and likewise in a BN. An unfolded BN is called a *dynamic Bayesian network* (DBN)[32; 75] and a *hidden Markov model* (HMM) is a special case. Figure 3.1 provides an example.



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

Figure 3.1: A cyclic BN and an equivalent, acyclic, DBN. The prior network[32] is not shown in this diagram. x' is x's temporal successor. This figure was sourced from [27].

3.3.2 Learning Bayesian Networks

A wide range of techniques exist for inferring η and θ given observations of the factors which underlie the variables. The observations may be noisy or incomplete, i.e. partially covered. Readers are referred to [27, section 5.4], and also [45; 74] and [32; 38] for an overview.

With certain assumptions and potentially noisy but entirely uncovered data, θ can be calculated analytically. More generally, the same techniques can be used for

η and θ inference. Machine learning algorithms which can be applied to network inference include greedy hill climbing with random restarts[82], the EM algorithm[18], *simulated annealing*[72] and *Markov Chain Monte Carlo* (MCMC)[43].

3.3.3 Bayesian Network Complexity

Unfortunately, Bayesian networks are plagued by algorithmic inefficiency, which makes learning large BN very difficult. This is due to cycles in the underlying undirected graph[14; 103]. DBN inference is even harder because the number of variables is doubled[75]. This means that the average size of BN GRN models (≈ 100) is much less than the number of genes in an organism (e.g. ≈ 6200 in *S. cerevisiae*). However, even models as simple as a Boolean network are exponential in $max(k_{in})$[61]. The problem of complexity is one that must be acknowledged and considered.

Chapter 4

Previous Research

There are three key areas of previous research that are centrally relevant to this research program. Readers are referred to [27] for an in-depth review of the wider field of GRN inference.

Section 4.1 describes the first, model inference for large GRN. Section 4.2 briefly summarises some recent work using fuzzy logic to infer causal networks, and section 4.3 describes the use of BN and DBN to infer GRN. Section 4.4 concludes the chapter and analyses some of the gaps in previous research.

4.1 Model Inference for Large Genetic Regulatory Networks

This section describes techniques used to infer models of large GRN. At the simplest level, a wide range of clustering techniques have been used to organise genes into modules. Subsection 4.1.1 briefly describes some of these. Subsection 4.1.2 describes two other techniques that infer models of a large GRN. These models are undetailed draft models (figure 1.1), but they are more specific than clustering alone.

4.1.1 Clustering

A lot of research on clustering gene expression data has been done, and [27] (our technical report) summarises some of this research. The research has used a mix of hierarchical, partitional, supervised and unsupervised clustering methods. Past research which has clustered gene expression data includes [4; 48; 87; 104]. Several recent surveys of gene expression clustering include [3; 23; 89; 120]. Biclustering has also been used for gene expression clustering, and [66] surveys this.

Horimoto and Toh [48] note that clustering gene expression data seems appropriate and fruitful. They found that nearly 20% of gene pairs were similarly expressed at a 1% significance level.

Algorithmic robustness in the face of noisy and missing data is important when inferring GRN. Jiang et al. [52] considers these problems, and [107] outlines several techniques for dealing with missing data.

4.1.2 Other Techniques

Two examples of research are summarised in this subsection. This is because they appear to be the only examples of large GRN inference which do not just cluster the genes into modules.

The first is the work carried out by Toh and Horimoto[48; 106], which clustered 2467 genes hierarchically and cut the dendogram according to a pre-specified maximum intra-cluster variance so that 34 clusters were formed. Following this, the conditional regulatory relationships amongst the clusters of genes were identified using a *graphical Gaussian model*.

This work is interesting because it is an example of a set of complementary machine learning techniques arranged into a sequence and being used together to find a better model than each technique could alone.

Bonneau et al. [9] describes the `Inferelator`, an algorithm that was used to infer a large undetailed draft GRN for the *Halobacterium NRC-1*. Dimensional reduction to make the problem tractable was achieved by specifying a list of 82 module regulators in advance and by biclustering the data and inferring bicluster regulators.

`Inferelator` finds kinetic (difference) equations with floors and ceilings, as simulated in `GreenSim` and `GeneSim`[115; 116]. Some regulatory non-linearities (no more than second order) can also be recovered. Further, the algorithm can make use of both time series and equilibrium gene expression data.

In general, `Inferelator` is interesting for two reasons. Firstly, like Horimoto and Toh's [48], it infers a more detailed model of the entire GRN than clustering does alone. Secondly it can also incorporate non-linearities into its model. This makes it a modern example that our research can be compared to.

4.2 Fuzzy Causal Inference

Recent work by Cao et al.[10–12] has used *fuzzy logical networks* (FLN) for GRN inference. A FLN is a generalisation of a *Boolean network*[16; 54; 55; 61; 64; 92] to one with fuzzy variable states. The same sorts of inference and analysis (e.g. attractor basins) can be done for FLN as for Boolean networks.

Like BNs and Boolean networks, a FLN can be represented by η and θ, where each $\theta_i \in \theta$ specifies a *fuzzy logic function*. A fuzzy logic function is analogous to a Boolean logic function, such as $A \vee B \wedge C$, but it uses fuzzy connectives instead[93; 101; 117].

In one case[12], model inference using 115 time series data points for each gene was carried out for 407 known cell-cycle genes from *Schizosaccharomyces pombe*. The 407 genes were preprocessed and filtered so that only 286 genes remained. Overall, 125 regulatory relationships were identified, of which 59 had been previously verified, 47 were novel and unknown and 19 were manually classified as dubious.

4.3 Bayesian Networks for Genetic Regulatory Networks

Relevant GRN inference research which uses BN falls into two key areas. The first of these is the comprehensive project carried out at Duke university. Subsection 4.3.1 describes this research. The second is research which focuses on the sparse nature of GRN, and includes work by Friedman et al. [33], Murphy [74] and Chan et al. [13].

4.3.1 Duke University Inference of Songbird Singing

The research carried out at Duke University by Jarvis et al. (including [51; 96; 97; 115; 116]) is the most comprehensive program of GRN inference known to us. The research spans GRN inference and neurological mapping. Due to technological limitations the data is simulated using `BrainSim`[96] and `GeneSim`[115; 116], the latter a precursor of `GreenSim` (section 6.2). DBN are used to infer models of the GRN.

The simulated training data was interpolated to maximise its informativeness and this was found to improve the quality of the inference. An *influence score* was developed[116] to aid in easy understanding of inferred GRN. This allowed complex relationships amongst pairs of genes to be characterised by a single number.

The work had two important weaknesses. Convergent motifs were not easily identified due to the small quantities of training data[51], and because only simulated data could be used it is difficult to evaluate the inference algorithms performance on biological data. Crucially, it is also intractable on networks with more than 20–100 genes[96; 116].

4.3.2 Sparse BN Algorithms

Friedman et al. [33] present an algorithm for learning a Bayesian network which is *sparse*. A sparse BN is a network which is much closer to the *empty* (edgeless) network than the *complete* (fully connected) network. The algorithm works by heuristically restricting the search for a graph so that only the most Shannon-informative factors for the factor x are eligible to be parents of x. This candidate-restricted search is carried out iteratively and the set of candidates changes each iteration.

Murphy [74] considers the related problem of *data sparsity*. Because there are so few observations, relative to the dimensionality of the network to be inferred, the problem is under-constrained. This means that there will not be a single best model, and when there is such a model is not likely to be right. One way of addressing this is by approximately integrating over the model space. Murphy outlines this integration over networks in their report.

Finally and most recently, Chan et al. [13] manually selects 16 genes known to be relevant and uses SBL[105], a Bayesian technique "designed to sparsify model parameters"[13] to infer a set of ODEs amongst the 16 genes.

4.4 Conclusion and Analysis

In general, it is clear that BN (primarily DBN) have been successfully used for causal network analysis, especially of GRN. Furthermore, fuzzy techniques have also been found to have a number of advantages. Stochastic and differential equation models cannot be as large as Bayesian models. FLN and Boolean networks have not been much larger[27] and do not possess the advantages of Bayesian approaches, such as statistical rationality and the capacity for rigorous causal inference[78].

Network inference has been one of two types. In the first type, undetailed models are found by clustering the data. Sometimes a more specific (still undetailed) model is found by inferring relationships amongst the clusters.

In the second type, smaller, more detailed models are inferred using a subset of genes already known to interact or act as module regulators. For example, most BN and DBN research infers networks of less than 100 genes (*S. cerevisiae* has more than 6000), and even Cao et al.'s [12] work only considered networks of $N \leq 300$.

Artificially restricting the search in this way is often necessary, however it intrinsically biases the results and is unhelpful when there is no prior knowledge to guide the inference. Even though `Inferelator`[9] may seem a counter example, it is important to remember that it also needed a pre-specified list of 82 module regulators and that it found undetailed module-regulatory relationships, not detailed gene-regulatory relationships.

Thus, the unrestricted inference of large causal networks remains an open research question. Furthermore, the need for it is made clear by the recent development of algorithms like `Inferelator`[9] and the wide range of specialised clustering algorithms which have also been used.

Chapter 5

Proposed Approach

This chapter summarises the proposed program of research in my DPhil. Section 5.1 gives an overview of the program, and sections 5.2–5.3 describe some of our initial work. Section 5.4 summarises the motivation and contribution.

5.1 Overview of Proposed Approach

The core goal of this research is the development of techniques for inferring large causal networks. As noted in section 4.4, previous research has been either intractable on large GRN, or created models which are not detailed.

Research by Toh and Horimoto [106] and Bonneau et al. [9] represent important attempts to address this problem, and use an integrated sequence of machine learning techniques (a pipeline) to infer more specific models of large networks than previous research. However the resulting models are still undetailed, as defined in section 1.2. This is because the dimensional reduction discarded information about the factors and treated them identically.

For this reason, we suggest the following pipeline:

1. Preprocess the data.
2. Find a *fuzzy cover* over the data.
3. Infer a fuzzy Bayesian network (FBN) over the covers.
4. De-fuzzify the covers to obtain a detailed draft model of the complete network.

The data can be preprocessed and normalised so that it has, for example, mean 0 and is $\log_2$ transformed. Furthermore, genes (factors) which do not vary much can be removed before finding a fuzzy cover. This is because it would be very difficult to determine which other genes they regulate, or which genes regulate them[23; 99].

Following this, a fuzzy cover of the data can be found. A fuzzy cover is a clustering which allows a gene to be a member of more than one cluster. This is a different use of the term "cover"; it is also used to describe missing data. Unfortunately the terms is used in both ways in the literature. Because it is a fuzzy cover, a factor can be a partial member of more than one cluster, and the factor's total membership in clusters does not need to sum to 1. The motivations for using a cover and other essential characteristics of it are detailed in the next section, section 5.2.

Next, an FBN amongst the covers could be inferred to determine the macro-structure of the network. This will not be significantly more complex than inference of a classic Bayesian network and can use the same techniques with only minor modification.

Finally, the covers could be linearly de-fuzzified to provide a detailed draft of the complete network. Because a detailed draft model has been found to guide further (final-type) inference, standard techniques such as regression and correlation analysis could now be applied and represent the natural next step.

5.2 Fuzzy Covers

Clustering and covering algorithms are an enormous field of research in their own right. Rather than developing an entirely new algorithm in this area we will select and modify an existing algorithm.

The algorithm that we use should be a cover rather than a cluster because any one gene can play multiple roles, even within a single phenotypic process, and it is usually regulated by different genes for each of these roles[4; 48].

So that it is as appropriate to the problem as possible, the selected method and distance measure should also possess a number of characteristics.

Because the covers should be defuzzifiable, the selection of the distance measure is very important. In particular, since the FBN inference process will specify the same regulatory function for every member of the cover before de-fuzzification, it is important that the members of a cover are all positively correlated. This suggests the Pearson correlation or the *Jackknife correlation*[52] as the distance measure. Defuzzification could be achieved by taking a membership-weighted sum of the regulatory functions for the covers a gene has non-zero membership in.

The pipeline will be most general and useful if the clustering is unsupervised. This is because supervised clustering often requires substantial prior knowledge about the network.

The method should also be robust in the face of noise (the magnitude of the noise is not clear, [57] and [23, p. 14865]) and covered data. For that reason the data must be preprocessed or the method must handle missing data robustly.

In the most general case it would also be important to consider the fact that a gene which is in more than one module may have different regulators in different phenotypic contexts. This means that the effect of some regulators could be inconsistent and contradictory and suggests biclustering the training data, as was used in [9]. However, because we are using time series data from the cell development cycle, as is discussed in section 6.3, this consideration should not be relevant.

These factors and potential algorithms are discussed in more detail in another unpublished report we have written [26].

5.3 Fuzzy Bayesian Networks

Readers are referred to our ECAI paper ([30] or appendix B) and our 2008 Oxford University DPhil Student Conference submission (appendix C) for a full introduction to the formalisation of FBN. Subsection 5.3.1 describes what a FBN is, subsection 5.3.2 broadly explains model inference for FBN, and subsection 5.3.3 introduces the notation we have developed[1]. Appendix D goes on to introduce our conceptual work on fuzzy probability. This theoretical work is initial and still in-progress, but it is becoming an essential foundation for fuzzy Bayesian networks.

5.3.1 What is a Fuzzy Bayesian Network?

A FBN is just a Bayesian network with variables which have fuzzy states (section 3.1). These states are simultaneously uncertain as well. Such a network combines the advantages of a fuzzy representation and Bayesian networks.

Furthermore, the formalisation that we have developed allows all existing BN inference and learning algorithms to be used on FBN. Although belief propagation in a FBN has different and subtle considerations compared to a classic BN, a key advantage of our formalisation is that most existing research can be transparently reused on FBN.

Note that FBN are independent of the pipeline described in section 5.1. Although such a pipeline may be necessary for large causal network inference with FBN, a FBN can be used for any problem which can be conceptualised as having fuzzy variable states.

5.3.2 Fuzzy Bayesian Network Inference

Because FBN are a generalisation and modification of classic BN, all existing BN inference techniques can be used with only trivial modification for FBN.

[1]Note that the notation used for probability distributions [30] is subtly different and less clear than the notation described in appendix C and this chapter.

Techniques for BN inference are summarised in [27] (our technical report) and discussed in more detail in [32; 38] and [45; 49]. BN inference is also mentioned in subsection 3.3.2. This subsection outlines the general process and the relationship to the training data, described in chapter 6.

FBN inference consists of searching for a good model. Typically a model with random η and a corresponding θ is generated as a starting point. Any search algorithm can then be used. These include hill climbing[82], *expectation maximisation*[18], *simulated annealing*[72] and *Markov Chain Monte Carlo*[43] (MCMC). Each of these is summarised in the associated references and [27].

While searching, a prospective model must be scored relative to the previous model. A wide range of scoring methods exist, including the likelihood of the data given the model, the posterior probability of the model given the data[115], and approximations and reformulations of the posterior, such as the *Bayesian Information Criterion*[86; 115; 116] (BIC), the *minimum description length*[37; 62], and the Local Criterion[45]. These are explained in the associated references; they and others are also summarised in [27].

5.3.3 Fuzzy Bayesian Network Notation

A FBN combines fuzzy knowledge of a variable state with uncertainty. Although these two properties of a variable may look similar they are distinct and cannot be mixed. For example, if a variable represents a committee, then a committee in which no one can make up their mind is very different from a committee in which half of the people are absolutely fixed on A and half of the people are absolutely fixed on B.

Similarly, there is a difference between diving into a pool that is a 50-50 mix of boiling water and ice water, and diving into a pool that has a 50% chance of being boiling water and a 50% chance of being ice water.

For this reason we have developed notation for multinomial FBN that ensures they remain visually distinct. A variable's degree of membership (μ) in a set of fuzzy sets (fuzzy values) is denoted using square brackets and μ is subscripted. For example, equation 5.1 denotes a variable X which has 0.3 membership in fuzzy value *hi* and 0.7 membership in fuzzy set *mid*.

$$X = [hi_{0.3}, mid_{0.7}] \tag{5.1}$$

A multinomial probability distribution (PD) is denoted using curly brackets, and the probability of each value is subscripted. For example, equation 5.2 specifies a PD. A sample from this PD has a 0.2 probability of being *lo*, a 0.3 probability of being *mid* and a 0.5 probability of being *hi*.

$$Y = \{lo_{0.2}, mid_{0.3}, hi_{0.5}\} \tag{5.2}$$

Finally, a variable may be simultaneously fuzzy and uncertain. This is denoted by combining the notation for fuzzy sets and PDs, and treating μ-annotated PDs as fuzzy sets whose value varies probabilistically. For example, equation 5.3 specifies the fuzzy state of Z.

$$Z = [hi_{0.3}, \{lo_{0.2}, mid_{0.4}, hi_{0.4}\}_{0.7}] \tag{5.3}$$

Although the question of fuzzy probability is very complex (see appendix D) we have assumed that samples from a FPD in FBN are *monolithic*, i.e. made up of just one value, and that the associated variable has the same μ in the sample as it does in the FPD.

This means that we know that Z will be either $[hi_{0.3}, lo_{0.7}]$ with probability 0.2, $[hi_{0.3}, mid_{0.7}]$ with probability 0.4, or $[hi_{0.3}, hi_{0.7}] = [hi_{1.0}]$ with probability 0.4.

5.4 Summary

In conclusion, the key advantage offered by FBN over previous approaches is as a part of a synergistic inferential sequence which uses a fuzzy cover to reduce the dimensionality. However, they may also bring the advantages of a fuzzy representation to the inference of small networks.

Furthermore, FBN can easily take advantage of previous BN search and scoring techniques.

Bayesian networks are fast becoming a standard model representation and have a number of advantages, including maximising the value of the training data and a capacity for modeling non-linear systems.

The key hypothesis that this research will test is the value of combining these two techniques and the applicability of the combined formalisation to the inference of large causal networks. It will also consider the theoretical and conceptual foundations of fuzzy probability for FBN.

Chapter 6

Data

Chapter 5 outlined the machine learning techniques which we will use to address the problem of large network inference, and some of the theoretical considerations. This chapter describes the data sets that we will use to evaluate and validate them.

Section 6.1 summarises the kind of data which is available. Section 6.2 very briefly summarises our GRN simulator `GreenSim`, and section 6.3 describes the biological data sets we will use in our research.

6.1 General Characteristics

Four kinds of data can be used to infer GRN. These are gene expression data, perturbation experiments, phylogenetic data and biochemical data[27, subsection 2.4].

Perturbation data is a special type of expression data. Unfortunately it is difficult to get meaningful experimental quantities of perturbation data, unless the organism is already well understood. Phylogenetic data and biochemical data is not currently available in the quantities or forms necessary to support machine learning. Some bioinformatic research has combined this data with expression data (e.g. [4; 40; 113]). However, such research has always been aimed at finding an answer to a single biological question. As with prior knowledge, we will not consider using multiple types of data in our research. This is because we aim to develop general machine learning techniques and then illustrate their applicability to problems in bioinformatics, rather than to answer particular bioinformatic questions. Using prior knowledge in a general and principled way is also a substantial research problem in its own right.

6.1.1 Gene Expression Data

Gene expression data measures how frequently a gene is transcribing protein. As transcription activity cannot be measured directly, the concentration of mRNA is used as a proxy.

As described in section 2.3, previous research suggests that non-perturbation expression data is informationally complete for GRN inference. This is despite any *biological noise*[77], *technical noise*[57] or post-transcriptional regulation.

6.1.2 Time Series Gene Expression Data

Gene expression data is discussed extensively in [27]. For several reasons, we have selected time series gene expression data.

Time series gene expression data is expression data collected as a cell undergoes some phenotypic process, such as the cell development cycle[99], *glycolysis* metabolism[87] or amino acid metabolism[4]. The data is usually calculated from some mutant genotype which allows the process to be arrested by a change in phenotypic contexts. This allows microarrays to be done at particular sequential points in time.

An advantage of time series expression data is that the temporal arrow shrinks the size of the equivalence class which we infer. The temporal arrow provides information, and this means that fewer models fit the data.

In addition, genetic regulatory networks can be exceptionally complex[24]. Regulatory relationships are often non-linear and can be conditional Boolean functions as well[88]. See also chapter 2. These factors combine to make inference of the complete regulatory function in all phenotypic contexts exceptionally difficult. By using time series data we ensure that the data comes from one or a small number of phenotypic contexts. This should reduce the conditional Boolean nature of the function and simplify the problem to one of non-linear function inference with noisy and missing data. Biclustering[27; 66] is one data pre-processing technique which could be used in the general case.

The specific biological data sets which have been sourced are described in section 6.3. The next section, section 6.2, summarises `GreenSim`.

6.2 Simulation with GreenSim

This section summarises `GreenSim` (the "GRN Simulator"). Data simulation is necessary to test and to validate the machine learning techniques across a suitably wide range of situations. Readers are referred to the associated technical report (appendix E or [28]) for more detail.

`GreenSim` is written in MatLab and is modular and easily customisable. In addition, in light of the discussion in section 6.1 and chapter 2, it has been designed to be as biologically accurate as possible.

6.2.1 Architecture

The simulation of a GRN can be divided into four stages:

1. Graph generation
2. Function generation
3. Sample generation
4. Sample corruption

These stages are described in the four following subsections. General functions that allow networks and sets of samples to be written and read from disk are also included in `GreenSim`.

6.2.2 Graph Generation

The first stage in simulating a GRN is generating a realistic network structure, i.e. the unannotated edge matrix which specifies the causal potency of each gene. This is done in three steps.

First, the in-degree of each gene (the number of other genes it is regulated by) is sampled from a geometric distribution. Second, the out-degree of each gene (the number of other genes it regulates, possibly including itself) is determined. This results in two sets of *half-edges*.

In the third step the in half-edges and the out half-edges are matched. Because the out-degree distribution is a power law it can be linearly scaled so that there are an equal number of in and out half-edges.

This generates realistic modules and motifs; they are discussed in [4; 20; 27; 28; 56; 73]. An example of a network is shown in figure 6.1. Networks of 10^4 genes can be generated with `GreenSim`.

6.2.3 Function Generation

Although past simulators (e.g. `GeneSim`[115; 116] and Kyoda et al.'s [60] unnamed simulator) have been essentially linear[1], `GreenSim` incorporates general non-

[1] As noted by Alexander Hartemink in personal communication following the development of `GreenSim`, the capped minimum and maximum expression levels for each gene introduce non-linearities in the edge cases.

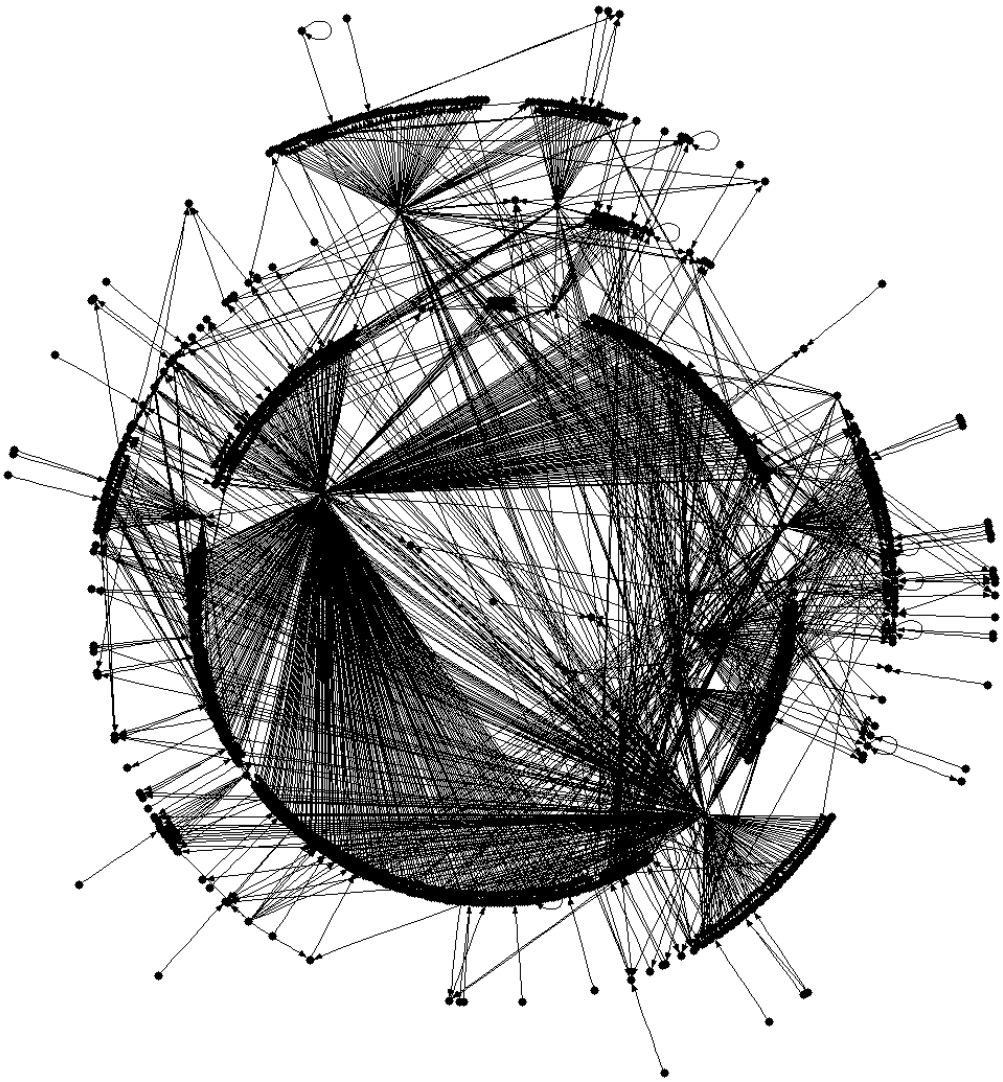


Figure 6.1: A `GreenSim` network. This figure shows an example of a 1000 gene network, generated by `GreenSim`[28]. Networks with as many as 10^4 genes can be easily generated. This example shows a realistic distribution of module size and the existence of motifs and other features, as discussed in chapter 2.

linear regulatory relationships as well. `GreenSim` could be seen as an extension of `GeneSim`. Readers are referred to the technical report (appendix E or [28]) for detail.

The regulatory effect of one gene or one pair of genes is selected uniformly at random from some range (e.g. $[-0.2, 0.2]$), where positive values represent excite-

ment and negative values represent inhibition, and the magnitude is the strength of the regulatory relationship. No single gene or pair of genes can affect the expression level of another gene by more than this in one time step.

6.2.4 Sample Generation

Using some column vector with one entry in the range $[0, 1]$ for each gene, users can draw n samples from the network with an increment of δ regulatory steps in between each sample. Because the range of the regulatory effect (subsection 6.2.3) is also parameterised, users can combine different regulatory ranges and different values of δ to simulate a range of realistic data collection technologies. Biological noise and its effect on expression levels is also simulated.

6.2.5 Sample Corruption

Once a sample has been generated it can be corrupted with technical noise. `GreenSim` models three kinds of technical noise: *value noise*, *spot noise* and *span noise*. These types of noise are described in more detail in [28] (appendix E).

6.3 Biological Data

Three different sets of time series data sets have been selected for use in this research. Each describes the cell development cycle (CDC) of either *Saccharomyces cerevisiae* or *Schizosaccharomyces pombe*, two species of yeast.

The CDC is a series of phases that a eukaryotic cell goes through as its DNA is duplicated and the cell reproduces. Understanding the CDC is important to fully understand the characteristics of a number of diseases, particularly cancer[83].

Subsection 6.3.1 further justifies the selection of this kind of biological data set and species. Subsections 6.3.2–6.3.4 describes each of the data sets, and subsection 6.3.5 summarises their overall characteristics.

6.3.1 Advantages of Yeast

A key advantage of yeasts is that they are very well studied model organisms. This means that primary literature can be used to validate the effectiveness of new research techniques on biological data. Furthermore it also means that data sets are more plentiful. This allows more informed analysis and comparison.

Yeast is such a well studied organism because it is single-cellular but eukaryotic. Eukaryotes are more complex than prokaryotes, and all multicellular life is eukaryotic. Eukaryotes usually have more complex regulatory functions.

For these reasons, yeast data sets represent an ideal source of data to validate new research into machine learning for regulatory and causal network inference.

6.3.2 Eisen

The classic Eisen et al. [23] data set consists of 6 *S. cerevisiae* time series. *S. cerevisiae* is known as "budding yeast". This is because it often reproduces through sporulation. Sporulation is a *meiotic* (recombinative) process which produces a daughter cell of smaller size than its parent.

The time series in Eisen et al. [23] are as follows.

- ***cdc-α.*** An example of the mitotic CDC. Samples were placed within the CDC using α factor arrest. This data set has 18 samples (time points).
- ***cdc-15.*** An example of the mitotic CDC. A mutant CDC 15 was used to control the percent proportion of the CDC (p_{CDC}) that had been completed at the point of each sample. This data set has 25 samples spanning a period of 280 minutes.
- ***Elutriation.*** Of the CDC. This data set has 14 samples evenly spread across 6.5 hours.
- ***cln-3* and *clb-5* induction.** Brief snippets of the CDC when the expression levels of *cln-3* or *cln-5* are artificially perturbed. This data set has one *cln-3* sample at 30 and 40 minutes and one *clb5* sample at 40 minutes.
- **Sporulation.** Gene expression levels as the cell undergoes meiosis. This time series is actually three separate series: 7 samples (11 hours), 3 samples (9 hours) and 3 samples (unknown, this sub-series is based on the period of *ndt80*, which controls exit from meiosis). Of these only the first series appears to have any substantial positive correlation between samples. For this reason we suspect inference over the others would be very difficult.
- **Diauxic Shift.** When a cell metabolises sugars it concentrates on the most easily metabolised sugars (such as glucose) first. As it transitions to metabolising the other sugars there is a pause while the cell transcribes enzymes for their metabolism. This is the *diaxuic shift*. This time series has 7 samples across a range of of presumed-sequential concentrations of glucose, from 19.0 g/L to 0.0 g/L.

Each of these data sets is for 6220 genes. The gene expression values are presented as the $\log_2$ of the ratio of the expression in the experimental condition versus the expression of the gene in a neutral control condition[23].

6.3.3 Spellman

Like the Eisen data set, the Spellman et al. [99] data set has four main time series from the *S. cerevisiae* cell cycle:

- ***cdc-15.*** A temperature sensitive *cdc-15* mutant was used to collect 24 samples over a 4 hour, 50 minute period. Samples were taken once every 10–20 minutes.
- ***cdc-α.*** An initial sample of cells was suspended by removal of the α factor. Following its reintroduction, one sample was taken every 7 minutes for just under 2 hours. In total, 18 samples were taken.
- **Elutriation.** 14 samples were collected using elutriation, one every 30 minutes during the CDC.
- ***cdc-28.*** The Cho et al. time series is detailed in [15] and included in Spellman. This time series used temperature-arrested *cdc-28* to pause the CDC while samples were taken. 17 samples were collected, one every 10 minutes. This means that the samples span nearly two complete cycles.

Although the full data set includes data from 6177 genes, Spellman et al. aimed to identify genes involved in the CDC. Through correlative analysis they were able to reduce the number of CDC genes to 800. Cho et al. [15] carried out a similar reduction.

Spellman et al. [99] also investigated cell cycle regulatory response to *cln-3p* and *clb-2p* and found that more than half of the 800 correlative genes responded to perturbations in one or both of these proteins.

Each time series and the *cln-3p* and *clb-2p* experiments were normalised so that the $\log_2$ of the ratio for each gene to itself over all samples had mean 0. Aberrant data values (data values where the temporal neighbours were both at least 3 times as large or at least 3 times as small) were covered during pre-processing.

6.3.4 Rustici

The Rustici data set[35; 83] is the CDC of *Sch. pombe*. The CDC of this species is particularly interesting as a contrast to the more widely used *S. cerevisiae*. In comparison, *Sch. pombe* shows evidence of greater post-protein transcription regulation[83]. This means it is an important test case for the strength of expression data-only regulatory inference and the informational completeness of expression data. Because protein transcription occurs after mRNA creation, inference of *S. cerevisiae* and *Sch. pombe* is equally vulnerable to other hidden pre-protein

transcription regulatory effects that occur after the mRNA has been synthesised (and measured by microarray technology).

Sch. pombe is also known as "fission yeast". This is because it reproduces almost solely through the elongation and mitosis of a single *Sch. pombe* rod which divides the original cell into two equal daughter cells. As noted in subsection 6.3.2, *S. cerevisiae* reproduces primarily through a process known as *sporulation*. This difference is another reason *Sch. pombe* is very interesting.

There are 5 time series in the Rustici data set. Samples were taken every 15 minutes over a period of several hours for each time series, and the samples were time-indexed by p_{CDC}. This was calculated in 3 cases using elutriation and in the other 2 using a temperature-sensitive *cdc-25* mutant to arrest the cell cycle. The mutants allow for more accurate identification of p_{CDC}, however the changing temperatures and lengthened cell cycle may have introduced artifacts into the data[83].

Expression levels were preprocessed so that each gene had $\sigma^2 = 1$ across all experiments and mean $\log_2 x = 0$. Each elutriation time series contains two cell cycles, as does one of the *cdc-25* time series. The other *cdc-25* time series has only one.

Table 6.1: The Rustici et al. [83] data set. It is unclear which *cdc-25* data set has only cell division cycle in it.

Time Series	Number of Genes	p_{CDC} Range	Number of Samples
cdc-25 (1)	4540	71–261.14	19
cdc-25 (2)	4702	63–249.13	18
Elutriation (1)	4433	78–258.38	20
Elutriation (2)	4605	69–254.06	20
Elutriation (3)	4476	57–254.92	20

Cao [11] further post-processed this data set by restricting their inference to just the 407 known cell cycle regulators and by combining the data sets to produce a series of 115 samples. In addition, genes with too many missing samples were not considered. Genes without sufficient variation in the normalised range of expression and genes with unusually large spikes that may have been a consequence of noise were also eliminated.

6.3.5 Summary

The Eisen and Spellman data sets are becoming benchmark standards which new machine learning and bioinformatic techniques can be evaluated against. Further-

more, the underlying species (*S. cerevisiae*) has the advantages described in subsection 6.3.1. For this reason these data sets represent a very good initial application of the proposed machine learning research.

The Rustici data set is also particularly valuable as it is from a related but different species. This means that performance on this data set will provide further information about the general strength of the proposed machine learning research.

Most of the time series are also of the standard cell division cycle. Hence, multiple time series can be combined and we can be confident that the regulatory functions are singular, rather than a complex logical combination of several different functions.

Chapter 7

Discussion

This chapter summarises the transfer report, presents a work plan and discusses contingency plans. Section 7.1 summarises the expected research contributions. Section 7.2 gives a month-by-month work plan and section 7.3 briefly outlines contingencies.

7.1 Contributions and Objectives

The key goal and contribution of this research is to tractably infer detailed drafts of large causal networks in the face of noisy and partially covered training data. This is an important and general problem in machine learning, with particular immediate applicability to bioinformatic problems. Currently, and despite a substantial review of the literature, there appear to be no techniques which infer a detailed (component-by-component) draft model of large networks.

Our formalisation of FBN combines the complementary strengths of fuzzy techniques and Bayesian networks and is the first major theoretical contribution. It illustrates how the strengths of these two complementary techniques can be tractably combined. Our theoretical analysis of fuzzy probability should provide a solid foundation for this formalisation, and the analysis may be applicable to other research areas as well.

The development of an integrated sequence (a pipeline) for unsupervised causal network inference represents the second major theoretical contribution. We aim to show how clustering and FBN can be used together to address new and more difficult problems.

We will illustrate the practical utility of these theoretical contributions by inferring large GRN. Even the most modern research (e.g. Bonneau et al.'s [9]) uses supervised clustering, requires substantial prior knowledge and does not employ dimension reduction techniques which allow detailed model inference. Thus, we see

practical solutions to this kind of problem as a major open question in machine learning.

7.2 Work Plan

Table 7.1 provides a tentative month-by-month breakdown of work to be done in the proposed course of research. This plan includes a two month "safety buffer".

The proposed course of research described in this transfer report could be extended in several ways. Further use of standard techniques on sub-graphs of the de-fuzzified FBN are necessary to maximise the value of the draft network. The principled inclusion of prior knowledge for any particular causal inference problem is also a valuable open area for future research. Analysis of FBN inference *sample complexity*[1; 31; 44; 63; 85] would be a valuable theoretical contribution.

In addition, the theoretical techniques have obvious applications to other kinds of biological networks (potentially including hyper-graphical chemical reaction networks) and more general areas of machine learning and causal inference. The development and evaluation of fuzzy HMM for natural language processing is one such area.

Further exploration of these areas after the completion of the DPhil would be essential to maximise the value of the contributions outlined in this report.

7.3 Contingency Plans

Although it is difficult to identify potential problems in advance, some elements of this plan are more risky than others.

If time was running short then focus could be maintained on the theoretical contributions, rather than on the application of the research to biological data sets.

Similarly, if the best available clustering algorithm is insufficient, then more focus would be placed on the modification and development of a novel clustering algorithm. This is unlikely to be necessary as several clustering algorithms have already been identified as likely candidates[41; 42; 47; 89].

Should the work take much less time than expected then the sample complexity of FBN would be valuable to analyse.

Finally, note that the expected strength of FBNs is that they synergistically make large network inference tractable, therefore they may not perform as well as a discrete or continuous model on all small networks.

Table 7.1: Month-by-month work plan. Paper submission and writing is in *italics*.

Timespan	**Proposed Work**
September	Update and *submit* appendix G; holiday
October	Implement FBN belief propagation and FBN inference in MatLab; *submit* expanded [30] to JHIS (invited to submit).
November	Finish implementing FBN inference, *submit* theoretical fuzzy probability research to UAI
December	Evaluate FBN on small simulated networks; holiday
January 2009	⋮
February	*Write paper* on FBN inference; holiday
March	Select and implement candidate clustering algorithms
April	*Write paper* on clustering algorithm; holiday
May	Evaluate large FBN on simulated data
June	⋮
July	*Write paper* on FBN inference of simulated data
August September	Apply FBN inference to biological data (small GRN); holiday
October	Apply FBN inference to biological data (large GRN)
November	⋮
December	Evaluate FBN inference; holiday
January 2010	⋮
February	*Write paper* on FBN inference of biological data
March	Write final thesis; holiday
April	⋮
May	⋮
June	⋮
July	There will probably be a month lost revising papers based on referee's feedback over the course of the research.
August	Buffer for overruns
September	⋮

Appendix A

Summary of Work To Date

This appendix gives a month by month breakdown of work and research done so far.

- October 2006
  - Bioinformatics course with Jotun Hein, Geoff Nicholls
  - Intelligent Systems I course with Stephen Cameron
  - Weekly Systems Biology seminars with Tom Melham and others
  - Initial background reading on GRN
- November 2006
  - Bioinformatics course with Jotun Hein, Geoff Nicholls
  - Intelligent Systems I course with Stephen Cameron
  - Weekly Systems Biology seminars with Tom Melham and others
  - Initial background reading on GRN
- December 2006 to November 2007

  Health problems (RSI) meant that I suspended my studies and returned to New Zealand during this period. Because of these problems my ability to do any work was very limited.
  - Read about GRN inference, molecular biology
- December 2007
  - Read about and considered the application of *computational learning theory* (COLT) to fuzzy systems and GRN inference

  - First draft of Ockham's Razor/Bayesian statistics paper (appendix G).

- January 2008
  - Worked on Ockham's Razor/Bayesian statistics paper
  - Continued to investigate COLT for BN and noisy data

- February 2008
  - Reading and discussion group with Jotun Hein, Aziz Mithani
  - Read and summarised fuzzy clustering, past research into combination of fuzzy theory and Bayesian networks
  - Machine Learning course with Vasile Palade

- March 2008
  - Selected biological data sets
  - Established working relationship with Joanna Davies, Eleni Giannolatou
  - Drafted formalisation of FBN
  - Machine Learning course with Vasile Palade
  - Considered clustering to preprocess for FBN in more detail

- April 2008
  - Published CS-RR-08-04 ("Machine Learning and GRN literature review")
  - Learnt MatLab
  - Wrote `GreenSim`, a GRN data simulator
  - Began `GreenSim` technical report
  - Assisted Davies, Hein with Gaussian Processes article
  - Attended Conceptual Foundations of System Biology seminar series
  - Started working on FBN paper
  - Contacted Paul Cao (Oberlin), Jing Yu (Duke), Alexander Hartemink (Duke) on matters relating to my research

- May 2008
  - Completed and published FBN paper (CIMA at ECAI)

  - Completed and published PRG-RR-08-07 ("GreenSim: A Genetic Regulatory Network Simulator")
  - Distributed `GreenSim` to Hartemink, Yu and received feedback.
  - Reading groups (sporadic) with Davies, Hein, Miklos
  - Seminar series: Conceptual Foundations of Systems Biology (CFSB) with Eric Werner and others
  - Demonstrator, Imperative Programming II

- June 2008
  - CFSB seminar series, focused on antireduction
  - Rewrote and submitted CS-RR-08-04 as a chapter in "Foundations of Computational Intelligence"
  - Demonstrator, Imperative Programming II

- July 2008
  - Presented FBN in CIMA at ECAI; attended ECAI
  - Organised "Learning" discussion group with Andras Salamon

- August 2008
  - Began working on "Fuzziness, Probability and Fuzzy Probability"
  - Submitted "Belief Propagation in Bayesian Networks: A Worked Example" to the DPhil student conference
  - Installed BNT in MatLab, began considering how to extend it for FBN

- September 2008
  - Completed transfer report
  - Began working on "Fuzziness, Probability, and Fuzzy Probability: A Conceptual Analysis" (appendix D).

Appendix B

Belief Propagation in Fuzzy Bayesian Networks

The paper included below was accepted after peer review at the Combinations of Intelligent Methods and Approaches Workshop (CIMA), at ECAI 2008 in Patras, Greece. The conference took place between the 21st and 25th of July, 2008.

Belief Propagation in Fuzzy Bayesian Networks

Christopher Fogelberg[1] and **Vasile Palade** and **Phil Assheton**[2]

Abstract. Fuzzy Bayesian networks are a generalisation of classic Bayesian networks to networks with fuzzy variable state. This paper describes our formalisation and outlines how belief propagation can be conducted. Fuzzy techniques can lead to more robust inference. A key advantage of our formalisation is that it can take advantage of all existing network inference and Bayesian network algorithms. Another key advantage is that we have developed several techniques to control the algorithmic complexity. When these techniques can be applied it means that fuzzy Bayesian networks are only a small linear factor less efficient than classic Bayesian networks. With appropriate pre-processing they may be substantially more efficient.

1 Introduction

Modern machine learning research frequently uses *Bayesian networks* (BNs)[6; 7; 15; 16]. However, BN inference is NP-complete due to cycles in the undirected graph[4], and belief propagation is exponential in the *tree-width* of the network. This makes them difficult to use for large problems.

Fuzzy[3] and hybrid fuzzy systems[11; 13] are also frequently used. In a fuzzy system, a variable's *state* is represented by a set of *fuzzy values* (FVs). Because fuzzy systems do not force a model to artificially discretise a continuous underlying state they are often more robust in the face of noise.

To date there has been very little research into BNs with fuzzy variable states. What there is has centred around the use of fuzzy approximations to perform inference and belief propagation in a *hybrid BN*[1; 12]. A hybrid BN is one where the parameters are a mix of continuous and multinomial distributions.

This paper's key contribution is a formal generalisation of classic BNs to *fuzzy Bayesian networks* (FBNs). In a FBN the variables can have *fuzzy states*. The paper also describes tractable belief propagation over FBNs. An important advantage of the presented formalisation is that existing inference algorithms (e.g. MCMC, simulated annealing) can be used without modification. Furthermore, FBNs may be only a small linear constant less efficient than classic BNs of the same size, and appropriate pre-processing may make some problems which were intractable for classic BNs tractable for FBNs.

The paper is structured as follows. Section 2 presents a fuzzy Bayesian network which will be used as an example. Section 3 introduces some notation (subsection 3.1), and presents belief propagation for variables with one parent (subsection 3.3) and for variables with multiple parents (subsection 3.4). Section 4 analyses the algorithmic efficiency of FBNs and how it can be controlled. Section 5 outlines an important bioinformatic domain where FBNs may be especially useful, and section 6 concludes the paper.

[1] Supported by the ACU and CSCUK

[2] Computing Laboratory, University of Oxford; Contact email: christopher.fogelberg@comlab.ox.ac.uk

2 A Fuzzy Bayesian Network

The structure of the FBN that is used as an example in this paper is shown in figure 1. Call this FBN $G = \langle \eta, \theta \rangle$, where η denotes the structure of G and θ its parameters.

For clarity of presentation, G is a multinomial (discrete) BN. However, the formalisation generalises easily and transparently to continuously-valued FBNs and hybrid FBNs.

The relevant conditional distributions of G are shown in figure 2. D's distribution is not shown and we will later assume a state for D with no loss of generality.

Because we have restricted the differences between BNs and FBNs to belief propagation, the specification of a FBN and a BN are identical.

3 Belief Propagation

Belief propagation in a Bayesian network involves calculating the updated probability distributions of variables in the network, given θ and the observed states of other variables.

3.1 Some Notation

The terminology and notation is as follows. A variable has a state, either a fuzzy state (FS) or a *discrete state* (DS).

A fuzzy state is made up of one or more *components*, and each component is annotated with the variable's degree of membership (μ) in that component. For example, equation 1 is an example of a variable (S) with two components. It has membership 0.7 in the component *hi* and membership 0.3 in the component *mid*. *hi* and *mid* are examples of *values* that the variable can take. When annotated with μ they are referred to as fuzzy values (FV). The set of all possible values (fuzzy values) that a variable can take is the *range* of that variable, e.g. *hi*, *mid*, *lo*.

$$S = [hi_{0.7}, mid_{0.3}] \tag{1}$$

In general, the components of a variable's state are enclosed in square brackets. A discrete state is just a special case of a fuzzy state. Discrete states have just one component with $\mu = 1$, and the square brackets and μ subscript can be omitted in this situation. We assume that $\sum_{c \in C} \mu_c = 1$ for a FS with C components and have not considered other situations.

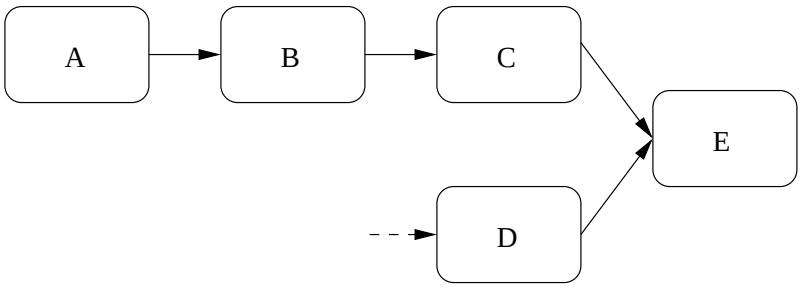


Figure 1. The fuzzy Bayesian network G, used as an example in this paper.

$\rightarrow A$	$A = lo$	$A = mid$	$A = hi$
	0.7	0.1	0.2

(a) θ_A, A's prior distribution

$A \rightarrow B$	$B = lo$	$B = mid$	$B = hi$
$A = lo$	0.6	0.2	0.2
$A = mid$	0.1	0.1	0.8
$A = hi$	0.1	0.2	0.7

(b) θ_B, B's conditional distribution

$B \rightarrow C$	$C = lo$	$C = mid$	$C = hi$
$B = lo$	0.1	0.1	0.8
$B = mid$	0.1	0.8	0.1
$B = hi$	0.7	0.2	0.1

(c) θ_C, C's conditional distribution

$C, D \rightarrow E$		$E = lo$	$E = mid$	$E = hi$
$C = lo$	$D = lo$	0.6	0.2	0.2
$C = lo$	$D = mid$	0.1	0.1	0.8
$C = lo$	$D = hi$	0.1	0.1	0.8
$C = mid$	$D = lo$	0.6	0.2	0.2
$C = mid$	$D = mid$	0.1	0.6	0.3
$C = mid$	$D = hi$	0.1	0.6	0.3
$C = hi$	$D = lo$	0.1	0.2	0.7
$C = hi$	$D = mid$	0.1	0.2	0.7
$C = hi$	$D = hi$	0.8	0.1	0.1

(d) θ_E, E's conditional distribution

Figure 2. θ for G. The conditional distributions of A, B, C and E.

Just as a component can be a value from the range of a variable, e.g. hi, a component can also be a probability distribution (PD). PD are denoted with curly brackets, e.g. $\{hi_{0.3}, mid_{0.2}, lo_{0.5}\}$. Because the value associated with each subscripted probability is implicit in the tuple order the value names can be omitted: $\{0.3, 0.2, 0.5\}$.

An example of a fuzzy state which mixes values and probability distributions is shown in equation 2.

$$T = [\{0.2, 0.1, 0.7\}_{0.2}, \{0.1, 0.8, 0.1\}_{0.6}, mid_{0.2}] \qquad (2)$$

A PD which is annotated (subscripted) with μ is called a *fuzzy probability distribution* (FPD). Samples are drawn from a FPD in the same way that they are drawn from a PD. However, a variable with membership μ in a FPD can only have μ proportion of its state determined by that FPD; a sample from a FPD will have the same μ as the FPD does. For example, a sample from $\{0.2, 0.1, 0.7\}_{0.2}$ will be one of $lo_{0.2}$, $mid_{0.2}$ or $hi_{0.2}$, and each of these components will be drawn with probability 0.2, 0.1 and 0.7 respectively.

This means that the state of a sample from the uncertain variable T (equation 2 will be some member from the set $[lo_{[0..0.8]}, mid_{[0.2..1]}, hi_{[0..0.8]}]$ and the distribution over members in this set is determined by the two FPD and one FV which make up the fuzzy state of T.

3.2 Assumptions

The full and general analysis of FBNs would also consider unrestricted interactions amongst components in a fuzzy state, allowing $\sum \mu \neq 1$ and so forth. In this article we make a number of linearising assumptions which make FBN belief propagation cheap, relative to the cost of full general propagation. They also greatly aid the clarity of the presentation in the space available. Furthermore, these assumptions are reasonable and do not restrict the general utility of FBNs. However it is important to make them explicit. The assumptions (and consequently the nature of full general propagation) will be briefly summarised in this section; a more general discussion is forthcoming.

3.2.1 Assumption: Total Membership

As noted above and in subsection 3.1, we assume that $\sum \mu = 1$. This is our first linearising assumption, and it can be conceptualised as follows. A variable's degree of membership in each of its $|C|$ components forms a $|C|$-dimensional *fuzzy state space*. If the variable has no uncertainty (no component is an FPD) then $|C|$ will be the same as the variable's range. Even if a FS has 0 membership in some of its range those values are still part of the state's FSS. By assuming that $\sum_{c \in C} \mu_c = 1$, we restrict our attention to a smaller $|C| - 1$ dimensional subspace. This subspace constrains the degrees of membership in each component in a cyclically conditional way on the degrees of membership in the other components. This assumption simplifies the combination of components of fuzzy states in subsection 3.4.

For example, if a FS has membership 0.5 in the value hi its membership in the values lo and mid in the FSS are constrained to be in the range $[0, 0.5]$. Furthermore, its membership in lo and mid mutually constrain (in this case define, as

there are no other values in the range) each other. Figure 3 illustrates the impact of the first assumption on the FSS for a fuzzy state with two components.

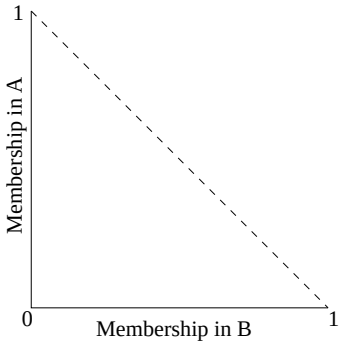


Figure 3. Imagine a fuzzy state with two components, A and B. Such a state would have a two-dimensional FSS, as in this figure. Without restriction, the state's degree of membership in each component could be specified by any point in the FSS. However, we assume that $\sum_{c \in C} \mu_c = 1$. Therefore its membership in components A and B must be specified by some point on the dashed line.

3.2.2 Assumption: Component Independence

We also make two further assumptions about FBN during belief propagation. The first is that components are independent. This means that when a variable has only one parent then its state will have one component for each component its parent has, and these components will have the same μ as the corresponding parent's component. For example, the children which have S (equation 1) as their only parent will have two components in their FS, one with $\mu = 0.7$ and one with $\mu = 0.3$.

When a variable has more than one parent then the components of the parents will be mixed and combined before propagation. This is described in subsection 3.4. Because we assume independence, the child's fuzzy state will have one component for each component in the mixed and combined parent set, and each of the child's component will have the same μ as the corresponding component in the parent set.

It will also be clear that we assume independence when we describe how the parents' components are mixed and combined in subsection 3.4.

3.2.3 Assumption: FPD Samples

The third assumption implicit in this model is the assumption that a sample from an FPD with $\mu = x$ will be a single fuzzy component with $\mu = x$. Although natural and intuitive we do not believe that this is automatically entailed, thus we explicitly assume it.

Consider an uncertain variable with a range of r in a standard (discrete) Bayesian network. Its state will be a probability distribution which specified a single point in the r dimensional *probability space* (p-space).

Now consider this variable in a FBN. Any uncertainty in its state will be represented by an FPD component with some μ. As described in subsection 3.1, a FPD is just PD with a fixed (0 dimensional) μ associated with it.

This definition of a FPD could be generalised so that μ could vary independently but was fixed for each of the r possible samples that could be drawn from the FPD. Call this a *slightly general FPD* (SGFPD) and call the FPD defined in subsection 3.1 a *standard FPD*. An SGFPD would specify a single point in an $r + r$ dimensional space, where r of the dimensions are the probability of each value and the other r dimensions specify the μ of a sample of each value. Just as the first r dimensions specify a p-space, the second r dimensions specify a *μ-space*. An example of such a space is given diagrammatically in figure 4, and it is used to contrast a SGFPD with a standard FPD.

An example of an SGFPD might be "there is a 0.2 probability of drawing a sample of *hi*, and any sample of *hi* will have $\mu = 0.3$, and there is a 0.3 probability of drawing a sample of *mid*, and any sample of *mid* will have $\mu = 0.5$, and..." and so forth. The assumption that $\sum_{c \in C} \mu_c = 1$ for a state could be relaxed if SGFPD were used.

After considering figure 4 it will be clear that SGFPD could be further generalised so that the μ of any sample also varied probabilistically, conditional on the value (*hi*, *mid*, etc.) of the sample. Such a *general FPD* (GFPD) would be an $r + r$ dimensional probability distribution over the joint μ and range of the variable. We believe that this represents the most general kind of inference and belief propagation in a FBN. Such inference is intractable and we do not consider it in this paper.

In summary, the assumptions which we have made substantially reduce the dimensionality of belief propagation and are necessary for it to be tractable. However, more general FBNs with GFPD do not have these restrictions; their utility will be considered in a forthcoming publication.

3.3 Single-Parent Belief Propagation

Assume that observations indicate $A = [mid_{0.2}, hi_{0.8}]$ in G. With this information we can calculate the updated distributions on B and C.

Because A has an observed (certain) FS and is B's only parent the components of B's updated FS can be read from θ_B. This shows that:

$$B = [\{0.1, 0.1, 0.8\}_{0.2}, \{0.1, 0.2, 0.7\}_{0.8}] \tag{3}$$

The FS over C is calculated similarly. Just as each of the fuzzy values in A lead to a weighted FPD in the FS of B the same occurs for C, and $C = [\alpha_{0.2}, \beta_{0.8}]$. The weighted distributions α and β are calculated using standard BN belief propagation, based on the conditional distribution of B. This is shown in equations 4, 5 and 6.

$$\begin{aligned} p(C|B = lo) &= \{0.1, 0.1, 0.8\} \\ p(C|B = mid) &= \{0.1, 0.8, 0.1\} \\ p(C|B = hi) &= \{0.7, 0.2, 0.1\} \end{aligned} \tag{4}$$

$$\begin{aligned} \alpha &= \{0.1, 0.1, 0.8\} \times 0.1 + \{0.1, 0.8, 0.1\} \times 0.1 + \\ &\qquad \{0.7, 0.2, 0.1\} \times 0.8 \\ &= \{0.01, 0.01, 0.08\} + \{0.01, 0.08, 0.01\} + \\ &\qquad \{0.56, 0.16, 0.08\} \\ &= \{0.58, 0.25, 0.17\} \end{aligned} \tag{5}$$

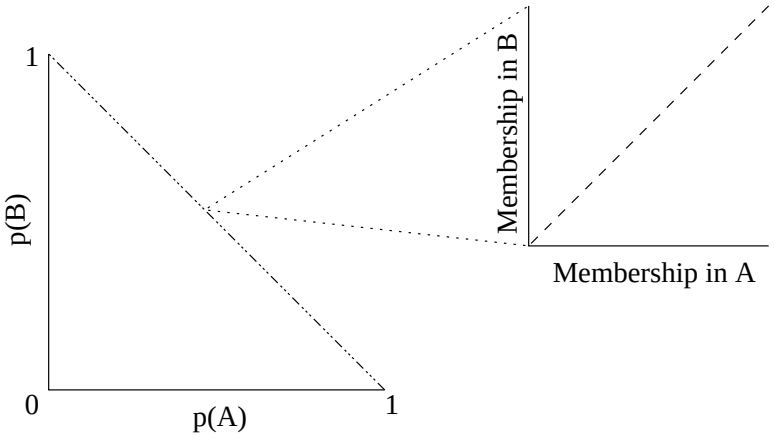


Figure 4. Assume a variable with a range (r) of 2 (A and B). Each point in the 2 dimensional p-space (left hand side) could be considered an index into a 2 dimensional μ-space (right hand side), as diagrammed. All proper probability distributions ($\sum p = 1$) fall on the dotted dashed line in the p-space. The μ-space that is indexed by some point in the p-space could be unique to that point. In a standard FPD, the μ-space is reduced to a single point on the dashed line and that point on the line in the μ-space is specified precisely by the μ of the FPD. Because any sample from a standard FPD, regardless of its value, will have the same μ, r of the dimensions are eliminated. In addition, we assume that an FPD has a proper probability distribution. In total, these reduces the dimensionality from $r + r$ to $r - 1$. In a SGFPD the μ-space would also be reduced to a single point, but any point in the μ-space would be a valid reduction, so an SGFPD with a proper distribution over the values still has $r + r - 1$ dimensions.

$$\begin{aligned}\beta &= \{0.1, 0.1, 0.8\} \times 0.1 + \{0.1, 0.8, 0.1\} \times 0.2 + \\ &\qquad \{0.7, 0.2, 0.1\} \times 0.7 \\ &= \{0.01, 0.01, 0.08\} + \{0.02, 0.16, 0.02\} + \\ &= \{0.49, 0.14, 0.07\} \\ &= \{0.52, 0.31, 0.17\} \end{aligned} \tag{6}$$

The calculated FS for C is shown in equation 7.

$$C = [\{0.58, 0.25, 0.17\}_{0.2}, \{0.52, 0.31, 0.17\}_{0.8}] \tag{7}$$

3.4 Multi-Parent Belief Propagation

Subsection 3.3 illustrated belief propagation in a FBN when a variable has only one parent. This subsection shows *naive FBN belief propagation* in the case of a variable with multiple parents. Section 4 outlines several more nuanced approaches which address the problems with naive propagation.

Take the calculated value of C, and assume a fuzzy state for D (equation 8). What is the updated fuzzy state of E?

$$\begin{aligned} C &= [\{0.58, 0.25, 0.17\}_{0.2}, \{0.52, 0.31, 0.17\}_{0.8}] \\ D &= [\{0.45, 0.30, 0.25\}_{0.3}, \{0.1, 0.8, 0.1\}_{0.7}] \end{aligned} \tag{8}$$

Any *combination* of components, one from each parent, can be used to calculate an updated probability distribution for a variable. However, this raises the question of how to combine and weight each combination of component distributions in the parent FSs to calculate an updated FS for the child.

Because the parents are conditionally independent given the variable being updated[3], any particular combination of PD and observations can be summed over, as one was in each of equations 5 and 6. The summed over combinations became components of C's updated distribution.

[3] And also given their updated state and the acyclic nature of the graph

In the naive approach to belief propagation the Cartesian product of the parents' FS is used to find all possible combinations of components. μ for each one of these combinations is calculated using the *product t-norm*[2]. Any other fuzzy conjunction (normalising μ where necessary) could also be used. Because we assumed that $\sum \mu = 1$ holds for each of the parents though, using this fuzzy conjunction guarantees that $\sum \mu$ over the child's components will also equal 1 and no normalisation is necessary.

For example, if we use the first components of C and D ($\{0.58, 0.25, 0.17\}_{0.2}$ and $\{0.45, 0.30, 0.25\}_{0.3}$, respectively, equation 8) then standard Bayesian propagation and using the product t-norm to calculate μ shows that one member of E's updated FS is:

$$\alpha = \{0.3165, 0.2189, 0.4647\}_{0.06} \tag{9}$$

The full FPD for E will have four members, one for each member of $C \times D$ (equation 10, below). For clarity, the calculated α from equation 9 has not been substituted into this equation.

$$E = [\alpha_{0.06},\ \beta_{0.14},\ \gamma_{0.24},\ \delta_{0.56}] \tag{10}$$

In general a variable with k parents that each have an FS with m components will have an updated FS of size m^k. Assuming all variables have k parents, the grand-children will have updated FS of size m^{k^k}, and so forth. This is the *fuzzy state size explosion* (FSSE), and it makes naive belief propagation in a FBN intractable.

4 Dealing with Complexity

There are several ways that the explosion in the complexity can be controlled by approximating the FS. This section discusses four kinds of control. The bimodal fuzzy state $X = [\{0.9_\alpha, 0.1_\beta\}_{0.5}, \{0.1_\alpha, 0.9_\beta\}_{0.5}]$ is used as an example in several places in this section. Such a variable could represent

a committee of two in which the committee members (components) hold diametrically opposite beliefs about the outcome of some future event.

4.1 Linear Collapse

A first approximation that addresses the FSSE is to linearly collapse a FS that is made up only of FPDs, immediately after they are calculated. Each component can be weighted by its fuzzy membership and they can be summed to calculate a single, discrete, PD. For example, B (equation 3) can be collapsed as shown in equation 11. Collapsed FS are denoted with a prime.

$$\begin{aligned} B &= [\{0.1, 0.1, 0.8\}_{0.2}, \{0.1, 0.2, 0.7\}_{0.8}] \\ \therefore B' &= \{0.1, 0.18, 0.72\} \end{aligned} \tag{11}$$

However, this approximation is unsatisfactory: it conflates probability with fuzziness and may change the *expected value* of the variable. Although it may be approximately correct in some circumstances, a simple thought experiment will show why it is insufficient.

Consider the bimodal FS X. The expected sample from X is $X' = [\alpha_{0.5}, \beta_{0.5}]$. Although this sample does not reflect any of the uncertainty in X it does reflect the bi-modality (indecision) of the variable (committee) as a whole. Subsection 4.4 returns to this approach.

If X is linearly collapsed though then $X' = \{0.5, 0.5\}$. No sample drawn from this PD can be half α and half β. Important information in X has been lost. Although further belief propagation will not be biased if this variable is summed over[4], there is no way to compare the linearly collapsed value X' with any observed value for X when trying to evaluate the quality of an inferred network.

Other approximations to the naive approach have been developed. They are discussed in the next three subsections.

4.2 Strict and Dynamic Top Fuzzy Combinations

Consider again the full (naive) FS of E, reproduced in equation 12.

$$E = [\alpha_{0.06},\ \beta_{0.14},\ \gamma_{0.24},\ \delta_{0.56}] \tag{12}$$

Some of the components barely contribute to the overall state and will not have a substantial influence on any children either. Such components could be ignored, and the remaining components could have their μ normalised. For example, if just the top three components of E were used then the updated FS would take the form:

$$E = [\beta_{0.149}, \gamma_{0.255}, \delta_{0.596}] \tag{13}$$

The number of components retained could be either *k-component strict selected* or *ϕ-dynamically selected*. In the former case, the top k components would be selected. In the latter, the $|C|$ components with greatest μ would be selected so that $\sum_{c \in C} \mu_c > \phi$. Strict selection would mean that FBNs were only a small linear factor less efficient than classic BNs of the same size. However, the top k components may not be an accurate reflection of the full FS, thus ϕ-dynamic selection may be more appropriate in some cases.

[4] Due to the use of the product t-norm.

4.3 Clustering the Fuzziness

Another way of controlling the FSSE is to calculate the full FS of each variable during belief propagation. However, before using the full FS to update the state of its children, its components could be clustered so that FPD which specified similar distributions were combined together.

For example, the FS $[\ldots, \{0.7, 0.2, 0.1\}_{0.3}, \{0.6, 0.3, 0.1\}_{0.2}, \ldots]$ might cluster to $[\ldots, \{0.66, 0.24, 0.1\}_{0.5}, \ldots]$.

Because the clustering problem would only have as many dimensions as the range of each FPD, we speculate that a simple fixed-k clustering algorithm like k-means would work very well.

Although this approach is more complex than selection or linear collapse, the total increase in complexity in belief propagation would be related to and bound by the maximum in-degree and range of a variable.

4.4 Expected Values

A fourth kind of control is inspired by particle filtering and the Condensation algorithm[9]. The general *sequential Monte Carlo* (SMC) method will be outlined first. Although this approach is not as efficient as others it is applicable in all cases and is strictly correct.

Consider X. An infinite sequence of independent samples drawn from this uncertain fuzzy state will take something like the form $[\alpha_{0.5}, \beta_{0.5}]$, $[\alpha_{0.5}, \beta_{0.5}]$, $\ldots$, $[\alpha_1]$, $[\alpha_{0.5}, \beta_{0.5}] \ldots$ and so forth.

The properties of this sequence are identical to those of the fuzzy state, and a long-enough finite sequence will be a good approximation to it. For example, 100 samples could be drawn from X. Each of these samples could then be used to propagate the uncertain state of X to X's children. The relative efficiency of this technique compared to clustering depends on the range and k_{max} of the variables, but in certain situations it may also be better.

As noted in subsection 4.1, the expected value of a variable is easily calculated analytically. For example, the expected value of X is $[\alpha_{0.5\times0.9+0.5\times0.1}, \beta_{0.5\times0.1+0.5\times0.9}] = [\alpha_{0.5}, \beta_{0.5}]$.

Doing this expectation calculation is analogous to summing over or numerically integrating a probability distribution, and we call it *fuzzy integration*. Like clustering the impact on efficiency of fuzzy integration depends on the range and k_{max}.

This approach is very similar to linear collapse but it has a number of key advantages. Firstly, like linear collapse, it does not bias any further belief propagation. This is untrue of selection and clustering. Secondly, the expected value X' which is the result of this fuzzy integration can be meaningfully compared with observed values of X when performing network inference. In many cases, users are only interested in the expected (integrated) value. In these cases the expected value of a FS is ideal.

5 A Bioinformatic Domain

Inference of large *genetic regulatory networks* (GRN) is a central problem in modern bioinformatics. However, algorithmic complexity has limited detailed inference using BNs[6;

15; 16] to $N \lessapprox 100$ genes[5]. Approaches which can be applied to larger numbers of genes include modern clustering methods[10] and the inference of *graphical Gaussian models* over clustered gene expression data[8; 14].

FBNs suggest a novel approach to detailed exploration of large GRN. Such a methodology generalises to the inference of other large causal networks as well.

If the data is pre-processed by using a *fuzzy cover* algorithm the dimensionality of the problem may be reduced by an order of magnitude or more. This could lead to an exponential reduction in the algorithmic complexity which would more than offset any increase caused by the fuzzy state size explosion and its collapse.

A fuzzy cover is a clustering algorithm which *covers* the data, rather than clusters it. In a fuzzy cover, a variable (gene) can have $\sum_{c \in C} \mu_c > 1$, where C is the set of covers that the algorithm finds.

Inference over the covers is performed using a standard algorithm to find a virtual GRN. Using the retained μ_c for each $n \in N$ and $c \in C$, most of the original fidelity can be recovered after the inference has been performed by linearly devolving and normalising the network of covers back down to a network of genes. The synergistic use of dimension-reduction and FBNs are what we believe will be most useful.

The authors are using this approach (fuzzy covering, FBN inference, FBN devolution) to infer and explore large *genetic regulatory networks*. With fuzzy clustering and FBNs we expect to be able to perform more detailed exploratory inference for $N \approx 1000$.

6 Contributions and Future Work

This paper has presented a new formalisation which combines fuzzy theory and Bayesian networks. Because of the way that it extends classic BNs, all existing algorithms, tools and machine learning techniques for classic BNs can be used immediately with FBNs.

Several techniques for tractably propagating fuzzy beliefs across a FBN are also described. Using these techniques, previously used BNs can be assigned fuzzy variable states and updated accordingly. This means that existing networks, often learnt only after substantial effort, can be easily reused.

Furthermore, the difference in BN and FBN efficiency with sensible fuzziness collapse may be as little as a small linear constant in some circumstances. This means that there are few disadvantages to using FBNs instead of BNs.

The possibility of integrating FBNs into a machine learning pipeline which involves dimension-reduction and network devolution also suggests that the inference of larger causal networks will be possible using FBNs.

Future research may uncover more efficient methods for integrating, clustering or otherwise collapsing a FS. In addition, the authors plan to present an even more generalised formalisation which relaxes the assumptions made in subsection 3.2.

REFERENCES

[1] 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, pp. 630–635, (May 2003).

[2] F. Bobillo and U. Straccia, 'A fuzzy description logic with product t-norm', in *Proceedings of the 16th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007)*, pp. 652–657, London (United Kingtom), (July 2007).

[3] 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).

[4] David M. Chickering, 'Learning Bayesian networks is NP-Complete', in *Learning from Data: Artificial Intelligence and Statistics V*, eds., D. Fisher and H. J. Lenz, 121–130, Springer-Verlag, (1996).

[5] Christopher Fogelberg and Vasile Palade, 'Machine learning and genetic regulatory networks: A review and a roadmap', Technical Report CS-RR-08-04, Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford, OX1-3QD, (April 2008).

[6] 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*, 437–449, (2002).

[7] David Heckerman, 'A tutorial on learning with Bayesian networks', Technical report, Microsoft Research, Redmond, Washington, (1995).

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

[9] Michael Isard and Andrew Blake. Condensation – conditional density propagation for visual tracking, 1998.

[10] 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).

[11] Daniel Neagu and Vasile Palade, 'A neuro-fuzzy approach for functional genomics data interpretation and analysis', *Neural Computing and Applications*, **12**(3-4), 153–159, (2003).

[12] 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).

[13] 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).

[14] 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).

[15] 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).

[16] 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 *BIOCOMP*, eds., Hamid R. Arabnia and Homayoun Valafar, pp. 41–47. CSREA Press, (2006).

Appendix C

Fuzzy Bayesian Networks — A Worked Example

This appendix contains the long version of the paper which has been submitted to the 2008 Computing Laboratory DPhil Student Conference for peer review. It includes a second discussion of FBN and contrasts belief propagation in classic BN and fuzzy BN, using this contrast to highlight the differences and strengths of FBN as a model representation.

The 2 page abstract submitted to the conference is available online at: `http://syntilect.com/cgf/pubs:fbn-example`.

Belief Propagation in Fuzzy Bayesian Networks: A Worked Example

Christopher Fogelberg

September 13, 2008

Abstract

Fuzzy Bayesian networks (FBN) are a model representation for machine learning techniques. They are graphical structures with variables that are simultaneously fuzzy and uncertain.

Although very similar to classic discrete or multinomial Bayesian networks (BN), and able to take advantage of existing BN techniques and algorithms, belief propagation in a FBN is different and can be too slow on some graphs. Propagating expected values effectively addresses this problem. Because the variables are simultaneously fuzzy and uncertain it can also be confusing. In this paper we summarise fuzzy Bayesian networks and provide examples of forward propagation, backward propagation and explaining away for the same classic BN and FBN.

By comparing, contrasting and interpreting classic BN and FBN side by side, belief propagation in FBN is made clearer, and the strengths of FBN over classic BN in certain situations are also highlighted.

1 Introduction

Fuzzy Bayesian networks (FBN) are a graphical machine learning model representation with variables which are simultaneously fuzzy and uncertain[3]. This paper describes the formalisation of FBN, introduces our notation, touches on the conceptual foundations and illustrates belief propagation in FBN.

By doing this it reifies FBN belief propagation and makes the formalism easier to understand. Further, it illustrates the advantages and expressivity of FBN in certain situations. We briefly extend this contrast to discuss

the interpretation of fuzzy variables, and what it means for a variable to be simultaneously fuzzy and uncertain. Readers are referred to the original paper[3] and current (in progress) research[2] for a more detailed discussion of these questions.

1.1 Motivation

Bayesian networks[5; 7; 12; 14] (BN) are commonly used in machine learning. This is due to their statistical rationality, capacity for rigourous causal inference, and robustness in the face of noisy, partially missing and realistic data. They are also more easily human-interpretable than other machine learning representations such as neural networks, and experts can specify prior knowledge in a principled manner to guide the machine learning search. A wide range of search algorithms have been developed for structural and parameter inference, including structural EM[4] and MCMC. Classically, Bayesian networks use continuous (Gaussian) or multinomial variables.

Similarly, a fuzzy model has a wide range of advantages. Fuzzy models are also robust in the face of noise-corrupted data. The use of linguistic terms aids human comprehension of the learnt model, and they are particularly useful when the data is insufficient to formulate a precise model. The need to specify membership functions also forces the designer to consider the semantic interpretation of the model parameterisation and construction more explicitly.

For these reasons, FBN (which combine these advantages) may be useful. Theoretical analysis in current research[2] also indicates that fuzzy variables can be more expressive than multinomial or continuous variables. Further, FBN may be used as part of an integrated sequence of machine learning techniques that include reversible dimensionality reduction techniques such as fuzzy cover clustering algorithms. This may allow larger problems to be addressed with FBN than with classic BN.

1.2 Structure

Section 2 summarises fuzzy Bayesian networks. Section 3 presents a Bayesian network (its structure and parameters) for classic and fuzzy BN. Forward propagation on this network is shown in section 4, back propagation in section 5, and explaining away in section 6.

Section 7 summarises the differences in belief propagation over a classic BN and over a fuzzy BN, and discusses how the results can be interpreted. This highlights the strengths of FBN as a model representation.

Variables in this paper are denoted with capital letters. A variable has a *state*, which for a Boolean variable is either the *value* true or the value false. All variables are Boolean and the discrete state of a variable is denoted with a lower case letter, e.g. $A = true$ and a are equivalent, as are $A = false$ and $\neg a$. By extension, $p(a)$ denotes the probability that $A = true$, and $p(A)$ is the probability distribution over A.

2 What is a Fuzzy Bayesian Network?

A Bayesian network is a graphical model of the probabilistic dependencies among a set of variables. A FBN is a Bayesian network which has fuzzy variables. Classically, a variable's states may be either discrete (multinomial) or continuous. This section introduces the essential elements of our work, and readers are referred to the original paper[3] for a more complete presentation.

It is important to note that fuzziness and probability are distinct. Previous research which has combined fuzziness and BNs has arbitrarily mixed them, using fuzziness as an approximation so that intractable problems can be tractably approximated[8; 10]. Although other mathematical research has also considered the rigourous unification of fuzziness and probability[13], ours appears to be unique in that it explicitly maintains the distinction while considering them simultaneously.

Our formalism consists of two key elements. First, there is the conceptual unification (subsection 2.1) and notation for multinomial probability distributions and fuzzy variables (subsection 2.2). Secondly, there are the fuzzy uncertainty semantics (subsection 2.3) and how this and the conceptualisation affects belief propagation (subsection 2.4).

2.1 A Conceptual Overview of FBN

The two most important ideas in FBN are simple:

- A variable in the network may be simultaneously fuzzy and uncertain.
- Uncertainty and fuzziness are not conflated.

Importantly, a FBN may have the same structure and parameters as a classic BN. In fact, a classic BN that has been learnt can have fuzzy beliefs propagated over its variables without modifying the structure or conditional distributions[1].

[1]Depending on the nature of the relationship between the underlying variable and the fuzzified state though, the previously learnt conditional distributions may not be optimal.

A fuzzy BN differs from a classic BN only in the type of its variables, and in how belief propagation is conducted. This means that algorithms such as *structural EM*[4], *MCMC*[6] and the junction-tree algorithm[9] for belief propagation in graphs which aren't polytrees can be used, and they only need to be slightly modified to take FBN belief propagation into account. We do not consider network inference in this paper. Belief propagation is summarised in subsection 2.4, and sections 4–6 show some examples. Subsection 2.2 introduces the notation we use to express the formalisation and describe belief propagation.

2.2 Some Notation

To facilitate clear presentation, we need to be able to denote the state of a fuzzy, observed variable differently than the state of a discrete, uncertain variable or the state of a fuzzy, uncertain variable.

2.2.1 Fuzzy States

A fuzzy state is denoted with square brackets surrounding its *components*. Each component is one of the *fuzzy values*[2], that the variable can take on and is subscripted by the variable's membership in that fuzzy value. For example, $C = [f_{0.3}, t_{0.7}]$ says that C has 0.3 membership in false and 0.7 membership in true. This might denote that it was cloudy for 0.7 (70%) of the day, and not cloudy for 30% of the day.

If we use lower case letters to denote the discrete states true and false then $c = [t_1]$ and $\neg c = [f_1]$. The types of relationships shown in equation 1 also hold amongst fuzzy states.

$$[f_{0.3}, t_{0.6}, f_{0.1}] = [f_{0.4}, t_{0.6}] = [t_{0.6}, f_{0.4}] \tag{1}$$

2.2.2 Probability Distributions

We consider BN with multinomial variables, and denote probability distributions (PD) with curly brackets. Each element of the *range* of a PD (usually the same range as the variable) has its probability subscripted. For example, a sample from the probability distribution $\{lo_{0.4}, mid_{0.5}, hi_{0.1}\}$ is lo 40% of the time, and mid or hi 50% and 10% of the time, respectively.

[2]Classically referred to as *fuzzy sets*, and analogous to the *values* that a multinomial variable may take on, e.g. hi, mid, lo, or $true$ and $false$.

2.3 Fuzzy Probability Distributions and Mixed States

We also need to be able to denote a *fuzzy probability distribution* and sample from it, and a variable which is fuzzy, partially uncertain and partially *observed*. The notation given in subsection 2.2 is extended to denote this.

2.3.1 Fuzzy Probability Distributions and Sampling

A fuzzy probability distribution is an uncertain component of a fuzzy state. Just as we annotate an observed (certain, definite) component of a fuzzy value with the state's μ in that value (e.g. $f_{0.3}$) we annotate a component probability distribution with a μ value, and this represents the state's certain membership that probability distribution, e.g. $[\ldots, \alpha = \{f_{0.4}, t_{0.6}\}_{0.5}, \ldots]$

As discussed in our current research[2], there are several ways of sampling from a probability distribution over a fuzzy variable. With FBN, we have adopted the convention that a sample from a FPD is only and entirely one value, and each value is drawn with its associated probability. For example $f_{0.5}$ will be drawn as a sample from the FPD α 40% of the time, and $t_{0.5}$ will be drawn 60% of the time. These are the only samples which could possibly be drawn from α.

Note that this approach relaxes the tight interdependencies amongst the fuzzy values. Consider a variable with membership in the fuzzy values *hi*, *mid* and *lo*. Usually its membership in *mid* would also define its membership in *hi* and *lo*, and vice versa. However this is not necessarily true of a fuzzily uncertain variable. We do not consider the theoretical implications further in this paper, but refer interested readers to our unpublished note[2] which indicates that it increases the expressive power of a variable in certain situations.

2.3.2 Mixed States

Mixed states combine the notation for FPD and fuzzy values and represent variables which are partially observed and partially uncertain. For example, a sample from the partially observed state $C = [f_{0.3}.t_{0.6}, \{f_{0.2}, t_{0.8}\}_{0.1}]$ will be $[f_{0.4}, t_{0.6}]$ with probability 0.1, and $[f_{0.3}, t_{0.7}]$ with probability 0.8. If C denotes a cloudy sky this might denote having observed the sky for 70% of the day, or an incomplete and uncertain weather forecast. We further discuss the interpretation of fuzzy states in section 7.1.

2.3.3 Implicit Assumptions in Fuzzy Uncertainty

To summarise, note that we have made three assumptions for FBN, and two are relevant to FPD. Firstly, a variable's membership in its set of fuzzy values always sums to 1, and the sampling strategy we have adopted maintains this. Secondly, FPD samples are *monolithic*; a sample from a FPD will only be of one value.

2.4 Belief Propagation in FBN

Belief propagation in FBN is very similar to belief propagation in classic BN, and the examples provided in sections 4–6 will be clearer than a formal explanation. However, this section highlights the third key assumption we have made to make FBN inference tractable, and explores the consequences of that assumption for belief propagation to variables with multiple parents.

The third assumption is that components within a variable don't interact during belief propagation. Thus if $C = [f_{0.3}, t_{0.7}]$ then the first component ($f_{0.3}$) is propagated through the conditional distributions of C's children exactly as in classic BN belief propagation without regard to the remainder of C's components, and C's child will have μ 0.3 in the result of propagating $f_{0.3}$.

The second component is propagated in a similarly independent way, which means that all variables which only have C as a parent will have two components in their updated state: one with membership 0.3 and propagated according to the C-value f, and one with membership 0.7 and propagated according to the C-value $true$.

If a variable has more than one parent, and if its parents each have more than one component, then we need to combine the parent's components. The selected way of doing this is with the Cartesian product of the sets of parent components, and μ for each member of the Cartesian product can be calculated using the product t-norm[1].

However, representing a variable's state with multiple independent components and combining them with the Cartesian product means that probabilistic propagation is often intractable. For example, if C and a coparent each have two components then their child will have four. If the child's coparent also has four components then the grandchildren will have 16, and so forth.

Fogelberg et al. [3] outlines four ways of addressing this problem. Propagating the fuzzy expected value, which can be calculated using *fuzzy integration*, means that FBN propagation is usually only a small factor less efficient than classic BN propagation. Further, propagating the fuzzy expected value

does not appear to have any disadvantages when compared to classic probabilistic propagation, and the graphs that it is markedly worse for are often intractable as classic BN anyway. For that reason it is adopted in this paper.

Consider the variable $X = [\{f_{0.4}, t_{0.6}\}_{0.2}, \{f_{0.8}, t_{0.2}\}_{0.5}, \{f_{0.1}, t_{0.9}\}_{0.3}]$. The fuzzy expected value of X is easily calculated analytically, and this is shown in equation 2.

$$\begin{aligned} X &= [\{f_{0.9}, t_{0.1}\}_{0.2}, \{f_{0.8}, t_{0.2}\}_{0.5}, \{f_{0.1}, t_{0.9}\}_{0.3}] \\ &\rightarrow [f_{0.9\times0.2+0.8\times0.5+0.1\times0.3}, t_{0.6\times0.2+0.2\times0.5+0.9\times0.3}] \\ X &\rightarrow [f_{0.61}, t_{0.39}] \end{aligned} \tag{2}$$

3 An Example Network

Figure 1 shows the structure of the BN we will be using. It is a classic example, and describes the relationships between the sky (C, cloudy or not), whether or not it rains (R), whether or not the sprinkler is turned on (S) and the state of the grass (wet, or not, W).

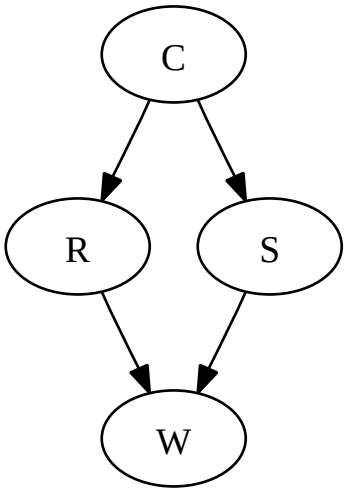


Figure 1: The cloud-sprinkler-rain-grass network.

The parameters for this BN are shown in tables 1–4.

Table 1:

$\rightarrow C$	$p(\neg c)$	$p(c)$
	0.5	0.5

Table 2:

$C \rightarrow S$	$p(\neg s)$	$p(s)$
$\neg c$	0.5	0.5
c	0.9	0.1

Table 3:

$C \rightarrow R$	$p(\neg r)$	$p(r)$
$\neg c$	0.8	0.2
c	0.2	0.8

Table 4:

$R, S \rightarrow W$		$p(\neg w)$	$p(w)$
$\neg r$	$\neg s$	1.00	0.00
$\neg r$	s	0.1	0.9
r	$\neg s$	0.1	0.9
r	s	0.01	0.99

4 Forward Propagation

Forward belief propagation in a BN is updating the probability distributions from parents to children, i.e. in the direction of the edges. This means that it only requires relatively straightforward applications of conditional probability calculations. Subsection 4.1 describes forward propagation in a classic BN, subsection 4.2 describes forward propagation in a FBN, and subsection 4.3 analyses the differences between classic and FBN forward propagation.

4.1 Forward Propagation in Classic Bayesian Networks

Assume we we are certain that the sky is cloudy (C is observed to be in the state c, perhaps we looked up). We want to know the posterior probability distribution for the grass, $p(W|c)$.

The effect of c on the updated conditional probabilities of R and S are just read off the respective tables (2 and 3):

- $S = \{f_{0.9}, t_{0.1}\}$
- $R = \{f_{0.2}, t_{0.8}\}$

Where we have used the notation described in subsection 2.2.

Given the updated uncertain beliefs about the states of R and S we can also calculate our updated belief about W. This is done by taking each conditional distribution in table 4, weighting it by the updated distributions over R and S and summing them.

Doing this, we find that:

$$\begin{aligned} p(W|c) &= 0.18 \times \{f_{1.00}, t_{0.00}\} + 0.72 \times \{f_{0.10}, t_{0.90}\} + \\ &\qquad 0.02 \times \{f_{0.10}, t_{0.90}\} + 0.08 \times \{f_{0.01}, t_{0.99}\} \\ &= \{f_{0.2548}, t_{0.7452}\} \end{aligned} \tag{3}$$

If we observed $\neg c$ then we would calculate $p(W|\neg c) = \{f_{0.451}, t_{0.549}\}$ in the same manner.

4.2 Forward Propagation in Fuzzy Bayesian Networks

Fuzzy BN propagation is described in [3] and readers are referred to that paper for another discussion of the details. In this subsection we illustrate an example of forward propagation which is analogous to that presented in subsection 4.1.

Assume $C = [f_{0.4}, t_{0.6}]$, where we continue to use the notation developed in [3]. Note that if $C = [t_1]$ or $C = [f_1]$ were observed then propagation in an FBN would give identical results to propagation in a classic BN in which c and $\neg c$, respectively, were observed. Given $C = [f_{0.4}, t_{0.6}]$, and by using component-wise propagation, we find that:

- $R = [\{f_{0.8}, t_{0.2}\}_{0.4}, \{f_{0.2}, t_{0.8}\}_{0.6}] \rightarrow [f_{0.44}, t_{0.56}]$
- $S = [\{f_{0.5}, t_{0.5}\}_{0.4}, \{f_{0.9}, t_{0.1}\}_{0.6}] \rightarrow [f_{0.74}, t_{0.26}]$

Where $\rightarrow$ denotes fuzzy integration. Using table 4 and the product t-norm (to find μ for each member of the Cartesian product of the parents), W is calculated in the same way:

$$\begin{aligned} W &\rightarrow 0.3256 \times \{f_1, t_0\} + 0.1144 \times \{f_{0.1}, t_{0.9}\} + \\ &\quad 0.4144 \times \{f_{0.1}, t_{0.9}\} + 0.1456 \times \{f_{0.01}, t_{0.99}\} \\ &\rightarrow [f_{0.3799}, t_{0.6201}] \end{aligned} \tag{4}$$

(5)

4.3 Comparing Classic and Fuzzy Forward Propagation

Forward propagation in a classic BN and a FBN are very similar. The key difference is that the state of a variable in a FBN often has multiple components. Each of these components is propagated independently of the others in the variable, and when a variable has multiple parents, each of which has multiple components, then the parents' components are combined using the Cartesian product set for calculation of the updated beliefs. The child's degree of membership in each member of the Cartesian product is calculated using the product t-norm.

5 Back Propagation

Through the application of Bayes's law, BN also support back propagation, from children to ancestors. How this is calculated is outlined in the following subsection, which presents an example of back propagation in a classic BN. Subsection 5.2 describes back propagation in a FBN and subsection 5.3 compares and contrasts classic and FBN back propagation.

5.1 Back Propagation in Classic Bayesian Networks

By applying Bayes's law, Bayesian networks also support back propagation, from children to ancestors. For example, assume we have observed that the grass is wet (w). What does this tell us about the sprinkler, the rain, and the cloudiness of the sky?

To calculate this we can use Bayes's Rule[3], which shows that:

[3]Or a message passing algorithm which also uses Bayes's Rule but which does not force us to calculate the full joint.

$$p(r|w) = \frac{p(r,w)}{p(w)} = \frac{p(w|r) \cdot p(r)}{p(w)} \tag{6}$$

And similarly for $p(\neg r|w)$, $p(s|w)$ and $p(\neg s|w)$.

Working with equation 6, we can find $p(r)$ and $p(w)$ by integrating over their respective ancestors. In subsection 4.1 we almost did this for w, by calculating $p(w|c)$ and $p(w|\neg c)$ instead. Thus we can complete the integration by summing over C now:

$$p(w) = 0.5 \times p(w|c) + 0.5 \times p(w|\neg c) = 0.6471 \tag{7}$$

By the same method, $p(R) = \{f_{0.5}, t_{0.5}\}$.

Finally, by summing over S in $p(w|S,r)$, where we need to use $p(S|r)$, we can calculate that $p(R|w) = \{f_{0.2921}, t_{0.7079}\}$. A similar set of calculations for S shows that $p(S|w) = \{f_{0.5702}, t_{0.4298}\}$.

By using the same method and carrying on our calculations, we find that:

$$p(C|w) = \{f_{0.4242}, t_{0.5758}\} \tag{8}$$

5.2 Back Propagation in Fuzzy Bayesian Networks

Similarly, as in a classic BN, we can also do back chaining in a FBN. Assume $W = [f_{0.1}, t_{0.9}]$. Using this we find:

- $R = [\{f_{0.8813}, t_{0.1187}\}_{0.1}, \{f_{0.2921}, t_{0.7079}\}_{0.9}] \rightarrow [f_{0.3510}, t_{0.6490}]$
- $S = [\{f_{0.9379}, t_{0.0621}\}_{0.1}, \{f_{0.5702}, t_{0.4298}\}_{0.9}] \rightarrow [f_{0.6070}, t_{0.3930}]$

Just as in FBN forward propagation, we use the product t-norm with the Cartesian product to find the expected state of C.

$$\begin{aligned} C &\rightarrow 0.2131 \times \{f_{0.6897}, t_{0.3103}\} + 0.1379 \times \{f_{0.9524}, t_{0.0476}\} + \\ &\quad 0.3939 \times \{f_{0.1220}, t_{0.8780}\} + 0.2551 \times \{f_{0.5556}, t_{0.4444}\} \\ &\rightarrow [f_{4681}, t_{5319}] \end{aligned} \tag{9}$$

5.3 Comparing Classic and Fuzzy Back Propagation

As in forward propagation, components within a variable are propagated independently, and the Cartesian product and product t-norm is used to combine different parents' components.

As is reasonable and expected, because the state in the classic BN (w) and the FBN ("0.9 w") were similar, the results are as well. However, their interpretation is somewhat different.

In the classic BN there is approximately a 42% chance that it was cloudy today, whatever it means for the weather to be cloudy. Does a single passing cloud on an otherwise sunny day make the day cloudy? Is it some proportion of the day? The FBN expected state is still vague, but less so. We would expect the cloudiness (whether that is measured by time or heaviness) to have been about half as much as it could be at most.

Through this analysis we begin to see how FBN may guide a more nuanced interpretation.

6 Explaining Away

Finally, we consider the phenomena of "explaining away", in which the posterior probability of a variable is affected because one of its co-parents was observed[11].

Although it is initially counter-intuitive (why is the probability of rain reduced when I learn that the sprinkler has been turned on?) careful thought and further analysis shows that the rain and sprinkler are indirectly related.

6.1 Explaining Away in Classic Bayesian Networks

For example, if we know that the grass is wet, but that it rained, then the posterior probability that the sprinkler was on is reduced. This is because the rain "explains away" the wetness of the grass:

$$p(S|r, w) = \{f_{0.8055}, t_{0.1945}\} \tag{10}$$

Thus, despite the grass completely soaked, we expect that the sprinkler was mostly off, rather than mostly on.

6.2 Explaining Away in Fuzzy Bayesian Networks

Finally, we present the phenomena of explaining away in a FBN. Assume we know that the grass is completely soaked ($W = [t_1]$) but that $R = [f_{0.25}, t_{0.75}]$. What is the expected state of the sprinkler?

Using the product t-norm on the Cartesian product of the parents and performing propagation for each pair of components, we find:

$$
\begin{aligned}
S &\rightarrow 0 \times \{f_{0.9325}, t_{0.0675}\} + 0.25 \times \{f_0, t_1\} + \\
&\quad 0 \times \{f_{0.9785}, t_{0.0215}\} + 0.75 \times \{f_{0.8055}, t_{0.1945}\} \\
&\rightarrow [f_{0.6041}, t_{0.3959}] \qquad (11)
\end{aligned}
$$

6.3 Comparing Classic and Explaining Away

Because fuzzy states are not as absolute as discrete states, the magnitude of explaining away is somewhat reduced; however, we can see that it still occurs as we expect. Even though w was observed, the distribution over S was much more biased towards the discrete Boolean value $\neg s$ than s. This is because the state of R was more like r than $\neg r$.

7 Discussion

Having presented the bare numerical results of classic and FBN propagation it is now possible to describe and interpret them in more detail.

7.1 Interpretations

The first important factor to consider and which has been largely implicit and ignored in our calculations is a semantic definition of $C = [\alpha_f, (1-\alpha)_t]$, and likewise for the other variables.

There appear to be two intuitive semantic mappings of C. The first is that it represents the proportion of the day that was cloudy. In this case $C = [f_1, t_0]$ is a day with constantly and completely blue skies, and $C = [f_0, t_1]$ is a day that is constantly cloudy.

The second is that it represents the heaviness and storminess of the cloud, with $C = [f_1, t_0]$ representing entirely blue skies and $C = [f_0, t_1]$ representing raging thunderstorms, a cyclone, or similar right now, or over the past 24 hours, or since sunrise, or since today began, or similar (we should definite both dimensions carefully for our model to be most useful).

Similarly, R's state (W's state) could represent the heaviness of the rain (wetness of the grass) or the proportion of the day that it rained for (that the grass was wet for). These are all important design decisions that need to be considered explicitly but which are easy to forget with a Boolean or multinomial model.

One approach which is prompted by this analysis is as follows. Rather than having just one C variable, split it into two: C_P, which denotes the

proportion of today which was cloudy, and C_H, which denotes the heaviness of the cloud. A possible structure for this model is shown in figure 2. Such a formulation is a much more nuanced representation of the system, and would not have been considered if not prompted by the FBN analysis.

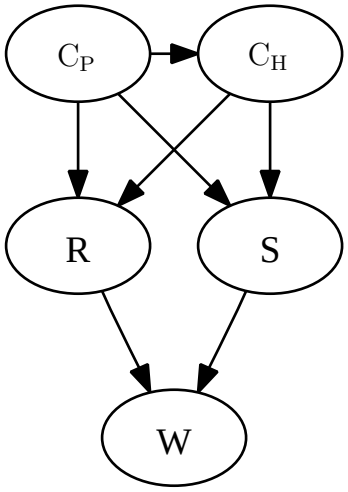


Figure 2: A revised cloud-sprinkler-rain-grass network, suggested by FBN-inspired analysis.

Whether or not this model is better or necessary depends on the application, but it is an advantage of FBN that it forces us to consider the meaning of our model in such detail.

7.2 Comparison

How do the results of classic and fuzzy propagation compare? Two important factors must be briefly noted.

7.2.1 Types of State

Firstly, the results of the FBN propagation are expected values. These are different from a probability distribution, the maximum likelihood state or the raw fuzzy state, made up of fuzzy probability distributions. As noted in [3], using the expected value does not bias further belief propagation. Further, the expected value can be meaningfully compared with observations when scoring and searching over models.

In contrast, the result of classic BN propagation is a probability distribution. This can be trivially converted into the expected value and offers

the same advantages. However a probability distribution does not have any further advantages, unless the distribution itself is necessary.

Further, an equivalent distribution over a discrete variable can be constructed from a fuzzy expected value just by discarding the fuzzy information. For example, if $C = [f_{0.6179}, t_{0.3821}]$ then $C = \{f_{0.6179}, t_{0.3821}\}$ could be an appropriate discrete distribution over it.

Although the probabilistic and fuzzy states of C are superficially and notationally very similar it is important to note that they have different semantics. However, we can directly compare the expected value with observations, or the constructed distribution with other information, and so check that the results match our expectations.

7.2.2 Observations

In addition, the results are also not directly comparable because the observations when propagating over a classic BN and when propagating over a FBN were different. For example, in forward propagation over a classic BN, the state c (equivalently, $C = [1_t]$) was observed, but for the FBN the state $[0.4_f, 0.6_t]$ was observed. Had the fuzzy values been discrete, with complete and sole membership in just one value (true or false) then the results and necessary calculations would have been identical.

However, reviewing the results shows that they can all be easily and reasonably interpreted. Consider $S = [0.6041_f, 0.3959_t]$, the expected value of S, given that the grass is soaked and that it has rained a lot. This state suggests that we can expect the sprinkler to have been turned on roughly 40% of the time. Although this might not be what a person would have done it probably accurately reflects a weather-sensitive automatic system's behaviour. Further, in many situations this expected value is more informative for us than knowing that there was a 20% chance the sprinkler was on today (whatever it means for the sprinkler to be on) even though it rained (all day, was a thunder storm) and the grass is wet. Because there is no gradation or subtlety in discrete values their interpretation can be more difficult and requires more forethought and knowledge of the system's design.

7.3 Final Words

This paper has described the formalisation of FBN and touched on some of the conceptual foundations. It has illustrated belief propagation over a FBN in a range of situations; it has also analysed the differences in belief propagation and interpretation with classic BN. These differences suggest

advantages that FBN may offer in certain situations and for doing certain types of model design.

References

[1] F. Bobillo and U. Straccia. A fuzzy description logic with product t-norm. In *Proceedings of the 16th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2007)*, pages 652–657, London (United Kingtom), July 2007.

[2] Christopher Fogelberg. Fuzziness, probability and fuzzy probability: A conceptual analysis. Current theoretical research., August 2008.

[3] Christopher Fogelberg, Vasile Palade, and Phil Assheton. Belief propagation in fuzzy Bayesian networks. In Ioannis Hatzilygeroudis, editor, *1st International Workshop on Combinations of Intelligent Methods and Applications(CIMA) at ECAI'08*, pages 19–24, University of Patras, Greece, 22 July 2008. ISBN 978-960-89282-7-5.

[4] 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`.

[5] 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.

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

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

[8] Xing-Chen Heng and Zheng Qin. FPBN: A new formalism for evaluating hybrid Bayesian networks using fuzzy sets and partial least-squares. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *ICIC (2)*, volume 3645 of *Lecture Notes in Computer Science*, pages 209–217, Hefei, China, August 23–26 2005. Springer. ISBN 3-540-28227-0.

[9] 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/.

[10] 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.

[11] Michael P. Wellman and Max Henrion. Explaining "explaining away". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15 (3):287–292, March 1993. URL http://ai.eecs.umich.edu/people/wellman/pubs/pami93.pdf.

[12] 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.

[13] Lotfi A. Zadeh. Generalized theory of uncertainty (GTU) — principal concepts and ideas. *Computational Statistics & Data Analysis*, 51(1): 15–46, 2006.

[14] 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.

Appendix D

Fuzziness, Probability, and Fuzzy Probability

This appendix contains the text of our current work on fuzzy probability. We expect to submit this to UAI in November. It uses a spatial (dimensional) conceptualisation of variables and their *states* and *range* of *values*. This is done with the aim of clarifying the possible definitions, interpretations, and semantics of a fuzzy Bayesian network.

Fuzziness, Probability, and Fuzzy Probability: A Conceptual Analysis

Christopher Fogelberg Phil Assheton

September 14, 2008

Abstract

NB: This report is in-progress. The analysis probably needs to be split into two papers. It also needs greater integration with possibility theory[1] and the generalized theory of uncertainty[16].

Contents

1 Introduction

This paper considers *fuzziness* and uncertainty (probability) and how they might interact. It does this by conceptually analysing variables (discrete, continuous, fuzzy) and probability distributions. The essential idea behind all fuzzy concepts is that an element's membership in a set is continuous, not absolute. Classically, in a Boolean system, today is either a hot day or it is not. However, in a fuzzy system, today might be a member of the hot day set with degree 0.3. Readers are referred to Zadeh's [15] original publication for information on logical connectives and similar in fuzzy systems.

The paper is organised as follows. Subsection 1.1 describes the notation we have developed. Section 2 introduces the context, essential terminology and example (temperature) that we will use in this paper. It also conceptualises continuous and discrete variables, setting the stage for the rest of the paper.

Section 3 broadens this foundation and explores what it means for a variable to be fuzzy. Section 4 briefly analyses probability distributions (PDs) in a way that allows us to consider fuzziness and probability simultaneously.

Using the research in sections 2–4, section 5 considers what it means for there to be a PD over a fuzzy variable. Section 6 outlines several ways we could draw samples from a PD over a fuzzy variable.

Section 7 concludes the paper. It suggests several situations where fuzzy probability could be useful in subsection 7.1. It then summarises the conceptual analysis and what this has suggested for *fuzzy probability* in subsection 7.2.

1.1 Some Notation

This paper uses the notation developed in [4] and fine-tuned in [3]. Multinomial probability distributions are denoted with curly brackets, and the probability of a sample being a particular *value* is subscripted. For example, equation 1 shows a probability distribution. There is a 0.3 probability of a sample from this distribution being *lo*, a 0.5 probability of it being *mid* and a 0.2 probability of it being *hi*.

$$\{lo_{0.3}, mid_{0.5}, hi_{0.2}\} \tag{1}$$

Similarly, a fuzzy variable has its *state* denoted with square brackets, and its membership in each *component* of its state is subscripted. For example, equation 2 shows a state which has 0.3 membership in *lo*, 0.5 membership in *mid* and 0.2 membership in *hi*. This notation is similar to the standard mathematical notation for inclusive ranges (e.g. $[0, 1]$), but is distinct.

$$[lo_{0.3}, mid_{0.5}, hi_{0.2}] \tag{2}$$

Although equation 1 and 2 may initially look similar it is important to note that the two represent very different states of the variable they are associated with. Maintaining the distinction between fuzziness and probability while working with them simultaneously is the goal of this paper, and it is important not to accidentally conflate them.

Finally, note that the two notations can be combined and used to represent *mixed* states. A partially uncertain state (equation 3) is an example of a mixed state. Such states will be discussed in section 5.

$$[lo_{0.3}, \{lo_{0.8}, mid_{0.1}, hi_{0.1}\}_{0.7}] \tag{3}$$

2 What is a variable?

Consider: what is a variable? Our answer to this question introduces the terminology and many of the concepts used later in the paper.

An example of a variable is T, which represents and refers to the temperature. This exists independently of its state, such as 29.4 °C or "Warm"[1]. The semiotic concepts of *denotata* and *sign* (or *object* and *representatem*) clarify this distinction[2; 11].

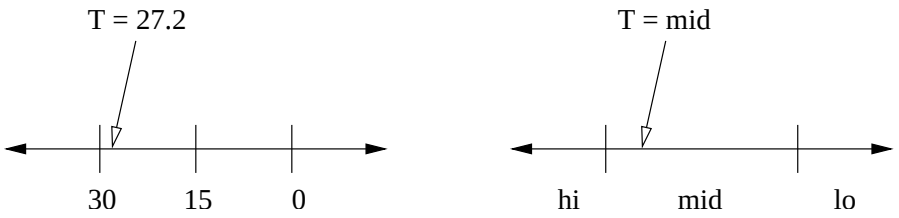
not the same as the variable itself. Although we can often conflate them, it is important for them to be conceptually distinct when we consider variables which are simultaneously uncertain and fuzzy.

The state of a variable is some member of its (possibly infinite) *range*, which is denoted R. The range is a set. An element of this set is a *value*. For example if T is discrete then its range might be *lo*, *mid* and *hi*, or if it is continuous then it may have any value in the range $-273.15\,^{\circ}\mathrm{C} = 0\,^{\circ}\mathrm{K} < T < \textit{arb-max}$.

A variable with a continuous (infinite) range can be thought of as occupying a point in a continuous, (possibly unbounded) one dimensional space. This point is the variable's state. Analogously, the state of a discrete variable (e.g. one with the range *lo*, *mid* and *hi*) can be thought of as occupying a point in a discrete one dimensional space. See figure 1.

The state of a discrete variable can also be thought of occupying a single point in a $|R|$-dimensional space, where each state is binary and the valid

[1]The state of a variable could also be described as its "value", however we give that term a precise, technical definition in the next paragraph.



(a) A continuous variable (b) A discrete variable

Figure 1: A one dimensional conceptualisation of continuous and discrete variables. The state of a continuous variable is represented by a single point in the variable's space. Similarly, the state of a discrete variable is represented by a single point in a discrete space.

states are at the origin in all but one dimension. This perspective on discrete variables will prove most natural, and we return to it shortly. See figure 2.

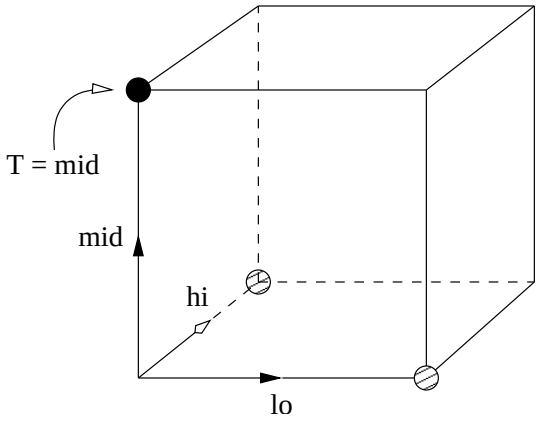


Figure 2: A binary R-dimensional conceptualisation of a discrete variable. The front bottom left corner of the cube is the zero point and denotes "not lo, not mid, not hi". The valid states of T in the full space are denoted by the small patterned circles, the observed state of T is denoted by the small filled circle.

In summary, a variable defines a space (its range), and the state of the variable is some point (value) in this range.

This general framework can be applied to a wide range of situations. In particular, we have successfully used it and its extensions for Bayesian networks with discrete variables and for fuzzy Bayesian networks (FBN)[4]. Subsection 7.1 discusses other potential applications of this conceptualisation.

3 What is a fuzzy variable?

Traditionally, the state of a fuzzy variable is described by the membership (μ) of an underlying continuous value in one or more *fuzzy sets*. Typically, this membership is restricted so that $\sum \mu = 1$. For example, if the variable T was continuous and then fuzzified then its state would be its degrees of membership in the fuzzy sets *lo*, *mid* and *hi*.

To emphasise the similarity of fuzzy sets with discrete values, we will also refer to fuzzy sets as *fuzzy values* (FV). Then, in the same way that the state of a discrete variable can be conceptualised as being a point in a binary $|R|$-dimensional space, the state of a fuzzy variable can also be conceptualised as a point in a continuous $|R|$-dimensional space.

Each FV, e.g. *lo*, is associated with a membership function which determines the degree of membership of the underlying continuous value of a state in that FV. This is analogous to a hard discretisation of a continuous variable, such a discretisation could be thought of as using a set of rectangle functions which have an empty intersection and whose union is valid over the whole domain of the continuous variable.

Thus, just as a discrete variable can also be seen as a point in a higher dimensional space, fuzzification can be thought of as projecting a continuous variable into a higher dimensional space.

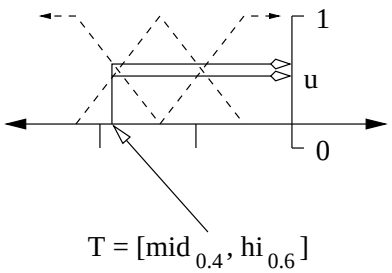
After T has been fuzzified it is no longer a point in a one dimensional space, or a point in a binary R-dimensional space. Instead it is a point in an analogous and (larger) continuous R-dimensional space. Thus, compared to the underlying continuous value (e.g. 29.4 °C), the fuzzy state or T is in a higher dimensional *fuzzy space*. For example, T's fuzzy space is 3 dimensional.

3.1 How are fuzzy states constrained?

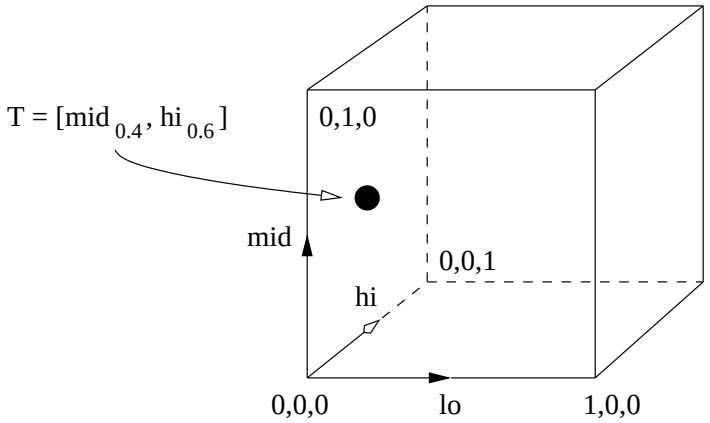
Typically the set of membership functions are defined so that $\sum \mu = 1$, and also so that $0 \leq \mu \leq 1$. However, although $0 \leq \mu \leq 1$ is possible for each FV, the FV are also very interdependent, else the condition $\sum \mu = 1$ would not hold. This can be seen in figure 4.

For this reason fuzzification is actually the projection of a variable's state to a subspace of the whole fuzzy space. Thus, some parts of the fuzzy space are *incoherent* and have no relationship with the underlying variable through the membership functions. We will discuss incoherent fuzzy states more in subsection 5.3. The *coherent* fuzzy space is just the set of states which are mapped on to by the membership functions.

One consequence of this analysis is that the coherent fuzzy space will be



(a) The fuzzified underlying interpretation of fuzziness



(b) The spatial interpretation of fuzziness

Figure 3: Two interpretations of fuzziness. In the first, set-oriented, interpretation (figure 3(a)) the fuzzy state of T is conceptualised as the membership of an underlying continuous state in each of the fuzzy sets (fuzzy values). In the second, spatial, interpretation (figure 3(b)) the fuzzy state of T is not necessarily explicitly and directly related to any underlying state.

exactly the same size as the range of the underlying variable (its underlying space) if each possible discrete or continuous state maps uniquely to a fuzzy state. Alternately, the valid fuzzy space will be smaller if several points in the original range map to the same point in the fuzzy space.

This means that there is either a bijective or surjective relationship (respectively) between the range and the valid fuzzy space. For example, if 29.4 °C and 29.4 °C + δ fuzzify distinctly, then it is bijective. If they are equal, or if other pairs of continuous values are equal, then the relationship between the underlying continuous space and the valid subspace of the fuzzy space is surjective.

$$
\begin{aligned}
\mu_{hi}(t) &= 0.6 \\
\therefore x &= 29.4 \\
\therefore \mu_{mid}(x) &= 0.4 \\
\text{and } \mu_{lo}(x) &= 0.0
\end{aligned}
$$

And similarly...

$$
\begin{aligned}
\mu_{mid}(t) &= 0.4 \\
\therefore x &= 29.4 \\
\therefore \mu_{hi}(x) &= 0.6 \\
\text{and } \mu_{lo}(x) &= 0.0
\end{aligned}
$$

Figure 4: The interdependencies amongst the membership functions. If we know a state's membership in any one of the values in its range then we can use that information and the set-oriented interpretation to calculate its membership in all of the other fuzzy values. This makes clear that the coherent fuzzy space can be no larger than the range of the underlying variable.

If the relationship is surjective then fuzzification is throwing away information. The fuzzy variable is less informative (precise, exact) than the underlying variable.

However, if the relationship is bijective then fuzzification has not simplified the description of the variable's state in any information theoretic manner. The fuzzification may have also made the description no less complex, or even more complex. This is the quandary of fuzziness, that it does not appear to offer any advantages.

3.2 How is fuzziness an advantage?

The quandary of fuzziness (subsection 3.1) could also be described as a false promise. Fuzziness appears to simplify variables and recast them in approximate ways while preserving necessary characteristics; however, our analysis has shown that in the general case this is not true.

If knowledge of the precise membership in each fuzzy value is not known or necessary then fuzzification may appear to offer some advantages. Unfortunately, this analysis is flawed. If imprecise fuzzy knowledge is sufficient then equivalent vague discrete or approximate continuous knowledge would also be sufficient. Remember that accurate fuzzification is nothing more than a projection of the variable's state into a higher dimensional subspace,

as discussed above. Inaccurate fuzzification is nothing more than accurately projecting inaccurate states.

Despite this, fuzziness may have a number of advantages in certain situations. The first is that the fuzzy values are linguistic terms which may carry more information for a human. In particular, the relation amongst fuzzy values may concisely express information to a human: knowing that it's 29.4 °C might not be as useful as knowing that it's $[hi_{0.6}, mid_{0.4}]$. This is because the fuzzy state indicates that the temperature is near the top of its range, but still somewhat moderate. The additional knowledge imparted by the fuzzy state may assist human interpretation and design of a fuzzy system.

Fuzziness also has advantages when a detailed mathematical model of the underlying system cannot be tractably constructed or offers few advantages[5].

Thirdly, fuzziness also lends itself to combining several machine learning techniques[12]. If each machine learning technique can be adapted to work with fuzzy variables then fuzziness may work as a common language. This has been shown to be effective in neuro-fuzzy systems[14], in fuzzy-symbolic neural networks and as a useful approximation for hybrid Bayesian networks[6; 10]. It has also shown promise for general *fuzzy Bayesian networks*[4] (FBN). Fuzzy Bayesian networks are networks which combine fuzziness and uncertainty by using fuzzy probability.

To summarise: a fuzzy variable has a state which is the projection of the underlying variable's state to a subspace of a higher dimensional space. This subspace usually has the same size as the variable's original space. Such a fuzzy state probably can't be usefully less precise than the underlying variable could be, however it lends itself to human-assisted machine learning, the combination of machine learning techniques, and it can be fruitfully applied to problems which are difficult to model precisely.

4 What is a probability distribution?

What is a probability distribution (PD)? What is probability, and what is uncertainty? These are deep and complex questions, with roots in philosophy and statistics[7; 8; 13]. We will not substantially consider any of these issues in this paper.

Instead, for convenience, we will adopt a conceptualisation of a PD that usefully allows us to combine the concepts of uncertainty and fuzziness. More detailed investigation of the most appropriate conceptualisation of probability will be described in further research.

We define a PD for a variable as a function (p) which is defined over the entire range of that variable. For practical reasons we assume $0 < p(x) < 1$

(there may also be further theoretical reasons which justify this). Further, a PD p over a variable with range r is linearly normalised so that $\int_r p(x)\,\delta x = 1$.

As examples of probability distributions, consider $T = \{lo_{0.2}, mid_{0.2}, hi_{0.6}\}$ and $T = N(\mu, \sigma^2) = N(18, 16)$. In the first case, the state of T is an uncertain multinomial distribution, and a sample from this distribution will be *lo* 20% of the time, *mid* 20% of the time and *hi* 60% of the time. In the second case T is a normally distributed with mean 18 °C and standard deviation 4 °C.

5 What is a fuzzy probability distribution?

Sections 2, 3 and 4 thus lead to the central question which this paper addresses. What does a PD over a fuzzy variable (a *fuzzy probability distribution*, FPD) look like? This section considers this question, and section 6 completes our initial analysis by suggesting several answers to the related question of sampling from a FPD.

As described above, a PD is a function over a space. Thus a FPD is a function over a fuzzy space. If this distribution is correctly defined then all fuzzy states (coherent and incoherent) may be sampled. Such a distribution may be very high dimensional (with the problems that represents).

For example the variable T has three FV. Consider also the fuzzy variable S, which represents how sunburnt someone gets. This also has three FV. The distribution over T will be a three dimensional distribution. If $p(S,T) = p(S|T) \times p(T)$ then $p(S,T)$ and $p(S|T)$ will both be six dimensional distributions. As the joint distributions and conditional distributions grow in size the curse of dimensionality becomes an important factor, and we have not developed notation which conveniently represents this kind of distribution yet. However, we will return to this kind of distribution shortly, as it is the full and general definition of a FPD.

A subclass of the full and general distribution is a component-oriented model. This uses mixed states. An example of a mixed state was shown in equation 3, another is shown below, in equation 4.

$$T = [lo_{0.2}, mid_{0.1}, \{lo_{0.4}, mid_{0.2}, hi_{0.4}\}_{0.7}] \tag{4}$$

What kind of samples might be drawn from the third component of T in equation 4 is discussed in section 6. One thing this example shows though is that only "0.7 the state of T" is uncertain. The remainder is known. This has interesting semantic consequences. While we discuss the semantics of FPD more in subsection 5.3 it can be noted that a mixed state for T suggests that it has somehow been partially observed. Perhaps the partially uncertain state

of T in equation 4 is the temperature of the United Kingdom, and we only have reports from Scotland so far.

5.1 How is fuzzy probability different from classic probability?

If incoherent states are disallowed, a FPD may appear to be isomorphic with the PD over the underlying variable, or even only surjective if multiple underlying states map to the same fuzzy state. This is due to the interdependencies amongst the membership functions which ensure $\sum \mu = 1$.

For example, given the pre-defined membership functions, T's membership in *lo* also defines its membership in *mid* and *hi*. Likewise, T's membership in *mid* defines its membership in *hi* and *lo*, and T's membership in *hi* defines its membership in *mid* and *lo*. This is as discussed above. Consider figure 5, which is figure 4 but presented again here for the reader's convenience. Maintaining these interdependencies makes fuzzy probability relatively trivial and boring. In the same way that a set-oriented fuzzy state is just a bijective or surjective projection, the interdependencies mean that a FPD will just be a bijective or surjective projection of a PD over the underlying variable.

$$
\begin{aligned}
\mu_{hi}(t) &= 0.6 \\
\therefore x &= 29.4 \\
\therefore \mu_{mid}(x) &= 0.4 \\
\text{and } \mu_{lo}(x) &= 0.0
\end{aligned}
$$

And similarly...

$$
\begin{aligned}
\mu_{mid}(t) &= 0.4 \\
\therefore x &= 29.4 \\
\therefore \mu_{hi}(x) &= 0.6 \\
\text{and } \mu_{lo}(x) &= 0.0
\end{aligned}
$$

Figure 5: The interdependencies amongst the membership functions. This is figure 4 printed again for the reader's convenience.

5.2 What if we relax the interdependencies?

What if we relax the interdependencies amongst the FV? Section 6 describes several ways that a sample could be drawn from a FPD.

However, for a full and general FPD p will be correctly defined ($0 < p(x) < 1$, rather than $0 \leq p(x) < 1$) over the entire $|R|$-dimensional fuzzy space, and not just over the underlying space or the subspace of the fuzzy space which the underlying space is isomorphic or surjective on. This means that $0 < p(\mathbf{x}) < 1$, where $\mathbf{x}$ is a $|R|$-dimensional vector whose elements are in the range $[0, 1]$. For example $p([lo_{[0.2,0.3]}, mid_{[0.1,0.6]}, hi_{[0.4,0.5]}]) = 0.2$, where subscripted $[\ldots]$ are used to indicate ranges over μ. Relaxing the interdependencies in this way has two important consequences.

The first is that a sample from a FPD may have $\mu > 1$. Similarly, if the FPD is part of a mixed state then the sample may have μ greater than the μ of the distribution itself.

This means that the variable's state may have $\sum \mu > 1$. Similarly, it may also have $\sum \mu < 1$. Upon acknowledging this, one asks the question: what would it mean for T to be 0.4 *hi*, 0.5 *mid* and 0.3 *lo*? Depending on the situation it may be possible to find an interpretation of a fuzzy state which drew some meaning from the relative proportion of each FV, and which also drew meaning from $\sum \mu$. Perhaps the latter refers to the quantity or quality of the measurements?

In any case, this situation can be addressed by a simple linear normalisation. Further, although *undersamples* and *oversamples* may seem nonsensical at first glance they may have useful semantic interpretations in certain situations, as we suggest.

Secondly, and more importantly, it breaks the relationship between the underlying variable and the fuzzy values that the membership functions provide. As with undersamples and oversamples, this is semantically vague. However, there is a perfectly reasonable interpretation for some variables which we will sketch out in the next subsection. This interpretation demands that a designer give a more detailed definition of what the state of the variable means and refers to, but it also offers the potential to increase the nuance, richness and expressiveness of the system. By relaxing the dependencies we can differentiate amongst states that would classically be indistinguishable, even with a continuous variable.

5.3 What are the semantics of fuzzy probability?

At first glance, it seems that relaxing the dependencies amongst the different fuzzy values leads to nonsense. Consider T, with its membership in *lo*, *mid*

and hi. Imagine that $t = [lo_{0.5}, hi_{0.5}]$ was a normalised sample for T which had been drawn from a FPD.

What does this sample mean? Assume that equations 5–7 translate a continuous state of T into a fuzzy state.

$$\mu_{lo}(x) = \begin{cases} 1 & \text{if } x < 7 \\ (21 - x)/14 & \text{if } 7 \leq x < 21 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$\mu_{mid}(x) = \begin{cases} 0 & \text{if } x < 7 \\ (x - 7)/14 & \text{if } 7 \leq x < 21 \\ (35 - x)/14 & \text{if } 21 \leq x < 35 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

$$\mu_{hi}(x) = \begin{cases} 0 & \text{if } x < 21 \\ (x - 21)/14 & \text{if } 21 \leq x < 35 \\ 1 & \text{otherwise} \end{cases} \tag{7}$$

With these definitions, there is no single underlying value of T which can give rise to t. It is incoherent, and the meaning of such a sample appears vague at best.

To extract useful meaning from such a sample we need to consider in more depth exactly what a particular underlying value of T represents. In actual fact, a temperature is an aggregate measurement. It might be explicitly aggregate (if T is today's average temperature) or implicitly (if T is the average kinetic velocity of the molecules in some system at time ϕ).

If this detail is important or useful to us then an incoherent sample carries important information, and the details of the specific problem may provide a means for mapping some or all points in the full fuzzy space back to the lower dimensional space that the underlying states lie in. For example, if the components of the fuzzy state represent the kinetic energy of portions of the system, then the underlying continuous temperature of $[lo_{0.5}, hi_{0.5}]$ would be the same as the underlying continuous temperature represented by $[mid_1]$, even though $[mid_1]$ is the only fuzzy representation of this temperature which can be obtained using the membership functions in equations 5, 6 and 7.

Based on this analysis, we can conclude that incoherent samples may be meaningful with more nuanced approaches to fuzzification if those approaches give more detailed semantic interpretations to the membership functions. In particular, this may motivate a viewpoint which views fuzzy-value membership as being calculated from a sample of the components, as we suggested

above for T and the kinetic energy of the underlying molecules. However, when a membership function is nothing more than the mapping $\mathbb{R} \to [0, 1]$, incoherent samples from a FPD present problems.

6 How can we sample from a fuzzy probability distribution?

Using the framework developed in sections 2–5, we have shown that fuzzy probability may provide a window into a more detailed approach to some problems. This section considers how we might draw samples such as $t = [lo_{0.5}, hi_{0.5}]$ from a FPD. There seem to be three kinds of technique that could be used: we can sample the underlying variable, we could sample from the full fuzzy space, or we could divide $\mu = 1$ into portions and sample each portion independently.

6.1 We could sample the underlying variable.

The simplest approach is just to sample the underlying variable. For example, the temperature might be uniformly distributed in the range $[-10\,°\mathrm{C}, 50\,°\mathrm{C}]$. A sample could be drawn from this distribution and the value fuzzified using the membership functions.

This approach discards any of the advantages (and disadvantages) of fuzzy probability. For this reason we will not consider it in any further detail.

6.2 We could fix nothing.

The most complex and general approach is to sample from the fuzzy space. This has R dimensions, as opposed to the underlying variable, which is usually 1 dimensional.

The need to specify one or several such high dimensional probability distributions may introduce too many free parameters into the model that is being used. This could precipitate overfitting and means that more data would be required if machine learning was being used.

The magnitude of the overfitting problem could be reduced by using common approximations, such as noisy-or[9] and multi-dimensional Gaussian distributions[13].

Another problem that might arise if we sample from the full fuzzy space is that some of those areas are nonsensical. For example, we might sample $t = [lo_{0.3}, mid_{0.5}, hi_{0.4}]$, where $\sum_t \mu = 1.2$. This is an oversample. Similarly, undersamples are just as likely, and in many cases it would be very rare for

a sample to have $\sum \mu = 1$. However, this problem can be easily resolved by linearly normalising each component so that $\sum \mu = 1$ for the normalised components. After normalisation some samples will become coherent. There may also be some useful semantic interpretation of such a sample, as discussed in subsection 5.2.

Overall, sampling from the fuzzy space may not be necessary as many of the unnormalised samples will be equal after normalisation, and the full expressive power (subsection 5.3) may not be necessary or applicable. Further, introducing so many free parameters may require too much prior knowledge and make the learning problem too difficult.

6.3 We could use a component-wise approach.

Another third approach which ensures that $\sum \mu = 1$ in a more straightforward manner is to use several PD as components to specify the state of T, and to associate some degree of membership with each PD. This approach to sampling from a FPD is inspired by the component-oriented analysis of mixed states in section 5. Each μ_i represents T's membership in that probability distribution, and a sample of the value v from the PD i with μ_i means that T has membership μ_i in v.

For example, assume the fuzzy uncertain state of T is as shown in equation 8.

$$t = [\{0.3, 0.2, 0.5\}_{0.3}, \{0.2, 0.2, 0.9\}_{0.1}, \{0.5, 0.3, 0.2\}_{0.6}] \tag{8}$$

Consider the first component of T's state t, the FPD $\{0.3, 0.2, 0.5\}_{0.3}$. A sample from this FPD could be one of three values: *lo*, *mid* or *hi*. This is the range of T. With probability 0.3 the sample will be *lo*, with probability 0.2 the sample will be *mid*, and with probability 0.5 the sample will be *hi*. If the sample were *lo* then the sampled state of T would be $[lo_{0.3}, \ldots]$. If the samples from the second and third components were *hi* and *lo* respectively then the sampled state of T would be $[lo_{0.3}, hi_{0.1}, lo_{0.6}] = [lo_{0.9}, hi_{0.1}]$.

Provided some reasonable way of dividing μ up into portions can be found then this approach represents a good compromise. It allows a wider proportion of the fuzzy space to be explored but has fewer parameters and is more tractable than full general sampling, which is likely to be underconstrained. This technique has been adopted in our formalisation of FBN and appears to lead to useful and theoretically reasonable results. FBN are Bayesian networks with simultaneously uncertain and fuzzy variables[4]

7 Conclusion and Contributions

This section concludes our conceptual analysis of fuzziness, probability and fuzzy probability in two stages. First, we briefly enumerate areas where fuzzy probability could be useful. Following that we summarise the theoretical contributions of this research.

7.1 Possible Applications

Possible applications of fuzzy probability and this conceptual analysis are wide ranging. Fuzzy probability has shown promise for FBN[4]. Other areas where the conceptualisation could probably be easily applied include neuro-fuzzy systems[14] and fuzzy-symbolic neural networks.

Further, possibility theory[1] and type II fuzzy logic are examples of other areas of research which a spatial conceptual analysis may be useful for.

7.2 Theoretical Contributions

This paper has made two key contributions.

Firstly, it has provided a conceptual space-based framework which allows fuzziness and uncertainty to be considered simultaneously. This framework allows us to analyse and understand exactly what a variable is in quite a fundamental way, and it facilitates clear and unambiguous thinking.

By extending this analysis from discrete and continuous variables to fuzzy variables it allows us to analyse and understand the latter in the same way, and also to clearly see how fuzzy variables can be related to the variable and its range which underlies them.

Similarly, it allows us to reason about and understand the relationship between a variable and a probability distribution over that variable in a simple, intuitive (spatial) manner.

Elucidating and explaining this framework is the first contribution of this paper.

Secondly, we have used this framework to consider how fuzziness and uncertain can be combined and used simultaneously. We have analysed several semantic interpretations of a FPD, depending on which assumptions we relax. To complete this analysis we have considered how a FPD could be sampled from. Further work is necessary in this area to clarify the range of options and select the best.

Several different options with different advantages have been suggested. These range from sampling from the space of the underlying variable to sampling from the full fuzzy space. Which is most appropriate depends on the

problem and data, and in previous research we have adopted a compromise which allows us to sample from the most meaningful and relevant parts of the fuzzy space.

One cost of using fuzziness is that it requires greater human specification and prior knowledge, to define the membership functions. This problem may be exacerbated with fuzzy probability, if natural and intuitive definitions of the membership functions aren't immediately available.

Overall, we have provided a solid conceptual foundation for further research in combined fuzziness and uncertainty. The meta-concept of a spatial framework could also be useful in thinking about how different machine learning concepts such as symbolic and evolutionary could be combined.

References

[1] Didier Dubois and Henri Prade. Fuzzy sets and probability: misunderstandings, bridges and gaps. *Second IEEE International Conference on Fuzzy Systems*, 2(1993):1059–1068, 1993. doi: 10.1109/FUZZY.1993.327367.

[2] Jennifer Ferreira, Pippin Barr, and James Noble. The semiotics of user interface redesign. In Mark Billinghurst and Andy Cockburn, editors, *AUIC*, volume 40 of *CRPIT*, pages 47–53, Newcastle, NSW, Australia, January/February 2005. Australian Computer Society. ISBN 1-920682-22-8.

[3] Christopher Fogelberg. Belief propagation in fuzzy Bayesian networks: A worked example. In Shamal Faily and Standa Zivny, editors, *2008 Oxford University Computing Laboratory Student Conference*, October 2008. URL `http://syntilect.com/cgf/pubs:fbn-example`.

[4] Christopher Fogelberg, Vasile Palade, and Phil Assheton. Belief propagation in fuzzy Bayesian networks. In Ioannis Hatzilygeroudis, editor, *1st International Workshop on Combinations of Intelligent Methods and Applications(CIMA) at ECAI'08*, pages 19–24, University of Patras, Greece, 22 July 2008. ISBN 978-960-89282-7-5.

[5] Ryan N. Gutenkunst, Joshua J. Waterfall, Fergal P. Casey Kevin S. Brown, , Christopher R. Myers, and James P. Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS Compututational Biology*, 3(10):e189, Oct 2007. doi: 10.1371/journal.pcbi.0030189. URL `http://dx.doi.org/10.1371\%2Fjournal.pcbi.0030189`.

[6] Xing-Chen Heng and Zheng Qin. FPBN: A new formalism for evaluating hybrid Bayesian networks using fuzzy sets and partial least-squares. In De-Shuang Huang, Xiao-Ping Zhang, and Guang-Bin Huang, editors, *ICIC (2)*, volume 3645 of *Lecture Notes in Computer Science*, pages 209–217, Hefei, China, August 23–26 2005. Springer. ISBN 3-540-28227-0.

[7] Edwin Thompson Jaynes. The well posed problem. *Foundations of Physics*, 3:477–493, 1973.

[8] Edwin Thompson Jaynes. *Probability Theory: The Logic of Science*. Cambridge University Press, 2003.

[9] 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/`.

[10] 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.

[11] Charles Peirce. *Collected Papers of Charles Sanders Peirce*, volume 2 of *Elements of Logic*. Harvard University Press, 1932.

[12] 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.

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

[14] Zhenyu Wang, Vasile Palade, and Yong Xu. Neuro-fuzzy ensemble approach for microarray cancer gene expression data analysis. In *Proceedings of the 2006 International Symposium on Evolving Fuzzy Systems*, pages 241–246. IEEE, September 2006. doi: 10.1109/ISEFS.2006.251144.

[15] Lofti Asker Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[16] Lotfi A. Zadeh. Generalized theory of uncertainty (GTU) — principal concepts and ideas. *Computational Statistics & Data Analysis*, 51(1): 15–46, 2006.

Appendix E

GreenSim

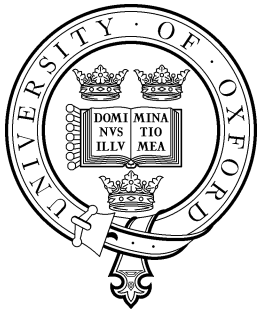
This appendix includes the full text of our technical report on `GreenSim`, the genetic regulatory network simulator which we have written.

Programming Research Group

GREENSIM: A GENETIC REGULATORY NETWORK SIMULATOR

Christopher Fogelberg and Vasile Palade[1]

PRG-RR-08-07



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

[1]Email: christopher.fogelberg@comlab.ox.ac.uk

Abstract

Inference of complete genetic regulatory networks is a central problem in modern bioinformatics. However, because good biological data is still relatively rare, it is hard to evaluate new machine learning techniques for network inference. In this report we describe `GreenSim`, a modular, customisable and extensible genetic regulatory network simulator. It accurately models motifs, non-linear regulatory functions and can generate networks ranging in size from $N = 10^0$ to $N = 10^4$ genes. Code is available online and from the authors.

1 Introduction

Accurate inference of an organism's complete *genetic regulatory network* (GRN) is a central problem in bioinformatics. Knowledge of the GRN is necessary to understand an organism's ontogeny, its phylogenetic relationships and to accurately predict its response to external stimuli and drugs. Only when such relationships are understood can research be conducted quickly and efficiently.

Despite the development and increasing use of high-throughput biotechnology, well understood and validated biological data sets are rare and exist for only a few species. This means that realistically simulating biological data is essential to fully evaluate new machine learning and bioinformatic techniques.

In this paper, we describe `GreenSim`, the genetic regulatory network simulator. In the rest of this section (subsection 1.1) we summarise the features of GRN that must be simulated. Section 2 describes `GeneSim`, and section 3 describes Kyoda's simulator. Section 4 outlines the characteristics and modular structure of `GreenSim`, its usage, and analyses some networks generated by it. Section 5 concludes and outlines future work.

1.1 Characteristics of Genetic Regulatory Networks

The causal relationships between the genes in an organism make up its genetic regulatory network. This network is a directed graph, and an edge from the gene i to the gene j and annotated with a function means that the expression level of i regulates (contributes to, causes) the expression level of j.

Because of their evolutionary origins[16], GRN are characterised by certain common features across cell type and species. This section summarises the most important of these features, and a good GRN simulator should implicitly or explicitly create them.

At the structural level it is important to understand the different distributions across the in-degree and out-degree of genes (variables, nodes) in GRN. Research[2] suggests that GRN are not random directed graphs. Instead, the probability distribution for the out-degree appears to follow a power law[8] with $2 < \lambda < 3$. The distribution for the in-degree appears to be an exponential distribution, and μ is approximately 3 for prokaryotes and in the range 4–8 for eukaryotes[4; 11]. These distributions help create a modular structure in the network. There is no characteristic module size.

GRN are also characterised by the overabundance of *motifs*. A motif is a small subgraph which is much more common in GRN than it would be in a random graph. Motifs

include *auto-regulatory*, expanding *cascade*, *convergence* and triangular *feed-forward* motifs. In a feed-forward motif the gene i regulates j and k, and j regulates k as well. Motifs are detailed in [6]; it is important that they are appropriately present in any simulated GRN.

Two key aspects of the regulatory relationship between any pair of connected genes must be simulated. The first is that genetic regulatory functions are often complex, non-linear relationships whose functional form varies significantly from case to case. The gene i may up-regulate (down-regulate) the gene j very strongly (very weakly) and it may regulated it non-linearly. The nature of these relationships is described in more detail in [3; 5; 17].

Regulatory functions can also be conditional on the current phenotypic context[13]. Furthermore, the actual biological system is stochastic and discrete; however, a discrete simulation is not computationally tractable. Nor is a discrete model. This means that stochastic *biological noise*[12] will exist in any continuous representation of a GRN.

External phenotypic contexts excluded and assuming a weak central dogma, note that non-genetic regulatory influences can be ignored in GRN inference. This is because, under the weak central dogma, the GRN is assumed to be informationally complete. Non-genetic regulatory factors just manifest themselves as (more complex) genetic regulatory relationships.

2 GeneSim

`GeneSim` was developed at Duke University as part of a project which tried to understand songbird singing behaviour[7; 14; 15; 18; 19]. Used in most of the GRN research there, [18] and [19] describe `GeneSim` in detail.

It works as follows. A network of small size ($N \leq 100$) is generated and connected in a uniformly random way. Connections can also be manually specified. Time is measured in discrete intervals and every δ time steps the expression level is recorded as a sample. This interval represents the (hidden) changes in expression level that occur in between successive samples in a time series microarray data set.

$$y_{t+1} = y'_t + A \times (y'_t - b') + \epsilon' \tag{1}$$

Equation 1 defines the vector equation which calculates the expression level of all genes (capped at a maximum of 100 and a minimum of 0) at time $t + 1$, given the expression level of all genes at time t. Column vectors are denoted with prime ('), and transposition is implicit. ϵ' represents the biological noise and is sampled from a vector Gaussian with $\mu = 0$ and σ_b as defined by the user. A represents the relationship between $y_{t+1}(k)$ and y_t. It defines a linear difference equation for each gene. b' represents the constitutive level of gene expression.

The constitutive level of gene expression can be explained as follows. Imagine that each gene has some "normal" level of expression. This is its constitutive level. If i up-regulates j and i is present at more than its constitutive level then it will cause j to be more strongly expressed. On the other hand, if i is expressed less strongly than usual

then the absence of i will cause a lower-than-usual expression of j. It is claimed by Yu et al. that this model leads to more biologically plausible time series.

Note two key aspects of `GeneSim`. Firstly, non-linear gene interactions are not modeled. These are essential for biological realism. Secondly, the network edge matching algorithm does not create motifs or biological distributions over k_{in} and k_{out}.

`GeneSim` also does not consider the phenotypically conditional nature of gene regulation. However this not a serious problem, for two reasons. Biological data sets typically come from just one phenotypic situation, and inferring a single network given samples generated from potentially contradictory regulatory networks is a different and much more difficult research question. In summary, `GeneSim` is a useful simulator, however it does not model several important features of biological GRN.

3 Kyoda's Simulator

The simulator in Kyoda et al. [10] uses ODEs to model the regulatory relationships. Each gene was randomly matched to between 1 and k other genes. These genes and random rate constants defined its regulatory ODE. The auto-regulatory motif was not allowed. Networks of up to $N = 100$ with $k = 2$ and $N = 20$ with $k = 8$ were simulated.

4 GreenSim

`GreenSim` is the GRN simulator we have developed in MatLab. Inspired by `GeneSim`, it aims to realistically simulate any kind and size of GRN. This section describes its functional characteristics (4.1), its modular structure (4.2), how to use it (4.3) and analyses some `GreenSim`-generated networks (4.4).

4.1 Characteristics

`GreenSim` can simulate GRN of size $N = [10^0, 10^4]$. It generates realistic non-linear regulatory functions, and ensures that the degree distributions are accurate. These distributions also lead to the emergence of motifs in the simulated networks. Section 5 discusses extensions that would allow it to model multiple phenotypic contexts.

4.2 Structure

`GreenSim` is designed in a modular fashion, and each of the modules can be replaced without affecting the others. For example, the edge matching module was recently replaced, and no other portions of the code needed to be modified. The algorithm is structured as follows.

First, *half edges* (in-edges and out-edges for each gene) are generated according to the appropriate distributions. Next, the half edges are matched by randomly permuting the genes and then matching the out-edges with the nearest in-edges. This reflects the biological reality (genes are frequently physically located near the genes they regulate)

and is also necessary to create realistic *technical noise*[9]. Technical noise is described shortly.

Next, the linear and non-linear regulatory functions are generated. Each gene has a linear regulatory function as described in equation 1. Call the change in the expression level of the gene k according to its linear regulatory function l_k and its expression level after the previous time step n_k.

A 2nd order non-linear regulatory function for each pair of genes which regulate k is defined as shown in equation 2. NL_k is a random matrix which defines a weight for the non-linear contribution of each pair of genes i, j $(i < j)$ which regulate k. The bracketed term in the equation, $\big((y_t - b)' \times (y_t - b)\big)$, takes the product of the expression level of each pair of genes. This product and NL_k are multiplied together member-wise and the members of the resulting matrix are summed to calculate the total non-linear regulatory contribution. The updated expression level for k is $n_k + nl_k + l_k$.

$$nl_k = \sum \big(NL_k \times (y_t - b)' \times (y_t - b)\big) \tag{2}$$

As in `GeneSim`, expression levels are updated at discrete intervals and can be sampled at intervals of size δ. Such a sequence of *samples* is called a *set of samples*. Samples can also be corrupted by parameterised levels of technical noise[9]. Technical noise refers to uncertain, missing or inaccurately measured gene expression levels in the set of samples. This occurs because of limitations in the biotechnology used to gather samples. Three kinds of technical noise are modeled.

Spot noise is the low, independent probability that any particular gene expression value in a set of samples will be covered. This can be caused in biological datasets by a faulty microarray or by poor handling of the microarray after the experiment has concluded.

Span noise is the low, independent probability that some contiguous subset of the genes in a sample is covered. Because of the edge matching algorithm such genes are likely to be directly involved in each others regulation. The size of the span is normally distributed according to user-specified parameters.

Value noise is added to the biological noise and further corrupts the gene expression levels. It represents the ambiguity that can be caused by using mRNA concentrations and fluorescence levels as a proxy for the level of gene activity[1].

4.3 Using GreenSim

In this subsection we describe how to use `GreenSim` to simulate GRN and generate samples from them.

Readers are encouraged to refer to the MatLab source files, comments and included help documentation for further information.

4.3.1 Generating a Network Structure

To generate a network structure, users call the `genNetwork(n, mu, maxReg, noise)` function. This function returns a network structure, and the parameters are as follows:

n is the number of genes

μ is the location parameter for the exponential distribution over the gene in-degree.

maxReg is the maximum regulatory influence that any one gene or pair of genes can have on another. A and NL_k are in the range $(0, 1)$ and are scaled by this parameter.

noise is the standard deviation of the biological noise in the network, σ_b.

In most cases, the value of the `noise` parameter in network generation should be no more than $0.5 \times$ `maxReg`. If this is not the case then the influence of regulatory genes may be obscured and the result will be a random walk.

4.3.2 Generating A Set of Samples

A set of samples can generated by calling `genSampleSet(net, inity, num, delta)`. This function returns a matrix of gene expression levels uncorrupted by noise. The (i, j)'th entry in the returned matrix is the expression level of the i'th gene in the j'th sample. The function's parameters are:

net is a network structure, as returned by `genNetwork`

inity is a column vector specifying the initial expression level of each gene.

num is the number of samples (columns) to be generated.

δ is the number of hidden gene expression updates between each sample.

In real biological situations, gene expression levels may change between samples. As in `GeneSim`, this is represented in `GreenSim` by a δ parameter and the gene expression values are updated δ times according to the regulatory functions and biological noise between each sample.

For example, if `num` $= 3$ and $\delta = 3$ then the gene expression values would be updated 9 times according to their regulatory functions, and the third, sixth and ninth vectors would make up a set of samples.

Note that the relative magnitude of `maxReg` and the δ used when generating samples will affect the difficulty of any network inference. We suggest that $\delta \leq \frac{1}{\texttt{maxReg}}$, else expression levels may change too much and in too complex a manner between samples.

A set of samples can be corrupted by the `corruptSample` function. This function takes a set of samples as its first parameter and then five other parameters for the technical noise. Covered gene expression values are denoted with -1.

Finally, a set of samples can be transformed using the `logSample` function. The log (base 2) of each gene expression value is calculated (adjusted for raw values less than $\epsilon = 0.000001$) and covered values in the transformed set of samples are denoted with 999.

4.3.3 Calculating Statistical Network Properties

The `genNetworkStats(net)` function calculates statistical metrics for a network. This function takes a network structure as a parameter and returns a statistics structure. The statistics it calculates are analysed in subsection 4.4.

4.3.4 Saving and Loading GreenSim Data

A network can be saved or loaded using the `writeNet` and `readNet` functions. Similarly, a sample can be saved or loaded using the `writeSample` and `readSample` functions and a network statistics structure can be saved or loaded using the `writeStats` and `readStats` functions.

For samples and statistics structures the `read...` functions take a source filename as a parameter. To save a sample or statistics structure the `save...` functions take a variable of the appropriate type as their first parameter and a target filename as the second parameter.

`writeNet` and `readNet` have the same parameters and parameter ordering as the functions for samples and statistics structures. However, networks are serialised to more than one file. This means that only a root filename needs to be specified in the function calls, and the functions will append the appropriate extensions to save or load the network.

To obtain an edge matrix which can be converted into a `dot` file for `GraphViz` the `writeEdgeMatrix` function can be used.

4.4 Examples and Analysis

This section describes some networks generated using `GreenSim`, ranging in size from 3 to 10^4 genes. Subsubsection 4.4.1 introduces the analysis by providing visualisations of networks' edge matrices, for networks ranging in size from 3 to 10^3 genes. Subsubsection 4.4.2 provides more detailed statistical analysis, for networks ranging in size from 3 to 10^4 genes.

4.4.1 Visual Analysis

This subsubsection contains figures and commentary depicting the edge matrices of networks ranging in size from 3 to 1000 genes. Each edge and each pair of edges to a gene defines a regulatory function which relates the expression level of the parent(s) at time t to the expression level of the child at time $t + 1$. The child's expression level can be calculated in full by using all parents and all pairs of parents, as described in subsection 4.2.

The network shown in figure 1 illustrates `GreenSim`'s ability to generate very small networks, and also shows how the combination of the different distributions over the in and out-edges can lead to auto-regulation and feed-forward motifs.

Figure 2 illustrates the automatic generation of modules, cascade and convergence motifs by the edge matching algorithm, even in small `GreenSim`-generated networks.

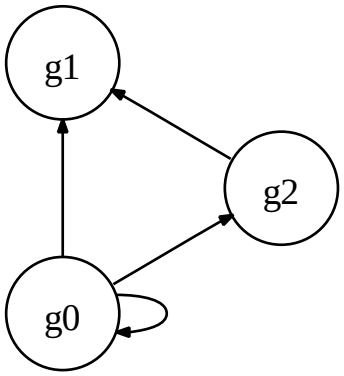


Figure 1: A `GreenSim` network with 3 genes.

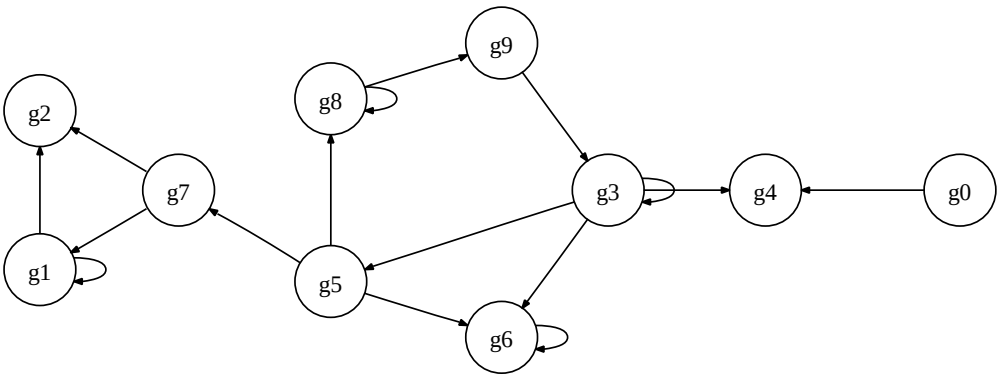


Figure 2: A `GreenSim` network with 10 genes.

Figure 3 shows the expected growth in module size as the network grows (a consequence of the power law distribution over out-edges). Cascade and convergent motifs also occur more often in larger networks.

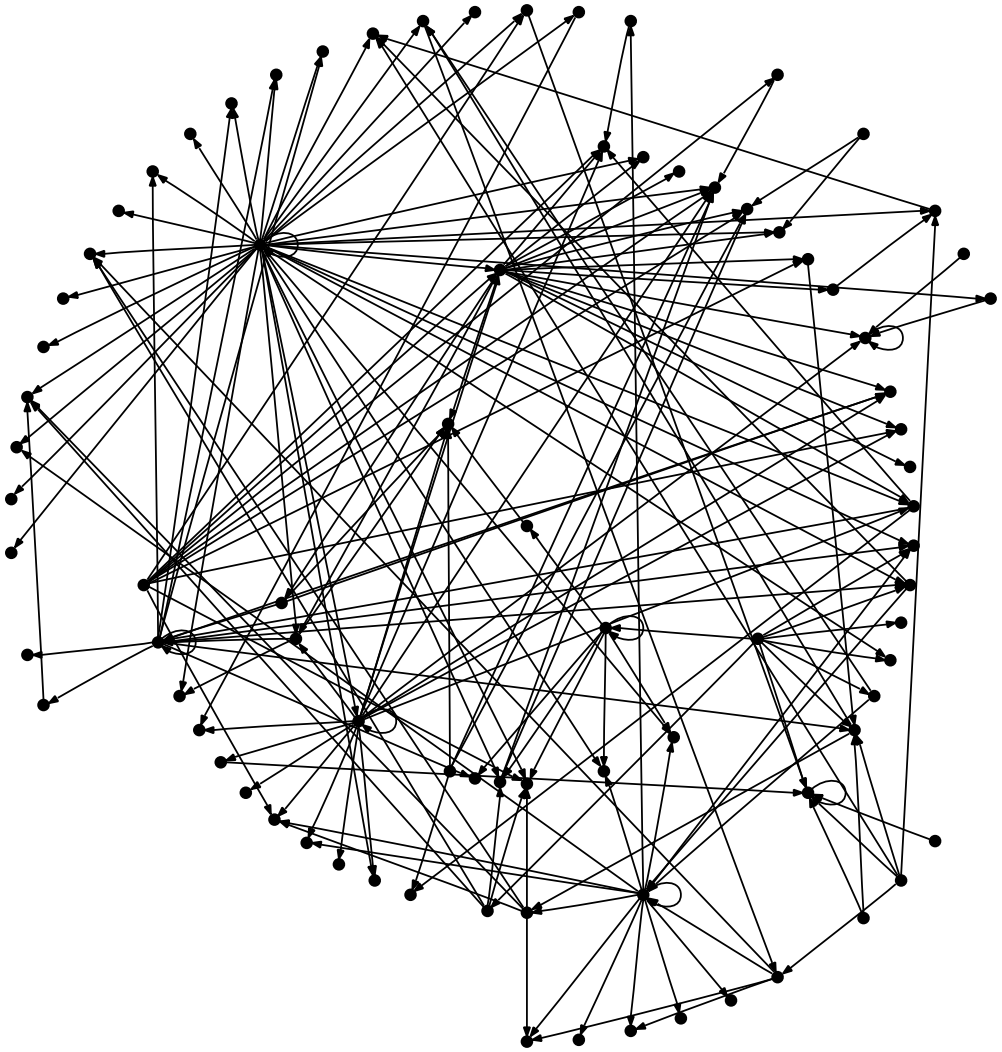


Figure 3: A `GreenSim` network with 100 genes.

Figure 4 is very difficult to interpret at the printed resolution. However, it is clear that maximum module size continues to grow while smaller modules and motifs also continue to exist.

Observation of these graphs shows that `GreenSim` networks possess many of the characteristics of real biological networks, including a messy modular structure, and the autoregulatory, feed-forward, cascade and convergence motifs.

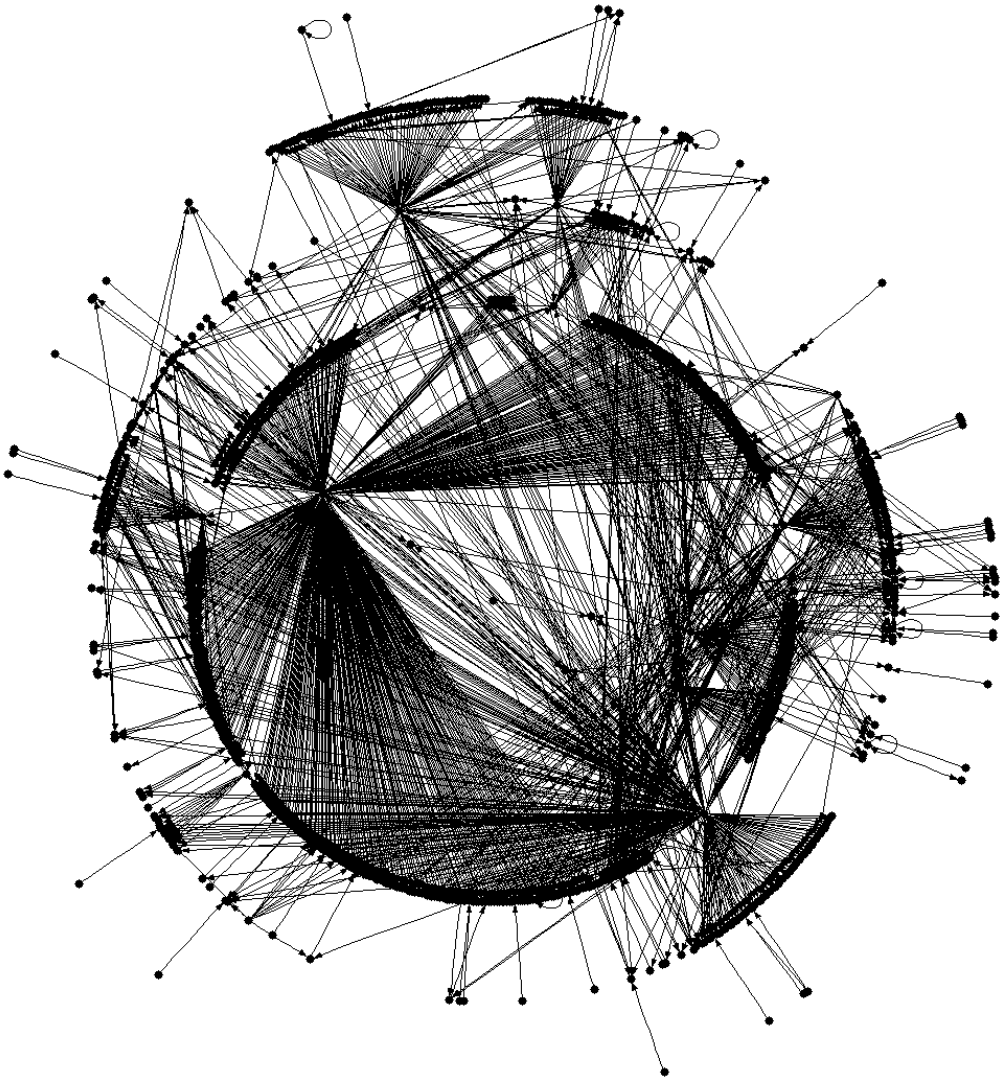


Figure 4: A `GreenSim` network with 1000 genes.

4.4.2 Statistical Analysis

This subsubsection presents and analyses some basic statistical properties of networks generated using `GreenSim`. Statistics for several different network sizes are shown in table 1 and more detailed discussion follows.

Table 1: This table contains the averages of 10 networks for each n. Statistics are displayed to 4 significant figures.

Statistic / Num. of genes	$n = 3$	$n = 10$	$n = 10^2$	$n = 10^3$	$n = 10^4$
Time (s)	0.1073	0.1404	0.2268	5.057	725.5
Total Edges	4.3	22	344.5	3518	35190
$max(k_{in})$	2.1	4.4	15.2	21.8	27.3
$max(k_{out})$	2.6	8.6	72.9	791.1	7968
Auto-regulatory motif	1.5	3.2	8.8	35.8	188.2
Feed-Forward Motif	0.3	13.1	723.4	4569	45150

The networks used to calculate the statistics displayed in table 1 were generated on an Asus M6A laptop (Pentium 4 1.8 GHz, 1 GB RAM, Win XP Pro). Observation of the results shows that the time it takes to generate a network is non-linear. Some of this is due to virtual memory paging issues. The remaining non-linearity appears to be created when the half edges are being matched.

For $n \geq 100$ we see that the number of edges per gene stabilises at ≈ 3.5. Given that the networks were generated with exponential in-degree $\mu = 3$ this is unsurprising. Furthermore in the full vectors of in and out-edges for each network we have observed that k_{out} and k_{in} are distributed correctly.

Analysis of the motifs is more interesting. It is clear that the incidence rate of the feed-forward motif is much larger in the mid-size networks. When $n = 3$ there is an average of 0.1 feed-forward motifs per gene and there are approximately 4.5 such motifs per gene when $n \geq 1000$. In contrast there are more than 7 feed-forward motifs per gene when $n = 100$. We suspect that this is a side-effect of the edge-matching algorithm but do not think it is a problem.

The change in the incidence rate of the auto-regulatory motif for very large networks is potentially more problematic. The source of this trend is difficult to determine. Given the network size, the correlation between the total number of edges and the incidence rate is almost 0. However, confusingly and independently of the number of genes, the correlation between the total number of edges and the auto-regulatory incidence rate is 0.792. There is also some correlation between network size and the incidence rate (0.747). The correlations of $max(k_{in})$, the network size and other combinations are all less than the correlation of the total number of edges and the incidence rate, independent of the network size, but greater than 0.6.

In short, naive correlations are uninformative and it is not clear what is driving this trend. However, it may be due to the distributions over k_{in} and k_{out} and the edge

matching algorithm. Because the range of k_{out} grows much more quickly than the range of k_{in}, and because the edge matching algorithm matches genes in descending order of the number of out-edges, if a network has "hub" genes which connect to the majority of other genes in the network then these genes may absorb all or most of the in-edges of a large proportion of most of the genes. This means that most genes cannot auto-regulate.

5 Discussion and Future Work

`GreenSim` is importantly different from previous simulators such as `GeneSim`. In particular, `GreenSim`:

- Explicitly and correctly models the in and out degree distributions.
- Implicitly models the other important features of a GRN, through its explicit modeling of the in and out degree distributions.
- Uses non-linear regulatory functions.
- Accurately models the various kinds of noise which exist in biological data sets.
- Is modular, easily customised and highly parameterised.
- Generates GRN ranging in size from 10^0 to 10^4 genes.

The visual analysis, statistical analysis and realistic regulatory functions suggests that `GreenSim` can be used to simulate and sample from realistic GRN of widely varying size.

One aspect of `GreenSim` that needs further investigation and which may need improvement is the edge matching algorithm. This is because it appears to under-represent the auto-regulatory motif in very large and very small GRN. If the biological literature confirms that this is inaccurate then a new matching algorithm would need to be developed.

Like `GeneSim`, the regulatory functions in `GreenSim` are only representative of a single phenotypic context. Although this is not a problem, for the reasons discussed in section 2, `GreenSim` could be extended to simulate the phenotypically context-dependent logical discontinuities in the regulatory functions that can be seen in biological organisms[13] as well.

For example, rather than generating just one set of functions (A and $\{NL_k\}$), `GreenSim` could be modified so that it generates m sets of functions. The set of functions (A^i and NL^i_k) used to generate each sample could be changed with some probability or according to environment-state variables. If a different edge matrix was necessary then the genes could be re-permuted and re-matched for each member of this set of regulatory functions.

Bibliography

[1] 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.

[2] 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`.

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

[4] 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`.

[5] 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`.

[6] Christopher Fogelberg and Vasile Palade. Machine learning and genetic regulatory networks: A review and a roadmap. Technical Report CS-RR-08-04, Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford, OX1-3QD, April 2008.

[7] 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.

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

[9] 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`.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

Appendix F

Machine Learning and Genetic Regulatory Networks

The full text of our technical report[27] on machine learning for genetic regulatory networks is available online at the following URLs:

- `http://syntilect.com/cgf/pubs:grnlr`
- `http://www.comlab.ox.ac.uk/techreports/cs/2008.html`

It has also been rewritten and submitted to "Foundations on Computational Intelligence" as a book chapter[29]. This shorter text is available on request.

The review summarises the relevant network biology of genetic regulatory networks and a range of previous machine learning research in the field. It goes on to analyse the gaps in this research and some of the problems facing regulatory network inference. It is written as an introduction to the field.

Appendix G

Ockham's Razor and Bayesian Reasoning

The full text of our position paper on Bayesian reasoning and model inference is available online at `http://syntilect.com/cgf/pubs:bayespp`.

This position paper describes one (mis)interpretation of a common statistical argument. This misinterpretation has lead some authors to erroneously claim that Bayesian inference objectively entails and justifies Ockham's razor.

We are currently rewriting the position paper and will submit it as a letter to a machine learning journal. The abstract of the unmodified position paper is below.

> This paper emphasises the importance to Computer Science of correctly understanding philosophical concepts. It shows that accurately interpreting Ockham's razor could lead to better Bayesian model scoring. We first consider the argument that Bayesian inference automatically embodies and justifies Ockham's razor, because it automatically balances goodness of fit with the number of parameters. Philosophical analysis shows that this is incorrect. Through theoretical considerations and by analysing examples we see the kinds of situations in which Bayesian reasoning embodies Ockham's razor and the kinds of situations in which it does not. We then explore the ramifications of our analysis for Bayesian model scoring.

Bibliography

[1] Dana Angluin. Computational learning theory: Survey and selected bibliography. In *STOC*, pages 351–369. ACM, 1992.

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

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

[4] 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`.

[5] 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`.

[6] 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.

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

[8] James C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms*. Kluwer Academic Publishers, Norwell, MA, USA, 1981. ISBN 0306406713.

[9] Richard Bonneau, David J. Reiss, Paul Shannon, Marc Facciotti, Leroy Hood, Nitin S. Baliga, and Vesteinn Thorsson. The inferelator: An algorithm for learning parsimonious regulatory networks from systems-biology data sets *de novo*. *Genome Biology*, 7(R36), 2006.

[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] Zeke S. H. Chan, Lesley Collins, and N. Kasabov. Bayesian learning of sparse gene regulatory networks. *Biosystems*, 87(5):299–306, 2007.

[14] 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.

[15] Raymond J. Cho, Michael J. Campbell, Elizabeth A. Winzeler, Lars Steinmetz, Andrew Conway, Lisa Wodicka, Tyra G. Wolfsberg, Andrei E. Gabrielian, David Landsman, David J. Lockhart, and Ronald W. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73, 1998.

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

[17] 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.

[18] 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.

[19] Arnaud Devillez, Patrice Billaudel, and Gérard Villermain Lecolier. A fuzzy hybrid hierarchical clustering method with a new criterion able to find the optimal partition. *Fuzzy Sets and Systems*, 128(3):323–338, 2002.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

[25] Christopher Fogelberg. Fuzziness and an initial plan. Briefly summarises relevant research into fuzzy clustering and fuzzy BN. Presents a draft research plan., 2008.

[26] Christopher Fogelberg. Clustering for fuzzy BN models. Describes some of the clustering algorithm characteristics which are necessary for FBN inference., 2008.

[27] Christopher Fogelberg and Vasile Palade. Machine learning and genetic regulatory networks: A review and a roadmap. Technical Report CS-RR-08-04, Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford, OX1-3QD, April 2008.

[28] Christopher Fogelberg and Vasile Palade. GreenSim: A genetic regulatory network simulator. Technical Report PRG-RR-08-07, Computing Laboratory, Oxford University, Wolfson Building, Parks Road, Oxford, OX1-3QD, May 2008.

[29] Christopher Fogelberg and Vasile Palade. *Genetic Regulatory Networks: A Review and a Roadmap*, chapter 0. Springer Verlag, 2008.

[30] Christopher Fogelberg, Vasile Palade, and Phil Assheton. Belief propagation in fuzzy Bayesian networks. In Ioannis Hatzilygeroudis, editor, *1st International Workshop on Combinations of Intelligent Methods and Applications(CIMA) at ECAI'08*, pages 19–24, University of Patras, Greece, 22 July 2008. ISBN 978-960-89282-7-5.

[31] Nir Friedman and Zohar Yakhini. On the sample complexity of learning Bayesian networks. In Eric Horvitz and Finn Verner Jensen, editors, *UAI*, pages 274–282. Morgan Kaufmann, 1996.

[32] 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`.

[33] 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`.

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

[35] Nicholas Paul Gauthier, Malene Erup Larsen, Rasmus Wernersson, Ulrik de Lichtenberg, Lars Juhl Jensen, Soren Brunak, and Thomas Skot Jensen. Cyclebase.org a comprehensive multi-organism online database of cell-cycle experiments. *Nucleic Acids Research*, 36(S1):D854–859, 2008. doi: 10.1093/nar/gkm729.

[36] Farah Gould. *Engineering Time: Practical Applications of Modern Quantum Metaphysics*. PhD thesis, RSSS, Australian National University, Canberra, Greater Antipodes, December 2079.

[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] 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`.

[40] 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.

[41] Erez Hartuv and Ron Shamir. A clustering algorithm based on graph connectivity. *Information Processing Letters*, 76(200):175–181, 2000.

[42] Erez Hartuv, Armin O. Schmitt, Jörg Lange, Sebastian Meier-Ewert, Hans Lehrach, and Ron Shamir. An algorithm for clustering cDNA fingerprints. *Genomics*, 66(3):249–256, 2000. A preliminary version appeared in Proc. RECOMB '99, pp. 188–197.

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

[44] David Haussler, Michael Kearns, and Robert E. Schapire. Bounds on the sample complexity of Bayesian learning using information theory and the vc dimension. *Machine Learning*, 14(1):83–113, 1994. ISSN 0885-6125. doi: http://dx.doi.org/10.1023/A:1022698821832.

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

[46] 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`.

[47] Ralf Herwig, Albert J. Poustka, Christine Meuller, Christof Bull, Hans Lehrach, and John O'Brien. Large-scale clustering of cDNA fingerprinting data. *Genome Research*, 9(11):1093–1105, November 1999.

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

[49] 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.

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

[51] 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.

[52] 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.

[53] A. Agung Julius, Michael Zavlanos, Stephen Boyd, and George J. Pappas. Genetic network identification using convex programming. Technical Report MS-CIS-07-20, University of Pennsylvania, Stanford University, Department of Electrical and Systems Engineering, Department of Electrical Engineering, July 2007.

[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] 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`.

[57] 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`.

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

[59] Mohit Kumar, Regina Stoll, and Norbert Stoll. A min-max approach to fuzzy clustering, estimation, and identification. *IEEE Transactions on Fuzzy Systems*, 14(2):248–262, 2006.

[60] 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.

[61] 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.

[62] 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.

[63] 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.

[64] 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.

[65] 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.

[66] 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`.

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

[68] Wolfgang Marwan, Annegret Wagler, and Robert Weismantel. A mathematical approach to solve the network reconstruction problem. *Mathematical Methods of Operations Research*, 67(1):117–132, February 2008.

[69] 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.

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

[71] 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.

[72] 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.

[73] 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`.

[74] 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`.

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

[76] Denis Noble. *The Music of Life: Biology beyond the Genome*. Oxford University Press, June 2006. ISBN 0199295735. URL `http://www.amazon.ca/exec/obidos/redirect?tag=citeulike09-20\&path=ASIN/0199295735`.

[77] 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.

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

[79] 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.

[80] 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.

[81] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

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

[83] Gabriella Rustici, Juan Mata, Katja Kivinen, Pietro Lió, Christopher J Penkett, Gavin Burns, Jacqueline Hayles, Alvis Brazma, Paul Nurse, and Jürg Bähler. Periodic gene expression program of the fission yeast cell cycle. *Nature Genetics*, 36(8):809–817, August 2004.

[84] Thomas Schlitt and Alvis Brazma. Modelling gene networks at different organisational levels. *FEBS Letters*, 579:1859–1866, March 2005. ISSN 0014-5793.

[85] Dale Schuurmans and Russell Greiner. *Learning to Classify Incomplete Examples*, pages 87–105. MIT Press, 1997. URL citeseer.ist.psu.edu/schuurmans93learning.html.

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

[87] 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.

[88] 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.

[89] 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).

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

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

[92] 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.

[93] Patrick K. Simpson. Fuzzy min-max neural networks — part 2: Clustering. *IEEE Transactions on Fuzzy Systems*, 1(1):32–45, February 1993. ISSN 1063-6706. doi: 10.1109/TFUZZ.1993.390282.

[94] Will Smart and Mengjie Zhang. Applying online gradient-descent search to genetic programming for object recognition. In Martin Purvis, editor, *Australasian Workshop on Data Mining and Web Intelligence (DMWI2004)*, volume 32 of *CRPIT*, pages 133–138, Dunedin, New Zealand, 2004. ACS. URL `http://crpit.com/confpapers/CRPITV32Smart.pdf`.

[95] Will Smart and Mengjie Zhang. Continuously evolving programs in genetic programming using gradient descent. In R. McKay, editor, *Proceedings of the 2004 Asia-Pacific Workshop on Genetic Programming*, page 16pp, December 2004.

[96] 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.

[97] 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.

[98] Alan D. Sokal. Transgressing the boundaries: Toward a transformative hermeneutics of quantum gravity. *Social Text*, 46/47:217–252, 1997.

[99] 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.

[100] 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.

[101] Michio Sugeno. An introductory survey of fuzzy control. *Information Sciences*, 36(1–2):59–83, 1985.

[102] 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.

[103] 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.

[104] 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.

[105] Michael E. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001. URL citeseer.ist.psu.edu/tipping01sparse.html.

[106] 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.

[107] 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.

[108] George E. Tsekouras, Haralambos Sarimveis, Evagelia Kavakli, and George Bafas. A hierarchical fuzzy-clustering approach to fuzzy modeling. *Fuzzy Sets and Systems*, 150(2):245–266, 2005. URL http://dblp.uni-trier.de/db/journals/fss/fss150.html#TsekourasSKB05.

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

[110] Hongchun Wang, Qingxi Shi, and Qin Zhang. Fuzzy reasoning in continuous causality diagram. In *Proceedings of the International Conference on Information Acquisition*, pages 46–49, 21–25 June 2004.

[111] 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.

[112] Xuanli Lisa Xie and Gerardo Beni. A validity measure for fuzzy clustering. *IEEE Trans. Pattern Anal. Mach. Intell.*, 13(8):841–847, 1991. ISSN 0162-8828. doi: http://dx.doi.org/10.1109/34.85677.

[113] 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.

[114] 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.

[115] 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.

[116] 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.

[117] Lofti Asker Zadeh. Fuzzy sets. *Information Control*, 8:338–353, 1965.

[118] Mengjie Zhang and Will Smart. Genetic programming with gradient descent for multiclass object classification. In *Proceedings of the 7th European Genetic Programming Conference*, volume 3003 of *Lecture Notes in Computer Science*, pages 399–408, 2004.

[119] Mengjie Zhang and Will Smart. Learning weights in genetic programs using gradient descent for object recognition. In Franz Rothlauf, Juergen

Branke, Stefano Cagnoni, David W. Corne, Rolf Drechsler, Yaochu Jin, Penousal Machado, Elena Marchiori, Juan Romero, George D. Smith, and Giovanni Squillero, editors, *Applications of Evolutionary Computing, EvoWorkshops2005: EvoBIO, EvoCOMNET, EvoHOT, EvoIASP, EvoMUSART, EvoSTOC*, volume 3449 of *LNCS*, pages 417–427, Lausanne, Switzerland, 30 March-1 April 2005. Springer Verlag. ISBN 3-540-25396-3. doi: doi: 10.1007/b106856.

[120] 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.