INFLUENCE MODELLING AND LEARNING BETWEEN DYNAMIC BAYESIAN NETWORKS USING SCORE-BASED STRUCTURE LEARNING

RITESH AJOODHA

A Ph.D. thesis submitted to the Faculty of Science, University of the Witwatersrand, in fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

Doctor of Philosophy
Supervised by Dr. Benjamin Rosman
School of Computer Science and Applied Mathematics
The University of the Witwatersrand

May 24, 2018

Ritesh Ajoodha: *Influence Modelling and Learning between Dynamic Bayesian Networks using Score-based Structure Learning,* Doctor of Philosophy.

A Ph.D. thesis submitted to the Faculty of Science, University of the Witwatersrand, in fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

SUPERVISOR:
Dr. Benjamin Rosman

LOCATION:
Johannesburg, South Africa

To Mrs. Soonpathee Ajoodha [1929-2016]

Mrs. Soonpathee Ajoodha was born in Putnispruit, South Africa, in 1929 on the 12th of January. She was 8 years old when she lost her mother. In 1947 she married Mr. Bhoodan Ajoodha and lived on 56 Croesus Avenue in Newclare. In 1960 she leased her farm in Lenasia (95 Nirvana Avenue) and, together with her husband, she bought the first house in Lenasia and had 12 children (6 sons and 6 daughters), 35 grand-children, 43 great-grand-children, and 3 great-great-grand-children. As one of her 93 descendents, I am proud to have shared some of her stories and experiences. She was the first Indian woman to travel on board the Concorde[1], crossing two international time-lines. At the age of 87 she lived a fruitful and memorable life, leaving behind an impressive legacy.

(a)

(b)

(c)

Figure 1: (a) Mr. and Mrs. Ajoodha abord the Concorde; (b) Mr. and Mrs. Ajoodha at their home, the first house in Lenasia; (c) Mrs. Ajoodha celebrating my 2nd Birthday.

1 The Concorde was a British-French turbojet-powered supersonic passenger jet.

ABSTRACT

Although partially observable stochastic processes are ubiquitous in many fields of science, little work has been devoted to discovering and analysing the means by which several such processes may interact to influence each other. In this thesis we extend probabilistic structure learning between random variables to the context of temporal models which represent partially observable stochastic processes. Learning an influence structure and distribution between processes can be useful for density estimation and knowledge discovery.

A common approach to structure learning, in observable data, is score-based structure learning, where we search for the most suitable structure by using a scoring metric to value structural configurations relative to the data. Most popular structure scores are variations on the likelihood score which calculates the probability of the data given a potential structure. In observable data, the decomposability of the likelihood score, which is the ability to represent the score as a sum of family scores, allows for efficient learning procedures and significant computational saving. However, in incomplete data (either by latent variables or missing samples), the likelihood score is not decomposable and we have to perform inference to evaluate it. This forces us to use non-linear optimisation techniques to optimise the likelihood function. Furthermore, local changes to the network can affect other parts of the network, which makes learning with incomplete data all the more difficult.

We define two general types of influence scenarios: direct influence and delayed influence which can be used to define influence around richly structured spaces; consisting of multiple processes that are interrelated in various ways. We will see that although it is possible to capture both types of influence in a single complex model by using a setting of the parameters, complex representations run into fragmentation issues. This is handled by extending the language of dynamic Bayesian networks to allow us to construct single compact models that capture the properties of a system's dynamics, and produce influence distributions dynamically.

The novelty and intuition of our approach is to learn the optimal influence structure in layers. We firstly learn a set of independent temporal models, and thereafter, optimise a structure score over possible structural configurations between these temporal models. Since the search for the optimal structure is done using complete data we can take advantage of efficient learning procedures from the structure learning literature. We provide the following contributions: we (a) introduce the notion of influence between temporal models; (b) extend traditional structure scores for random variables to structure scores for temporal models; (c) provide a complete algorithm to recover the influence structure between temporal models; (d) provide a notion of structural assembles to relate temporal models for types of influence; and finally, (e) provide empirical evidence for the effectiveness of our method with respect to generative ground-truth distributions.

The presented results emphasise the trade-off between likelihood of an influence structure to the ground-truth and the computational complexity to express it. Depending on the availability of samples we might choose different learning methods to express influence relations between processes. On one hand, when given too few samples, we may choose to learn a sparse structure using tree-based structure learning or even using no influence structure at all. On the other hand, when given an abundant number of samples, we can use penalty-based procedures that achieve rich meaningful representations using local search techniques.

Once we consider high-level representations of dynamic influence between temporal models, we open the door to very rich and expressive representations which emphasise the importance of knowledge discovery and density estimation in the temporal setting.

RELATED WORK

Some of the work in this thesis appears in the following peer reviewed publications:

[Ajoodha and Rosman 2018] **Ritesh Ajoodha** and **Benjamin Rosman**. *Advancing Intelligent Systems by Learning the Influence Structure between Partially Observed Stochastic Processes using IoT Sensor Data. AAAI SmartIoT: AI Enhanced IoT Data Processing for Intelligent Applications, New Orleans Riverside, New Orleans, Lousiana, USA.* February 2018.

[Ajoodha and Rosman 2017] **Ritesh Ajoodha** and **Benjamin Rosman**. *Tracking Influence between Naïve Bayes Models using Score-Based Structure Learning. PRASA-RobMech*. December 2017.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Dr. Benjamin Rosman, for supporting me over the five years that we have been working together, and for giving me the freedom and tools to explore and discover new areas of probabilistic AI. Dr. Rosman has been a hugh inspiration to me and our weekly meetings have been my best learning experiences at WITS.

My other colleagues, teachers, and mentors have also been very supportive to my research. Including (Dr.) Richard Klein; (Prof.) Turgay Celik; Kgomotso Monyepote; (Miss.) Diane Coutts; (Dr.) Nishana Parsard; Andrew Francis; (Dr.) Pravesh Ranchod; Mike Mchunu; (Dr.) Hima Vadapalli; (Dr.) Hairong Bau; (Prof.) Michael Sears; (Prof.) Montaz Ali; (Prof.) Clint Van Alten; (Prof.) Sigrid Ewert; (Prof.) Ebrahim Momoniat; (Prof.) Joel Moitsheki; and (Prof.) Charis Harley.

I would also like to thank the members of the Robotics, Autonomous Intelligence and Learning (RAIL) Laboratory for their support; constructive criticisms; and suggestions. Namely Ofir Marom; Steve James; Phumlani Khoza; Richard Fisher; Logan Dunbar; Ashley Kleinhans; Beatrice van Eden; Benjamin van Niekerk; (Dr.) Andrew Saxe; Adam Earle; and Jason Perlow.

It would have significantly harder for me to complete the background of this research had it not been for (Prof.) Daphne Koller and (Prof.) Andrew Ng who provided the world with Coursera[2]. By making free quality education available to the world, these authors have enabled many students the opportunity to learn about the pure sciences from world renowned researchers, despite their location and financial backgrounds. In particular, I would like to thank (Prof.) Daphne Koller and (Prof.) Judea Pearl for providing relevant and useful educational lectures which are available on YouTube.

Thank you to the NIPS; AISTATS; PRASA; Ph.D. panel members ((Dr.) Pravesh Ranchod, (Prof.) Michael Sears, and (Prof.) Montaz Ali); the participants at the Bayesian Inference Summer School in Battys Bay (namely Prof. Udo von Toussaint, Prof. Kevin H Knuth, and Prof. John Skilling); my anonymous Ph.D. examination panel; and various other anonymous reviewers for feedback and suggestions to strengthen our research tools and clarity of mathematical descriptions. I appreciate the time and effort spent to review my research papers and thesis. I would like to extend this gratitude to the representatives at the Deep Learning Indaba 2017[3], which took place at WITS this year, and Dr. Asad Mahmood who provided a useful discussion and review before the final submission of this work.

2 https://www.coursera.org/

3 http://www.deeplearningindaba.com

I would like to thank the members of the Stack-overflow community who helped me through the development of the tools necessary to implement and experiment on these ideas. I greatly appreciate your dedication and commitment to strengthening the programming community by being the most user-friendly trouble-shooting tool.

I would formally like to acknowledge the following funding awarded during my doctoral studies, without which this research would not have been possible: the Teaching Development Grant Collaborative Project funded by the Department of Higher Education, South Africa (Ref. APP-TDG-0020/21); the WITS Staff Bursary (Ref. A0034695); the NRF Scarce Skills Doctoral Scholarship (Ref. SFH14072479869); and the Doctoral Postgraduate Merit Award Scholarship (Ref. 468045). A full list of funding and awards can be found on my website[4].

My parents' tireless efforts to provide a quality education and a loving living environment for my brothers and I will always be appreciated. I cherish all of our family bonding times and know that I would have not been able to complete this research without them.

I would like to thank my brothers, Ravish and Rushil, and sister-in-law, Meera, for all their support and tolerance in having to listening to me prattle-on about my research. I thank my Nanie for making me all those delicious study-treats when I battled through the development of this project.

4 ritesh.ajoodha.co.za

DECLARATION

I, Ritesh Ajoodha (student number: 468045), hereby declare the contents of this Ph.D. thesis to be my own work. This thesis is submitted for the degree of Doctor of Philosophy in Computer Science at the University of the Witwatersrand. This work has not been submitted to any other university, or for any other degree.

Johannesburg, South Africa

Ritesh Ajoodha, May 24, 2018

CONTENTS

1 INTRODUCTION

1.1 INTRODUCTION

Stochastic processes are commonly used for describing the evolution of variables over time. Despite this, the question of how several of these processes may influence each other has received little attention in the literature. One may choose to represent these influence structures as a dynamic probability distribution which models the likelihood of the influence relations, between observable and partially observable multi-dimensional processes, relative to the data.

In the observable case, modelling the influence between multi-dimensional processes requires establishing a hypothesis test to assess if influence relations exist between these sets of processes. One would have to assess exactly the extent to which the sets of processes influence each other and move towards establishing an algorithm to capture them. Even the most optimistic computational complexity approximation suggests a search space that is super-exponential given the length and number of multi-dimensional processes involved.

Modelling the influence structure between a set of *partially observable* multiple dimensional processes is significantly harder. Partially observable processes have missing data which causes the likelihood of the processes to the data, even modeled independently, to have multiple optima. This leaves us with an intractable problem in establishing a hypothesis test to relate these sets of multi-dimensional processes for recovering their influence relation.

As an example, suppose we want to learn the influence distribution of traffic on a road network. We may have a set of features that describes each road over time (e.g. light level; number of cars, weather, number of collisions, etc.). These temporal observations may tell us about latent features such as the traffic condition on each road over a period of time. Our task is to then learn the influence of traffic between all of the roads over time.

In this thesis, we assess this problem. We devise a complete algorithm to recover the influence relations between a set of partially observable multi-dimensional stochastic processes by using score-based structure learning. In score-based structure learning we search for the most suitable structure by using a scoring metric to value structural configurations relative to the data. We also provide empirical results to demonstrate the effectiveness of our approach.

This introductory section is structured as follows. Section 1.2 provides a detailed assessment of the problem of learning influence between partially observed multi-dimensional processes; Section 1.3 provides an overview of the literature of structure learning as a viable solution to this problem; Section 1.4 provides an overview of the framework used to solve this problem; Section 1.5 outlines our major contributions; and finally, Section 1.6 provides an outline of the structure for the remainder of this thesis.

1.2 PROBLEM STATEMENT

Processes interact in various ways (eg. the stock market or traffic conditions of roads etc.). Understanding how the structure of interactions manifest between families of processes is useful for knowledge discovery (eg. learning the structure of the influence of traffic over time) and general density estimation (eg. estimating the distribution of traffic over a period of time). However, we often only observe the consequence of the interaction between several processes (since aspects of them are latent) and do not actually get to observe and therefore study the process directly (eg. inferring the latent variable traffic condition using observable features, since we do not have access to the variable traffic condition directly).

This leaves the task of being given a set of observable features (the data), that describe a set of latent processes, and recovering the underlying structure of the influence relations between these processes. More precisely, consider Figure 2 which shows an example of a set of partially observed processes each given by 3 features (A, B, and C). Our task is to deduce the probability distribution that most likely resembles the model which generated the data. That is, we need to learn some structure between processes and parameter setting (which describe the probability of data values) that is capable to describe this temporal distribution. The structure of influence in this case is given by the black bold solid arrows.

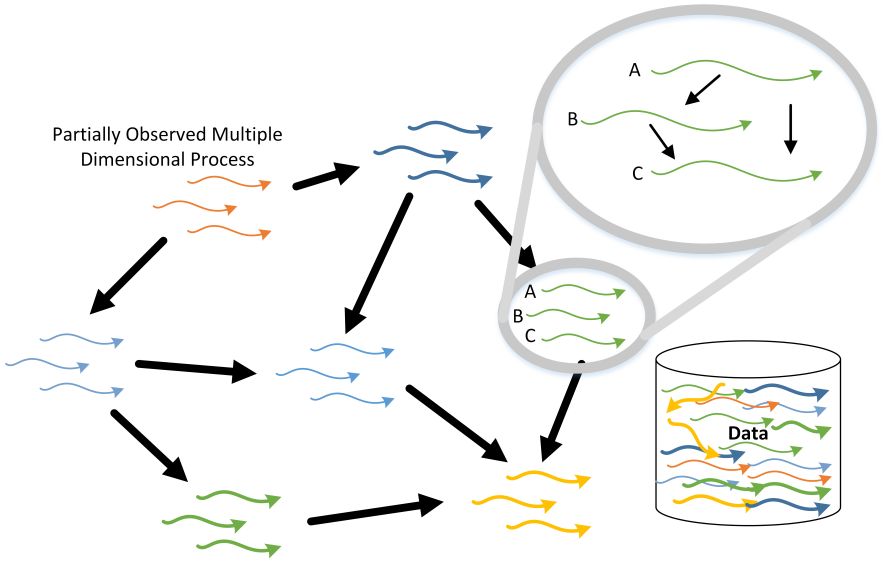


Figure 2: An influence structure between multi-dimensional processes. The colored solid arrows indicate features which describe processes and the black bold solid arrows indicate the influence between these processes.

A closer inspection of this problem reveals another learning task to identify the probability distribution for each individual multi-dimensional process with respect to the given set of observations. However, often we need to aggregate observable temporal features to tell us about high-level features which we can use to better describe each multi-dimensional process.

In this task we are only given the temporal features as data and are asked to deduce the dynamic probability distribution which explains the interaction between these

processes as illustrated by Figure 2. In scenarios where we have complete data we often resort to maximum likelihood estimation (MLE), which optimises the likelihood function for a set of parameters that describes the processes with respect to the data. Unfortunately, in this case we are not given complete data and instead are given a set of partially observable processes, each with a set of temporal observations, that we want to discover the probabilistic structure of influence between. In this case the likelihood function has multiple optima which we can not optimise by just using the derivative of the likelihood function [Koller and Friedman 2009].

Recovering the influence structure from temporal incomplete data, that is induced by a set of temporal observations, appears in most practical applications where we wish to perform density estimation to given observations; or learn the latent characteristics of an environment, which changes over time, for knowledge discovery (e.g. learning how traffic on roads in an area influence each other over time, or how the protein composition of a cell changes as its conditions change). In the next section we provide a brief overview of current practices in structure learning to address this problem.

1.3 OVERVIEW OF LITERATURE

As far as we know this particular problem has not been solved in current literature, however, several foundational learning practices can make our task simpler. In score-based structure learning we want to optimise a scoring function over different network configurations [Friedman *et al.* 1999]. That is, we use a structure score to search for the most suitable structure relative to the data. Most popular structure scores are variations on the likelihood score which calculates the probability of the data given a potential structure [Liu *et al.* 1996].

In observable data, the decomposability of the likelihood score [Carvalho *et al.* 2011], which is the ability to represent the score as a sum of variable family scores, allows for efficient learning procedures and significant computational saving [Koller and Friedman 2009]. Figure 3 shows an example of how score-based structure learning can be used to select a network which makes the data of X_1, X_2, X_3, and X_4 as likely as possible. The likelihood to the data is given by the score beneath each structure. The structure with the highest score is selected since it has the highest likelihood relative to the training data.

In incomplete data, the likelihood score is not decomposable and we have to perform inference to evaluate it [Tanner and Wong 1987]. This forces us to use non-linear optimisation techniques, such as expectation maximization or gradient ascent [Binder *et al.* 1997], which are methods used to optimise the likelihood function with multiple optima [Dempster *et al.* 1977]. Furthermore, local changes to the network can affect other parts of the network, which makes learning with incomplete data all the more difficult [Jordan 1998]. Score-based structure learning provides a way to select an optimal network for observable data but still leaves learning the latent aspects of the network open.

In this thesis we propose a score-based structure learning approach as being suited to the task of learning influence between partially observable processes since it can (a)

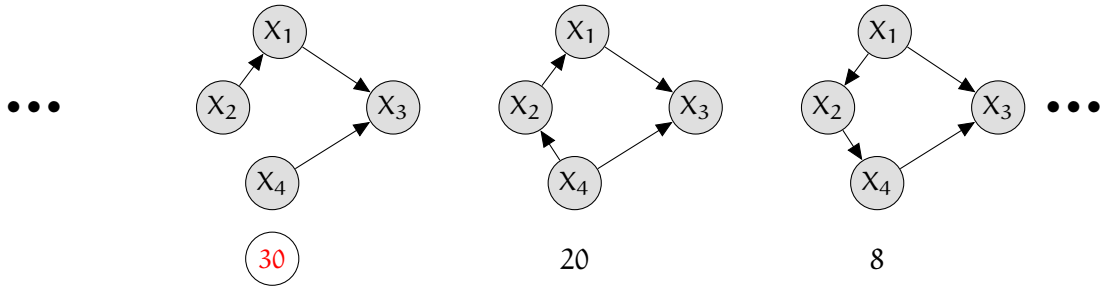


Figure 3: An example of how score-based structure learning selects the best network structure with respect to the data. The figure shows three structures each with a structure score below it. The shaded nodes represent observable variables. The highest structure score is selected (circled in red) and the corresponding structure is used.

consider the complete influence structure between processes as a state in the search space; (b) preserve basic score properties allowing for feasible computations; and (c) provide a clear indication of the independence assertions between temporal models relative to the data. A complete literature review is provided in Chapter 3. In the next section we discuss on overview of the proposed method to recover influence between processes.

1.4 OVERVIEW OF METHOD

We note the major difficulty of this problem lies in the representation of the latent components of the influence network. In this thesis we develop an algorithm which learns the probability distribution that describes interactions between processes which manifest in the temporal observations that describe each process.

The high-level architecture of the proposed algorithm is given by Figure 4. The general algorithm is as follows. We (i) input the processes; (ii) learn each stochastic process independently as a temporal model; (iii) relearn the parameters for an influence network (with the new independence assertions) by maximising the likelihood function for the hidden variables between the temporal models which represent each stochastic process; (iv) compute the structure score of the dynamic influence network; (v) check whether the condition to terminate the algorithm is met, either by convergence, some threshold, or if there is no way to improve the dynamic influence network relative to the data; (vi) slightly change the influence structure which encodes the distribution with the best possible change and continue with steps (iii), (iv), and (v). In (vii) we select the best candidate dynamic influence network.

Our method extends concepts in score-based structure learning for the domain of tracking influence between stochastic processes. We first factorise the distributions presented as stochastic processes into a set of temporal models. We then define an *assemble* and a *scoring function* to evaluate the quality of candidate influence networks (Chapter 5). At this point we have a combinatorial optimisation problem which requires us to traverse the search space for the optimal influence network, which we return as the goal structure that best fits the training data (Chapter 6). In the next section we discuss the novelty and contribution of the proposed method.

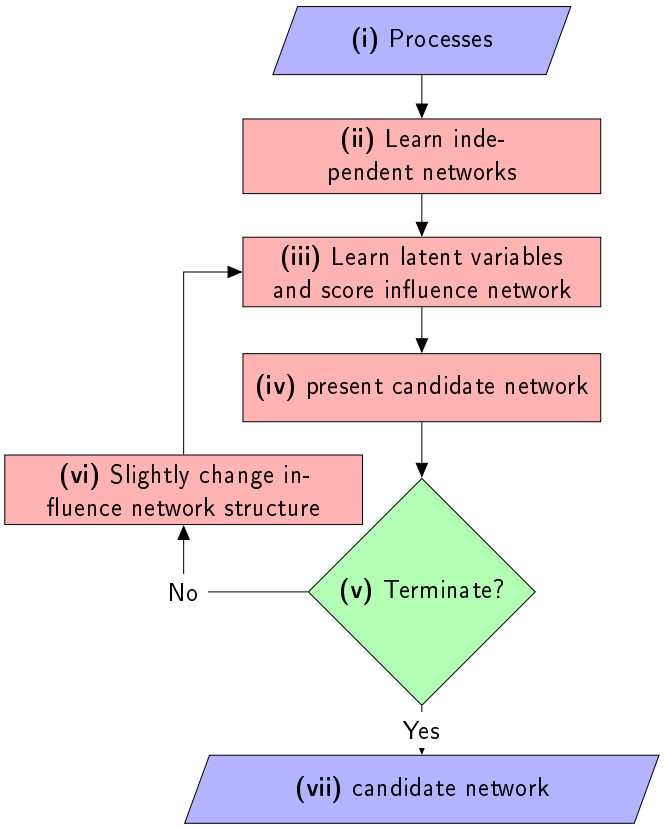


Figure 4: An overview of the proposed algorithm to recover influence between stochastic processes represented as temporal networks.

1.5 NOVELTY AND CONTRIBUTION

The novelty and intuition of our approach is to learn the influence structure in layers. We firstly learn a set of independent temporal models, and thereafter, optimise a structure score over possible structural influence configurations. Since the search for the optimal structure is done using complete data we can take advantage of efficient learning procedures and significant computational saving from the structure learning literature [Koller and Friedman 2009].

We provide the following high-level contributions:

1. The notion of influence between processes. This includes the formulations of direct and delayed influence (Chapter 4).

2. Several scoring function for dynamic influence networks by extending and adapting traditional scores for random variables along with their key properties (Chapter 5).

3. The notion of a structural assemble to relate temporal models for dynamic influence tasks (Chapter 5).

4. A learning procedure to recover the influence structure between temporal models with latent variables. We further extend the local search procedures to use assembles that link temporal models meaningfully while preserving decomposability and score-equivalence required for a manageable search (Chapter 6).

5. We provide empirical evidence for the effectiveness of our method with respect to a generative ground-truth distribution (Chapter 7).

1.6 THESIS STRUCTURE

This thesis is structured as follows. Chapter 2 provides the background work on Bayesian networks and parameter estimation necessary to understand this thesis; Chapter 3 provides relevant related work which served as the foundation for most of the algorithms presented in this thesis. Chapter 4 defines influence between processes and how to represent it using dynamic influence networks. Chapter 5 provides the structure scores necessary to evaluate the worth of dynamic influence networks and introduces the notion of a structural assemble to relate temporal models. Chapter 6 provides learning and evaluation procedures that traverse the search space and evaluate the learned model with respect to the ground truth generative model. Chapter 7 provides the results and discussion of this research. Finally, Chapter 8 presents concluding remarks and future directions.

2

BACKGROUND

2.1 INTRODUCTION

Our primary objective, in this thesis, is to track influence between stochastic processes. Stochastic processes are trajectories of complex statistical dependence that evolve over time and space [Doob 1953; Karlin 2014]. Common examples of stochastic processes include the exchange rate fluctuations [Bates 1996], the stock market [Gardiner 1985], temperature [Van Kampen 1992], speech signals [Rabiner and Juang 1986], audio signals [Kim 2000], video signals [Moore and Essa 2002], etc. Figure 5 shows an example of the monthly price of brent spot crude oil as a stochastic process [Cong *et al.* 2008].

Figure 5: Monthly price of brent spot crude oil, January 1995 to February 2016. Adapted from Independent Statistics & Analysis, U.S. Energy Information Administration (EIA) (Petroleum & Other Liquids) [Cong *et al.* 2008].

Stochastic processes can be modelled by relating random variables to each other over adjacent time-steps in dynamic Bayesian networks [Press 1989]. Dynamic Bayesian networks can expressively represent probabilistic dependencies between random variables over time, however, they present their own learning and representation problems [Schweppe 1973]. We explore the learning and representation of dynamic Bayesian networks in this chapter.

This chapter reviews the ground-work required to understand the content presented in this thesis. All of the concepts outlined here are traditional probabilistic

graphical frameworks that have been developed and refined over many years [Murphy and Russell 2002; Bishop 2006; Koller and Friedman 2009]. The intention of this chapter is to simply provide the reader with some context necessary to understand the developments of procedures presented in this thesis and are not intended to provide a tour of the practices of probabilistic graphical models.

Additional readings for understanding the representation, inference, and reasoning of probabilistic graphical models can be found in Koller and Friedman [2009]; Murphy and Russell [2002]; Pearl [1988]. In Section 2.2, we explore the representation of several Bayesian networks; and in Section 2.3, we present the fundamental practices to learning Bayesian networks.

2.2 BAYESIAN NETWORKS: REPRESENTATION

Suppose that we are given a *stochastic process* that is parameterised by a set of random variables: $\mathcal{X} = \{X_1, \ldots, X_n\}$. Our goal is to represent a joint distribution P over $\mathcal{X}$. Representing this joint distribution is both technically and computationally demanding [Bailey 1990; Sakamoto and Ghanem 2002]. This is because controlling a distribution parameterised over such an extensive space would take a large amount of computer memory and would require a super-exponential amount of prior information elicited by an expert.

The exploration of independence properties and alternative parameterisation has allowed us to express sparse compact representations that we use to explore complex joint distributions. We begin with a useful tool, called Bayesian networks, that uses independence assertions to compactly express a joint distribution [Pearl 2011].

Section 2.2.1 establishes the representation of a Bayesian network along with suitable applications and groupings; and Section 2.2.2 expands the Bayesian network representation into a template class of models that can describe complex statistical relationships over time.

2.2.1 *Bayesian Networks*

Representing probability distributions by graphical models has been explored before [Wright 1921 1934]. During these early innovations the joint probability distribution was described as influence between random variables encoded as a directed acyclic graphical structure which represented independence assertions [Smith 1989; Howard and Matheson 1984].

These developments gave raise to the notion of a Bayesian network which embeds independence assertions into a graphical structure together with a probability distribution. Pearl (and his colleagues) proposed the Bayesian network [Verma and Pearl 2013; Geiger and Pearl 2013; Geiger *et al.* 2013 1990]. We ascribe this preliminary work on Bayesian networks to the work laid out by Pearl [1988].

Section 2.2.1.1 begins by defining the Bayesian network and discuses some reasoning patterns performed on Bayesian networks; Section 2.2.1.2 introduces the notion

of I-maps and I-equivalence; and finally, Section 2.2.1.3 presents the simplest version of a Bayesian network called a naïve Bayes model, showcases various applications of Bayesian networks, and presents the problem of eliciting the structure of a Bayesian network manually by an expert.

2.2.1.1 *What is a Bayesian Network?*

A Bayesian network is a directed acyclic graph (DAG) whose nodes represent random variables and whose edges represent the influence of one variable over another [Pearl 2011]. A Bayesian network structure is often established as a set of conditional independence assertions between these random variables that encode a joint distribution in a compact way [Pearl 1988; Koller and Friedman 2009]. Two random variables X and Y are said to be conditionally independent given a set Z, denoted $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$, if once we know $\mathbf{Z}$, then knowing X does not give us any information about Y. We will use this notion of conditional independence to define the Bayesian network structure.

Definition 2.1. A Bayesian network structure, $\mathcal{G}^{\mathcal{B}}$, is a DAG whose nodes represent random variables $X_1, \ldots, X_n$. Let $\mathrm{Pa}_{X_i}^{\mathcal{G}^{\mathcal{B}}}$ denote the parents of X_i in $\mathcal{G}^{\mathcal{B}}$, and $\mathrm{NonDescendants}_{X_i}$ denote the variables in the graph that are not descendants of X_i in $\mathcal{G}^{\mathcal{B}}$. Then $\mathcal{G}^{\mathcal{B}}$ encodes the following set of conditional independence assumptions, called local independencies, denoted by $\mathcal{I}_l(\mathcal{G}^{\mathcal{B}})$:

$$\forall X_i : (X_i \perp\!\!\!\perp \mathrm{NonDescendants}_{X_i} | \mathrm{Pa}_{X_i}^{\mathcal{G}^{\mathcal{B}}}),$$

where $\perp\!\!\!\perp$ denotes independence.

Figure 6 provides an example of a Bayesian network structure. The graph in Figure 6 is a DAG with six random variables represented as nodes. Four of these are directly observed (observable, shaded) and two of these are not directly observed (latent, non-shaded). The influence between variables are represented as edges between them and are encoded as conditional independence assumptions.

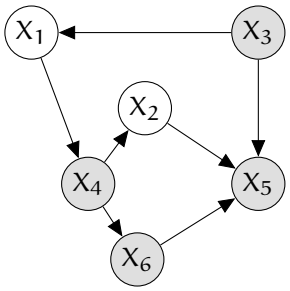


Figure 6: A Bayesian network with six random variables. Variables X_1 and X_2 are latent (not directly observed) and variables X_3, X_4, X_5, and X_6 are observed. The shaded nodes are called observable variables and non-shaded nodes are called latent variables.

Entries in the joint distribution in Bayesian networks can be expressed as a product of factors [Jensen 1996]. A factor encodes the conditional probability distribution (CPD) of a random variable and is constructed with respect to its parent variable(s)

[Pepe and Mori 1993]. Figure 7 provides an example of a Bayesian network which is used to predict the probability of the grass being wet given a set of variables [Pearl 2014]. Figure 7 explicitly presents each factor associated to each random variable. We note that each factor presented has an associated dependency with respect to the network structure [Pearl 2014]. The variable for wet grass is dependent on the variables Rain and Sprinkler. Each factor is represented by a Bernoulli distribution [Hilbe 2011] with values true (T) or false (F). We can condition on certain columns of the factor for Wet grass to give us more information about the distribution described.

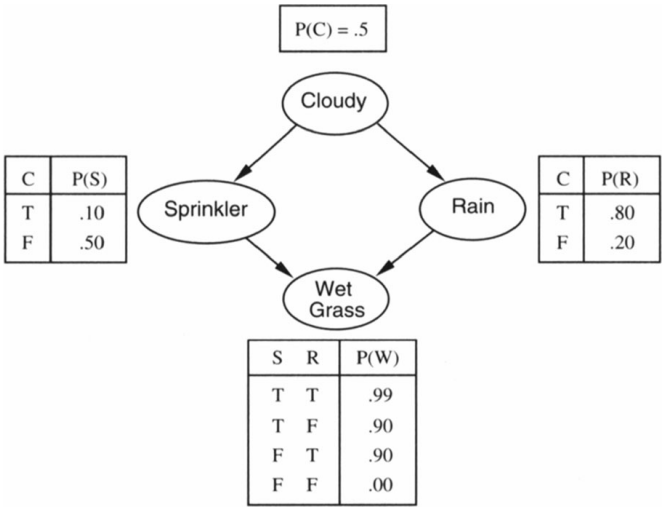


Figure 7: A Bayesian network showing the relationship between four random variables adapted from Pearl [2014]. Each node in the graph is a variable which is expressed by a table called a factor. The factor for a variable contains all the possible values for that variable assigned to a probability, this is called a parameter. The product of all factors must be a legal probability distribution.

The ability to represent the joint distribution in terms of a factorisation in Bayesian networks, as shown in Figure 7, is a important contribution [Charniak 1991]. We present the following factorisation known as the chain rule for Bayesian networks [Pearl 1988].

Definition 2.2. Let $\mathcal{G}^{\mathcal{B}}$ be a Bayesian network structure over the variables $X_1, \ldots, X_n$. We say that a probability distribution $P_{\mathcal{B}}$ over the same space factorises according to $\mathcal{G}^{\mathcal{B}}$ if $P_{\mathcal{B}}$ can be expressed as a product

$$P_{\mathcal{B}}(X_1, \ldots, X_n) = \prod_{i=1}^{n} P(X_i | Pa_{X_i}^{\mathcal{G}^{\mathcal{B}}}).$$

A Bayesian network is then just the conditional independences described by the Bayesian network structure and the distribution which is described by each variable's factor [Pearl 1988].

Definition 2.3. A Bayesian network is a pair $\mathcal{B} = (\mathcal{G}^{\mathcal{B}}, P_{\mathcal{B}})$ where the distribution $P_{\mathcal{B}}$ factorises over the independence assumptions in $\mathcal{G}^{\mathcal{B}}$.

Having defined a Bayesian network, we might consider performing reasoning or inference [Jensen 1996; Zou and Feng 2009; Cooper 1990]. Wellman [1990] crystallised

the three fundamental types of reasoning patterns in Bayesian networks: causal reasoning, evidential reasoning, and inter-causal reasoning. These patterns can be used to promote knowledge elicitation and learning [Renooij and van der Gaag 2002; Hartemink *et al.* 2002] as well as the development of intuition to inference tasks [Druzdzel 1993]. The importance of reasoning patterns are emphasised by Pearl [1988]. In the next section we will see how two Bayesian network structures can decompose as the same set of independence assumptions.

2.2.1.2 *I-maps and I-equivalence*

Recall that our Bayesian network structure, $\mathcal{G}^{\mathcal{B}}$, encodes a set of independence assertions, $\mathcal{I}(\mathcal{G}^{\mathcal{B}})$. Let us define these assertions more carefully:

Definition 2.4. Let P be a distribution over a set of random variables $\mathcal{X}$. The set of independence assertions, denoted $\mathcal{I}(P)$, is a set of statements of the form $(X \perp\!\!\!\perp Y \mid \mathbf{Z})$. These statements all hold in P.

$\mathcal{G}^{\mathcal{B}}$ is said to be an I-map of P since P satisfies the local independence assertions associated with the Bayesian network structure $\mathcal{G}^{\mathcal{B}}$. We denote this as $\mathcal{I}(\mathcal{G}^{\mathcal{B}}) \subset \mathcal{I}(P)$. More generally,

Definition 2.5. Let $\mathcal{G}$ be a graph structure and $\mathcal{I}(\mathcal{G})$ be its set of independence assertions. $\mathcal{G}$ is said to be an I-map for a set of independences $\mathcal{I}$ if $\mathcal{I}(\mathcal{G}) \subseteq \mathcal{I}$.

Intuitively we view an I-map, $\mathcal{G}$, as an incomplete indication about the set of independence assumptions that must hold in P [Koski and Noble 2011]. However, P may contain additional independence statements that do not hold in $\mathcal{G}$.

An important observation is that $\mathcal{I}(\mathcal{G}^{\mathcal{B}})$ provides an abstraction from the specific graphical structure to a set of independence assertions. This set stands to represent $\mathcal{G}^{\mathcal{B}}$ as a specification of independence statements, which suggests that although two graphical structures, $\mathcal{G}_1^{\mathcal{B}}$ and $\mathcal{G}_2^{\mathcal{B}}$, may be semantically different, they could encode exactly the same set of conditionally independence assumptions, that is $\mathcal{I}(\mathcal{G}_1^{\mathcal{B}}) = \mathcal{I}(\mathcal{G}_2^{\mathcal{B}})$ [Koller and Friedman 2009]. More formally,

Definition 2.6. Two Bayesian network structures, $\mathcal{G}_1^{\mathcal{B}}$ and $\mathcal{G}_2^{\mathcal{B}}$, over the same set of random variables $\mathcal{X}$ are said to be I-equivalent if $\mathcal{I}(\mathcal{G}_1^{\mathcal{B}}) = \mathcal{I}(\mathcal{G}_2^{\mathcal{B}})$.

It can be further stated that every possible configuration over the random variables $\mathcal{X}$ can be partitioned into sets of mutually exclusive I-equivalence classes as defined in Definition 2.6. Figure 8 shows an example of three graphical models that encode exactly the same independence assumption: $(A \perp\!\!\!\perp C \mid B)$. Notice that although the independence assumption is the same in all three networks, the edges can be oriented in different ways.

I-equivalence was defined by Verma and Pearl [1992]; Judea Pearl [1991]. The contribution by Chickering [1995]; Verma and Pearl [1992]; Judea Pearl [1991] provide powerful tools to prove properties of I-equivalent graphical structures.

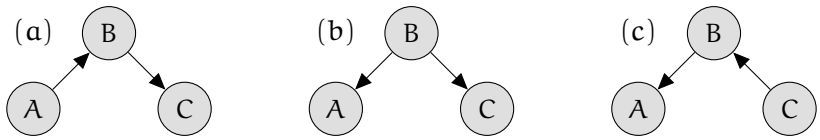


Figure 8: An illustration of three graphical models that encode the independence assumption: $(\mathbf{A} \perp\!\!\!\perp \mathbf{C} \mid \mathbf{B})$.

I-equivalence is undoubtedly an important concept when recovering influence relations since the true structure of a Bayesian network is seen as not identifiable from members of the I-equivalence class given observable data alone [Murphy 2012; Koller and Friedman 2009; Pearl 2011]. However, there are algorithms that can reconstruct the I-equivalence class given the observable distribution, such as those by Pearl and Verma [1995]; Verma and Pearl [1992]; Spirtes *et al.* [2000]; Meek [1995].

In the next section we explore the simplest kind of Bayesian network structure and probability distribution, called the naïve Bayes model.

2.2.1.3 *Naïve Bayes Model*

Perhaps the simplest example of a Bayesian network is the naïve Bayes model [McCallum and Nigam 1998] which has been used successfully by many expert systems [De Dombal *et al.* 1972; Gorry and Barnett 1968; Warner *et al.* 1961].

The naïve Bayes model predefines a finite set of mutually exclusive classes [Rish 2001]. Each instance (set of observations) can fall into one and only one of these classes, which is represented as a latent class variable. The model also poses some observed set of features $X_1, \ldots, X_n$. The assumption is that all of the features are conditionally independent given the class label of each instance [Murphy 2012]. That is,

$$\forall \mathbf{i}(\mathbf{X_i} \perp\!\!\!\perp \mathbf{X_{i'}} \mid \mathbf{C}), \text{ where } X_{i'} = \{X_1, \ldots, X_n\} - \{X_i\}.$$

Figure 9 presents the Bayesian network representation of the naïve Bayes model. The joint distribution of the naïve Bayes model factorises compactly as a prior probability of an instance belonging to a class, $P(C)$, and a set of CPDs which indicate the probability of a feature given the class [Koller and Friedman 2009], $P(X_i \mid C)$. We can state this distribution more formally as

$$P(C, X_1, \ldots, X_n) = P(C) \prod_{i=1}^{n} P(X_i|C).$$

The naïve Bayes model remains a simple, yet highly effective, compact, and high-dimensional probability distribution that is often used for classification problems [Shinde and Prasad 2017; Lewis 1998; Duda and Hart 1973]. The main limitation of the naïve Bayes model is the assumption that all features are conditionally independent given the class variable.

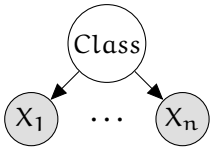


Figure 9: A graphical illustration of the naïve Bayes model.

The use of Bayesian networks span a range of applications including general diagnostic systems [Andreassen *et al.* 1987; Heckerman *et al.* 2016; Breese *et al.* 1992]; event forecasting [Abramson 1994; Gu *et al.* 1994; West 1996; Sun *et al.* 2006]; assessment of short free-text responses [Klein *et al.* 2011]; machine vision [Levitt *et al.* 2013; Binford *et al.* 2013; Buxton 1997]; manufacturing [Nadi *et al.* 1991; Wolbrecht *et al.* 2000; Weber and Jouffe 2003], and emergency evacuation [Wang *et al.* 2008] to name a few.

We have seen that the Bayesian network is an important tool for describing high-dimensional complex probability distributions and have outlined several successful applications. However, manually eliciting the conditional probability distribution (CPD) and network structure is a complex task that has plagued decision analysis for many years [Chesley 1978; Spetzler and Stael von Holstein 1975]. This difficult process is subject to many biases [Tversky and Kahneman 1975; Daneshkhah 2004] and although some contributing methods can be used to obtain the network structure and parametrisation from an expert [Schachter and Heckerman 1987], this remains a difficult processes.

In this subsection we explored the Bayesian network representation as a powerful tool to describe environments for numerous applications. In the next section we will extend this descriptive language for the temporal setting.

2.2.2 *Dynamic Bayesian Models*

Bayesian networks can be used to model a joint distribution over a set of random variables [Koski and Noble 2011]. In temporal settings, however, we wish to model distributions over trajectories (systems that change over time) [Murphy and Russell 2002].

For example, suppose we are interested in capturing the traffic condition of a road over time [Jayakrishnan *et al.* 1994]. Then we would be interested in building a distribution of features over different time-slices with respect to some time granularity to capture the distribution of traffic over time [Papageorgiou 1990].

In such cases it is often useful to construct a template model which unrolls as the trajectory evolves over time [Conati *et al.* 2002]. In this section we explore the rich and expressive language of dynamic Bayesian networks which can be used to describe distributions over trajectories [Pavlovic *et al.* 1999; Dojer *et al.* 2006].

Section 2.2.2.1 and Section 2.2.2.2 explore two common assumptions that are made when dealing with dynamic Bayesian networks; Section 2.2.2.3 introduces the language of dynamic Bayesian networks and explores some notable applications and practices of dynamic Bayesian networks; and finally, Section 2.2.2.4 presents a particular class of dynamic Bayesian networks which organises its random variables as a hierarchy.

2.2.2.1 *Markov Systems*

In this section we will consider dynamic Bayesian networks to represent systems that evolve over time. The dynamic Bayesian network is a temporal model represented by a set of template variables, denoted $\mathcal{X}$. We will denote the values of the template variables X_i at time t as $X_i^{(t)}$.

A stochastic process can be viewed as a trajectory over a set of discrete time steps specified by a time granularity. Over this time granularity, we can model the relationship between these time-slices using the chain rule for probabilities (Definition 2.2) [Koller and Friedman 2009] as,

$$P(\mathcal{X}^{(0:T)}) = P(\mathcal{X}^{(0)}) \prod_{t=0}^{T-1} P(\mathcal{X}^{(t+1)} \mid \mathcal{X}^{(0:t)}).$$

As T increases, $P(\mathcal{X}^{(0:T)})$ exponentially increases the number of independence assumptions over the trajectory length. Thus, a simplifying assumption is to model the next state as conditionally independent of the past given the present. We present the notion of a Markov system.

Definition 2.7. A dynamic system is said to be Markov, over the set of template variables $\mathcal{X}$, if for all $t \geqslant 0$,

$$(\mathcal{X}^{(t+1)} \perp\!\!\!\perp \mathcal{X}^{(0:(t-1))} \mid \mathcal{X}^{(t)}).$$

We note that use of the Markov assumption is a reasonable approximation if we use a rich description of the system state [Koller and Friedman 2009; Murphy and Russell 2002]. That is, increasing the number of variables which describe each time-slice might allow us to describe influences which persist through time.

2.2.2.2 *Time Invariance*

The Markov assumption simplifies the distribution over time, however, we need to make one more simplifying assumption about the general representation of the temporal model [Koller and Friedman 2009; Murphy and Russell 2002].

Definition 2.8. A Markov system is said to be *time invariant* if $P(\mathcal{X}^{(t+1)} \mid \mathcal{X}^{(t)})$ is the same $\forall t$. That is $\forall t \geqslant 0$, we describe the process as the transition model $P(\mathcal{X}'|\mathcal{X})$ as

$$P(\mathcal{X}^{(t+1)} = \xi' \mid \mathcal{X}^{t} = \xi) = P(\mathcal{X}' = \xi' \mid \mathcal{X} = \xi),$$

where ξ' is the next data instance and ξ is the current data instance.

With these assumptions in hand we can define a dynamic Bayesian network in the next section.

2.2.2.3 *What is a Dynamic Bayesian Network?*

The assumptions made in the previous section allow us to compactly represent a trajectory over time [Murphy and Russell 2002]. It is compact since we need to only specify a 2-time-slice Bayesian network that consists of the initial distribution and a transition model, $P(\mathcal{X}'|\mathcal{X})$ [Koller and Friedman 2009]. The transition model can then be unrolled using the Markov and time invariance assumptions into a dynamic Bayesian network. We begin by defining the 2-time-slice Bayesian network.

Definition 2.9. A 2-time-slice Bayesian network for a process over the set of template variables $\mathcal{X}$ is a Bayesian network over $\mathcal{X}'$ given $\mathcal{X}_I$, where $\mathcal{X}_I \subseteq \mathcal{X}$ is a set of interface variables.

The conditional Bayesian network described only has parents and hence CPDs for $\mathcal{X}'$. Interface variables refer to those variable that persist through the temporal aspect of the model (e.g. high-level changes in the process). Using our simplifying assumptions, the distribution defined can be described as

$$P(\mathcal{X}'|\mathcal{X}) = P(\mathcal{X}'|\mathcal{X}_I) = \prod_{i=1}^{n} P(X'_i|Pa_{X'_i}). \tag{1}$$

Consequently for every template variable we will have a template factor (Section 2.2.1.1) which will be initialised as the model unfolds. There are generally two types of edges defined in these models [Koller and Friedman 2009; Murphy and Russell 2002]:

- Inter-time-slice edges, which describe dependencies between time-slices. Inter-time-slice edges that are between copies of the same template variables are called persistent edges. We refer to variables with persistent edges as persistent variables.
- Intra-time-slice edges describe dependencies within each time-slice.

We now provide a definition of the unrolled dynamic Bayesian network which uses the notion of a 2-time-slice model.

Definition 2.10. A dynamic Bayesian network (DBN) is a pair $\langle \mathcal{B}_0, \mathcal{B}_\rightarrow \rangle$, where $\mathcal{B}_0$ is a Bayesian network over $\mathcal{X}^{(0)}$, representing the initial distribution over states and $\mathcal{B}_\rightarrow$ is a 2-time-slice Bayesian network for the process. For any desired time span $T \geqslant 0$, the distribution over $\mathcal{X}^{(0:T)}$ is defined as an unrolled Bayesian network, where, for any $i = 1, \ldots, n$:

- the structure and CPDs of $\mathcal{X}_i^{(0)}$ are the same as those for $\mathcal{X}_i$ in $\mathcal{B}_0$,
- the structure and CPDs of $\mathcal{X}_i^{(t)}$ for $t > 0$ are the same as those for $\mathcal{X}'_i$ in $\mathcal{B}_\rightarrow$.

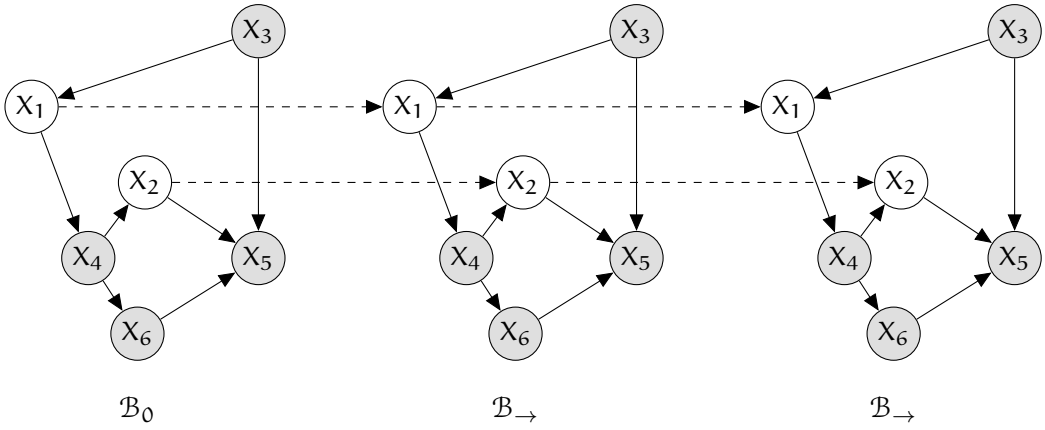


Figure 10: A dynamic Bayesian network with six template variables. Both X_1 and X_2 are persistent variables connected by dotted intra-time-slice persistent edges. The shaded template variables are observable and not a part of the interface set. The network is unrolled over three time-slices. Below each time-slice there is an indication from which model in the 2TBN it is derived from.

Figure 10 shows an example of an unrolled dynamic Bayesian network. Temporal models over trajectories have been discussed for many years, such models include hidden Markov models (HMMs) [Rabiner and Juang 1986; Rabiner 1989], Kalman filters [Kalman 1960], and possibly the first formal occurrence of dynamic Bayesian networks (DBNs) in Dean and Kanazawa [1989]. The connections between HMMs and DBNs have also been explored [Smyth *et al.* 1997]. Murphy and Russell [2002]; Koller and Friedman [2009]; Murphy [2012] provide an overview of temporal representations and DBNs.

DBNs have been used for dependability, risk analysis, and maintenance [Weber *et al.* 2012]; speech recognition [Zweig and Russell 1998]; recognising office activities [Oliver and Horvitz 2005]; vehicle classification in video [Kafai and Bhanu 2012]; the analysis of football matches [Huang *et al.* 2006]; and even in genetics, where DBNs capture temporal expression data to uncover gene interaction in cellular systems[1] [Friedman *et al.* 2000].

2.2.2.4 *Hierarchical Bayesian Networks*

DBNs are able to model trajectories of complex statistical dependences that evolve over time [Murphy and Russell 2002]. These are modelled by relating variables to each other over adjacent time-steps [Murphy and Russell 2002]. DBNs expressively represent probabilistic dependencies between variables but can be redefined to express dependencies between composite variables, which are mixtures of other variables [Peelen *et al.* 2010]. This is useful in situations when we wish to learn abstractions of a subset of observations for a process. We extend the definition of DBNs

1 This is done using DNA hybridization arrays which estimate the expression levels of thousands of genes to describe the transcription level within a cell [Friedman *et al.* 2000].

into hierarchical dynamic Bayesian networks (HDBNs) which expresses composite variables naturally in temporal models.

Hierarchical Bayesian networks (HBNs) are used extensively for reasoning under uncertainty with structured data [Gelman *et al.* 2014]. In this section we extend the HBN to one which expresses uncertainty over structured data over time. The major contribution of HBNs lies in their expressive power to aggregate random variables as composite structures of other variables [Gelman *et al.* 2014; Peelen *et al.* 2010; Murphy and Russell 2002]. We begin by defining a composite variable.

Definition 2.11. A variable, X, is said to be a composite variable if it has a component set $\{X_1, \ldots, X_n\}$ of variables.

These composite variables can be expressed recursively to construct complex hierarchical tree (h-tree) structures.

Definition 2.12. A hierarchical tree (h-tree) structure of a composite variable X, denoted h_X, is a directed tree structure rooted at X, with each of the elements in X's component set being children of X expanding along with each of the component's respective h-trees.

Gyftodimos and Flach [2002] provide an overview of hierarchical models using composite types. Figure 11 shows an example of a simple hierarchical structure for 11 composite variables. Each composite variable, alongside its factor that describes a distribution, also contains a set of composite variables which are dependent on it given the hierarchical structure. Leaf nodes have empty composite variable sets since they are at the lowest level of the hierarchy. These are usually observable. We now formally define a HBN structure using this notion of a h-tree.

Definition 2.13. A hierarchical Bayesian network structure of a composite variable X with corresponding h-tree structure is a set $\mathcal{H}_X = \{h_{X_1}, \ldots, h_{X_n}\}$, where h_{X_i} is the corresponding h-tree for the i^{th} component of X.

A hierarchical Bayesian network is simply a distribution which factorises over this hierarchical structure. More formally,

Definition 2.14. A hierarchical Bayesian network is pair $\mathcal{H} = (\mathcal{H}_X, P_{\mathcal{H}_X})$ where $P_{\mathcal{H}}$ factorises over $\mathcal{H}_{\mathcal{X}}$.

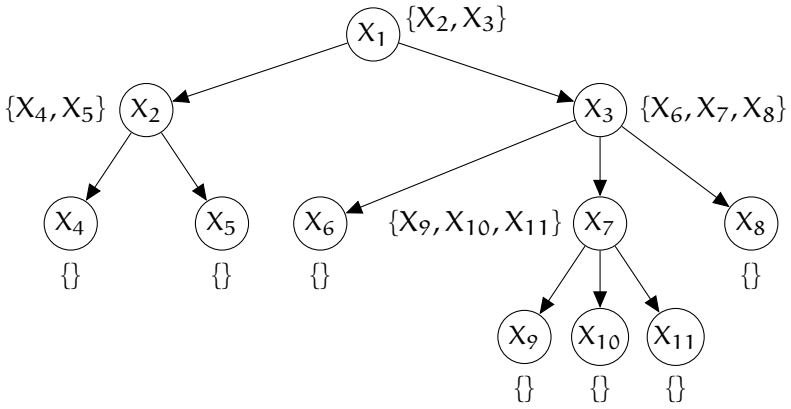


Figure 11: A hierarchical Bayesian network structure for 11 random variables. Each variable is a data-structure made up of a factor and a set of composite variables. The factor describes the conditional probability distributions (CPDs) and the set of composite variables contain the child variables relative to the structure.

An inherited property of hierarchical Bayesian network structure from standard Bayesian network structures is outlined in Definition 2.1. We now naturally extend this definition for hierarchical dynamic Bayesian networks (HDBNs) for structured time-series data.

Definition 2.15. A 2-time-slice hierarchical Bayesian network (2-THBN) for a process over a set of composite variables $\mathcal{X}$ is a hierarchical Bayesian network over $\mathcal{X}$ given $\mathcal{X}_I$, where $\mathcal{X}_I \subseteq \mathcal{X}$ is a set of interface variables.

We can now adopt the Markov and time invariance simplifying assumptions to define the hierarchical dynamic Bayesian network (HDBN).

Definition 2.16. A hierarchical dynamic Bayesian network (HDBN) is a pair $\mathcal{H}_{DB} = \langle \mathcal{H}_0, \mathcal{H}_{\rightarrow} \rangle$, where $\mathcal{H}_0$ is a hierarchical Bayesian network over $\mathcal{X}^{(0)}$, representing the initial distribution over states and $\mathcal{H}_{\rightarrow}$ is a 2-THBN for the process. For any desired time span $T \geqslant 0$, the distribution over $\mathcal{X}^{(0:T)}$ is defined as a unrolled hierarchical Bayesian network, where, for any $i = 1, \ldots, n$:

- the structure and CPDs of $\mathcal{X}_i^{(0)}$ are the same as those for $\mathcal{X}_i$ in $\mathcal{H}_0$,
- the structure and CPDs of $\mathcal{X}_i^{(t)}$ for $t > 0$ are the same as those for $\mathcal{X}_i'$ in $\mathcal{H}_{\rightarrow}$.

Figure 12 illustrates an example of a HDBN with 3 time-slices and 7 template variables. Some inter-time-slice edges are persistent and some persist to other variables. The main strength of HDBNs is its ability to represent uncertainty in structured data by aggregating variables in high-level features, this allows us to describe a rich probability distribution through the abstraction of observations even in the presence of incomplete data.

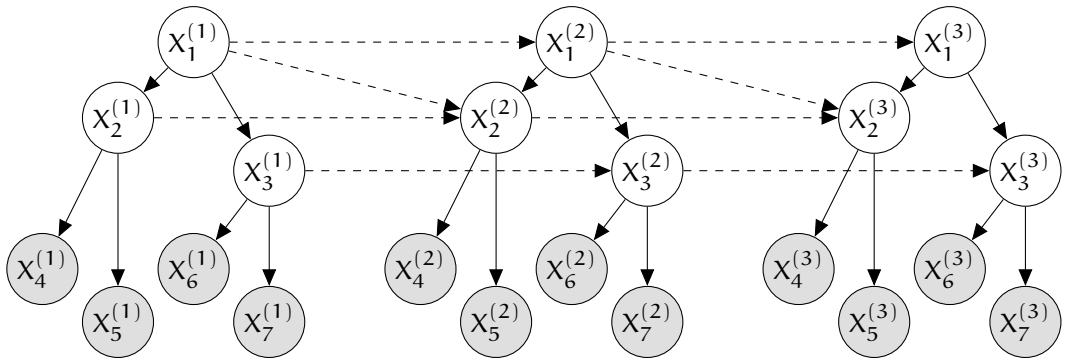


Figure 12: A hierarchical dynamic Bayesian network with three time-slices and seven variables per time-slice (3 latent and 4 observable). The inter-time-slice edges (dashed lines) are sometimes non-persistent which can enrich the distribution. The intra-time-slice edges (solid lines) describe the hierarchical structure which is captured in these models.

2.3 BAYESIAN NETWORKS: LEARNING

In this section we explore the task of learning Bayesian networks from data. Figure 13 shows the context of the learning process. Suppose we observe (or partially observe) a *true distribution* (P^*) as shown in Figure 13. We may also assume that the true distribution, P^*, is generated from a *true network structure* ($\mathcal{G}^*$). That is, a set of independence assumptions from which the true distribution is an I-map. From the true distribution we are able to obtain a set of samples. We might also have access to domain expertise which together with the samples generated will allow us to learn a *network*. Domain knowledge can come in the form of information about the structure between variables or possibly the general structure of a distribution of the involved variables. Hastie *et al.* [2001]; Bishop [2007] provide an overview of basic learning problems and algorithms in Bayesian networks.

The motivation behind learning Bayesian networks is primarily for density estimation [John and Langley 1995; Fraley and Raftery 2002], to make predictions over new instances, and as a framework for knowledge discovery [Fayyad *et al.* 1996; Heckerman 1996], that is, learning how variables may interact.

There are four different contexts of learning in Bayesian networks [Koller and Friedman 2009]: we may have (a) a known structure with complete data; (b) unknown structure with complete data; (c) known structure with incomplete data; or (d) unknown structure with incomplete data. We begin by discussing the learning problems (a) and (c) in Section 2.3.1, and leave structure learning problems (b) and (d) to be discussed with the related work in the next chapter.

2.3.1 *Parameter Estimation*

In this section we explore various parameter estimation learning algorithms. We begin be discussing maximum likelihood estimation (MLE) [Scholz 1985; Johansen and

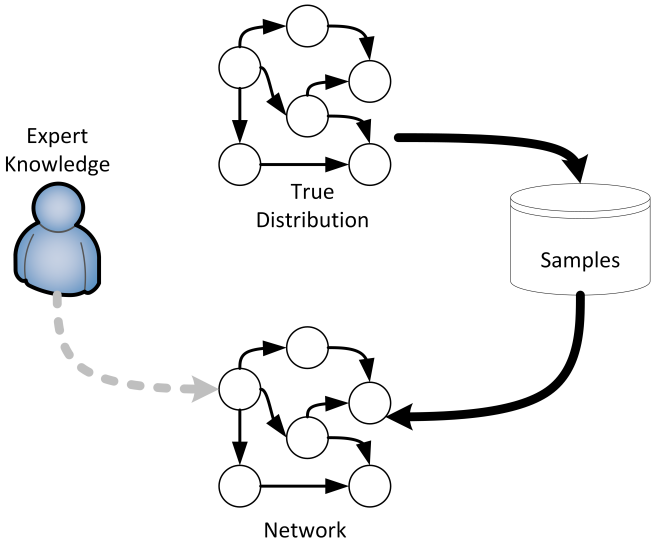


Figure 13: An illustration of the process of learning a Bayesian network from domain expertise and samples that are generated from a true distribution. Adapted from Koller and Friedman [2009].

Juselius 1990] in Section 2.3.1.1, which optimises the likelihood function for complete data. MLE is perhaps the most commonly used parameter estimation tool available [Lehn 2017; Scholz 1985; Enders and Bandalos 2001]. We then explore Bayesian estimation in Section 2.3.1.2, which is based on the Bayesian paradigm which states that anything we have uncertainty over should be expressed as a distribution [Shafer 1976].

We also address the problem of learning the parameters of DBNs which are used to express a distribution over trajectories [Koller and Friedman 2009; Murphy and Russell 2002]. We will see that temporal models are extensions of Bayesian learning due to the simplifying assumptions (Section 2.2.2.1 and 2.2.2.2). Finally, in Section 2.3.1.3, we address learning with incomplete data, which is a much harder problem.

2.3.1.1 *Maximum Likelihood Estimation*

Perhaps the simplest parameter learning problem is maximum likelihood estimation (MLE) from a set of observations [Scholz 1985]. MLE is foundational to many parameter learning problems and much work has been dedicated to its development [DeGroot and Schervish 2012; Schervish 2012; Hastie *et al.* 2001; Bishop 2007; Bernardo and Smith 2001]. Howard [1970] provides a tutorial on maximum likelihood estimation.

Suppose we are given a set of observations in the form of a dataset $\mathcal{D} = \{\xi_1, \ldots, \xi_M\}$ sampled independently and identically distributed (IID) from P^*. That is, the instances are independent of each other and sampled from P^* [Hoadley 1971; Gänssler and Stute 1979]. Our goal is to find the set of parameters, Θ, that predicts $\mathcal{D}$. In order to address this we often look at the likelihood of the parameters with respect to the data [Scholz 1985; Hastie *et al.* 2001; Bernardo and Smith 2001]. For example, suppose

that our data consists of a single observation, x, per instance, ξ, then the likelihood of the parameters relative to the data is given by

$$L(\theta : \mathcal{D}) = P(\mathcal{D}|\theta) = \prod_{m=1}^{M} P(x[m]|\theta). \tag{2}$$

The likelihood which is assigned to a particular parameter can be calculated by using the notion of a sufficient statistic [Koller and Friedman 2009].

Definition 2.17. A function $s(\mathcal{D})$ is a sufficient statistic from instances to a vector in $\mathbb{R}^{\mathcal{K}}$, where $\mathcal{K}$ is the total number of values for X, if for any two datasets $\mathcal{D}$ and $\mathcal{D}'$ and any $\theta \in \Theta$ we have

$$\sum_{x[i]\in\mathcal{D}} s(x[i]) = \sum_{x[i]\in\mathcal{D}'} s(x[i]) \implies L(\theta : \mathcal{D}) = L(\theta : \mathcal{D}').$$

In other words for any two datasets $\mathcal{D}$ and $\mathcal{D}'$, and any parameter θ, we have that if the sum of all of the sufficient statistics over all the instances in both datasets are the same, then their likelihood functions are the same [Koller and Friedman 2009]. We can express the likelihood of the parameters relative to the data as

$$L(\theta : \mathcal{D}) = \prod_{i=1}^{k} \theta_i^{M_i}, \tag{3}$$

where θ_i is the parameter for $x = x[i]$ and M_i is the sufficient statistic of observation $x[i]$. This distribution in the above equation is for a multinomial distribution. To choose the MLE for the parameter, θ, we simply find the optimum, $\hat{\theta}$, which yields

$$\hat{\theta}^i = \frac{M_i}{\sum_{i=1}^{m} M_i}. \tag{4}$$

Note that Equation 4 is just the fraction of the value x_i in the data represented by its respective sufficient statistic.

MLE is considered a simple way to select a parameter that predicts $\mathcal{D}$. The parameter is constructed by using sufficient statistics which represent the key statistical properties of the dataset $\mathcal{D}$ with computational efficiency and provides a closed form solution. Heckerman [1998]; Buntine [1996 1994] provide several tutorials on this foundational concept.

Although learning the parameters of independent events could be easily done using MLE, Bayesian networks seem to present a significantly more difficult parameter learning problem. In Bayesian network parameter learning we are interested in modelling complex conditional distributions. As it turns out learning parameters for a Bayesian network is not significantly harder than learning the parameters for independent variables since the likelihood function is decomposable with respect to the network's independence assumptions.

More specifically, Equation 2 presents the likelihood over a single variable. Suppose we extend this formula for the likelihood of all variables in a network given its particular structure. We can express the likelihood of all of the variables, given $\mathcal{D}$ using the chain rule for Bayesian networks as in Definition 2.2,

$$L(\Theta : \mathcal{D}) = \prod_m \prod_i P(x_i[m]|\mathbf{U}_i[m] : \Theta_i),$$

where $\mathbf{U}_i$ is the set of parent values for x_i with respect to the structure (i indicates the variable number and m indicates the instance number). We can now switch the order of the products from a product over variables to a product over data instances [Koller and Friedman 2009],

$$L(\Theta : \mathcal{D}) = \prod_i \prod_m P(x_i[m]|\mathbf{U}_i[m] : \Theta_i).$$

Therefore the likelihood of the parameters relative to the data is

$$L(\Theta : \mathcal{D}) = \prod_i L_i(\mathcal{D} : \Theta_i),$$

which yields the likelihood of each family of variables individually. The decomposability of the likelihood function can be further exploited by using table CPDs (which is omitted here. For further reading on the decomposability of the likelihood term for Bayesian networks please consult Murphy and Russell [2002]; Bishop [2006]; Koller and Friedman [2009]).

We can also use MLE when dealing with temporal models. Suppose we are given the following Markov chain [Kemeny and Snell 1960; Gilks *et al.* 1995] as shown in Figure 14.

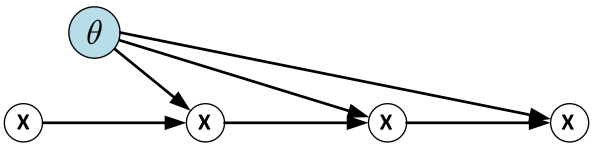


Figure 14: A Markov chain over four time-slices. The parameter for each variable is explicitly shown by the blue node. Adapted from Conati [2002].

Given the time invariance assumption (Definition 2.8), we can describe the temporal process as a transition model $P(\mathcal{X}'|\mathcal{X})$. Thus we can describe the likelihood function as

$$L(\theta : X^{0:T}) = \prod_{t=1}^{T} P(X^{(t)}|X^{(t-1)} : \theta).$$

If we consider how the temporal model decomposes over pairs of states X^i and X^j, then we can reformulate the product as

$$L(\theta : X^{0:T}) = \prod_{i,j} \prod_{t:X^{(t)}=X^i, X^{(t+1)}=X^j} P(X^{(t+1)}|X^{(t)} : \theta_{X'|X}).$$

In this case we are considering the probability over each transition X^i to X^j. Given the time invariance assumption we note that the parameters for the model are the same regardless of which time point we consider. Therefore, can rewrite the likelihood function as a product over pairs of time-slices given the particular sufficient statistic. That is, for X a multinomial random variable and $\theta \in \mathbb{R}$,

$$L(\Theta : X^{0:T}) = \prod_{i,j} \prod_{t:X^{(t)}=X^i, X^{(t+1)}=X^j} \theta_{X^i \to X^j} = \prod_{i,j} \theta_{X^i \to X^j}^{M[X^i \to X^j]}.$$

Similarly, we can extend this to the context of HMMs [Eddy 1996], such as the one in Figure 15. The likelihood function for the HMM decomposes as

$$L(\Theta : X^{0:T}, O^{0:T}) = \prod_{i,j} \theta_{X^i \to X^j}^{M[X^i \to X^j]} \prod_{i,k} \theta_{O^k|X^i}^{M[X^i, O^k]}, \tag{5}$$

where X and O are multinomial random variables, $\theta \in \mathbb{R}$, with the additional parameters which correspond to the observation k in the state i exponentiated by the number of times we observe both X^i and O^k [Blunsom 2004].

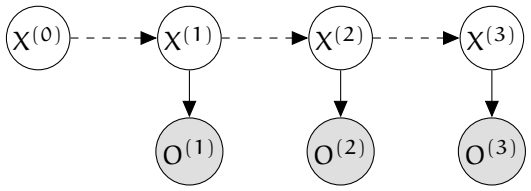


Figure 15: An illustration of a hidden Markov model with 4 time-slices. The dotted lines indicate the inter-time-slice edges for the persistent variable $X^{(t)}$. The solid line indicate the intra-time-slice edges for each respective time-slice.

To summarise, MLE provides a mechanism to select a parameter which makes the data as probable as possible. The convenience lies in its decomposability into local likelihood functions per observable family variable.

When optimising the likelihood function one must be careful of fragmentation, which states that as the number of parents increase, the number of possible assignments for each variable increases exponentially. This concept leads to poor parameter estimates in the severe cases. In such cases we might be able to perform better at density estimation by considering simpler structures even if they may be incorrect [Koller and Friedman 2009; Murphy and Russell 2002; Bishop 2006]. In the next section we explore an extension to MLE which provides a Bayesian alternative.

2.3.1.2 *Bayesian Learning*

An alternative to MLE for learning the parameters of variables in a Bayesian network is *Bayesian estimation* [John and Langley 1995; Heckerman 1998]. Bayesian estimation follows the Bayesian paradigm which views any event which has uncertainty as a random variable with a distribution over it [Shafer 1976].

The naïve Bayes model, illustrated in Figure 9, for classification is an early application for Bayesian estimation [Duda *et al.* 1979]. Unlike in MLE, where we attempted

to estimate the most likely parameter θ relative to the data, in Bayesian estimation we view the parameter θ as a continuous random variable ($\theta \in [0, 1]$). Each instance, ξ, is viewed as conditionally independent given θ. However, since θ is not known (what we are trying to learn), the instances are not marginally independent and information about every instance should tell us something about θ. Figure 16 illustrates how the dataset is dependent on the parameter θ which updates its values as more data is acquired over time.

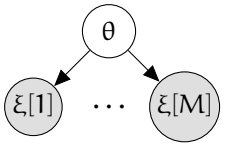


Figure 16: A graphical depiction of Bayesian estimation which views the parameter θ as a random variable. All the data instances are dependent on θ.

As an example consider Figure 17 which shows a simple Bayesian network with two random variables G and R. Here we see that the parameter values θ_G and $\theta_{C|G}$ are expressed explicitly as random variables in the model. Buntine [1994]; Gilks *et al.* [1994] were the first to introduce Bayesian estimation for temporal models in terms of the template model representations.

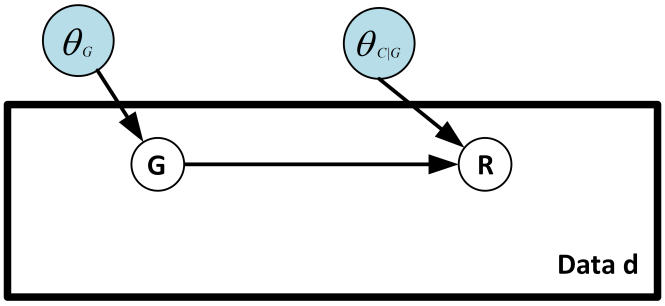


Figure 17: An example of a Bayesian network with variable parameters explicitly indicated as continuous random variables (blue nodes). The observable variables are indicated by the white nodes and the blue nodes explicitly indicate the parameters. Adapted from Wang *et al.* [2008].

Suppose each instance ξ contains only one observation x, then we can express the joint distribution of each observation by using the chain rule for Bayesian networks (Definition 2.2) and the dependencies specified in Figure 16 as

$$P(x[1], \ldots, x[M], \theta) = P(x[1], \ldots, x[M]|\theta)P(\theta).$$

Which gives us the prior over θ and the probability of each instance given θ,

$$P(x[1], \ldots, x[M], \theta) = P(\theta) \prod_{i=1}^{M} P(x[i]|\theta).$$

We note some similarities to MLE in Equation 2 with the addition of the prior probability over θ. The prior distribution allows us to express the posterior of this prior given the data using Bayes rule:

$$P(\theta|x[1],\ldots,x[M]) = \frac{P(x[1],\ldots,x[M]|\theta)P(\theta)}{P(x[1],\ldots,x[M])}.$$

There are many choices for a prior distribution. One common choice is the Dirichlet prior which is characterised by a set of hyper-parameters $(\alpha_1,\ldots,\alpha_k)$. The Dirichlet prior for Bayesian networks was defined by Heckerman *et al.* [1995a]. The Dirichlet probability distribution is described as a density function over θ which has the form

$$P(\theta) = \frac{1}{Z}\prod_{i=1}^{k}\theta_i^{\alpha_i-1}, \text{ where } Z = \frac{\prod_{i=1}^{k}\Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{k}\alpha_i)}, \ \Gamma(x) = \int_0^\infty t^{x-1}e^t dt. \tag{6}$$

Figure 18 shows several examples of a special case of the Dirichlet distribution with two hyper-parameters α and β (also called a beta distribution). We note that as we increase the hyper-parameter values while $\alpha = \beta$, we get a peak at the center of the x-axis. This corresponds to a stronger belief that the parameters are positioned around the center (e.g. when α and β are both 2).

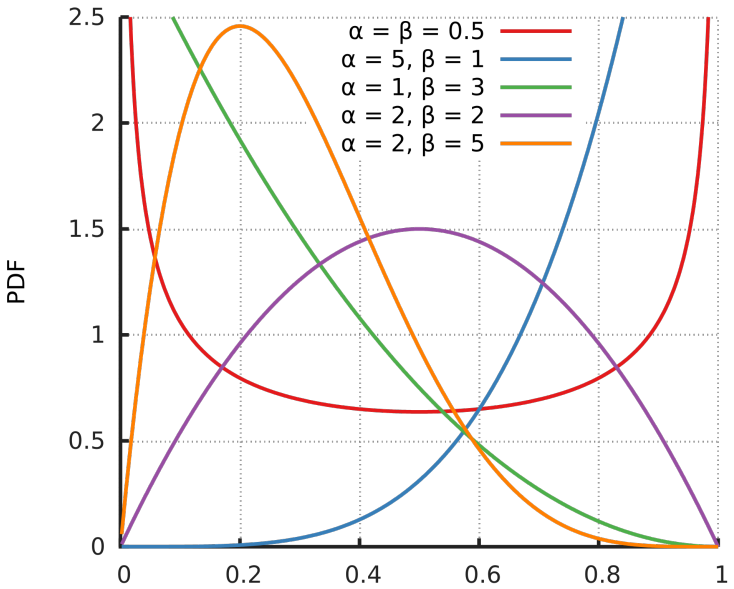


Figure 18: Examples of various Dirichlet Priors. Adapted from Horas [2014].

Let us now consider how the Dirichlet prior updates as we receive more evidence for a particular event. The Dirichlet prior, in Equation 6, and the likelihood function, in Equation 3, have the same form and so multiplying them gives us

$$\prod_{i=1}^{k}\theta_i^{M_i+\alpha_i-1}.$$

Therefore the posterior distribution is simply $\text{Dir}(\alpha_1+M_1,\ldots,\alpha_k+M_k)$, where M_i is the sufficient statistics for the value x_i. In this particular case, the prior had the

same form as the posterior, we refer to prior of this form as a conjugate prior. The Dirichlet distribution is a conjugate prior for the multinomial since it and the multinomial likelihood provide a Dirichlet posterior, which allows us to maintain a probability distribution in a closed form.

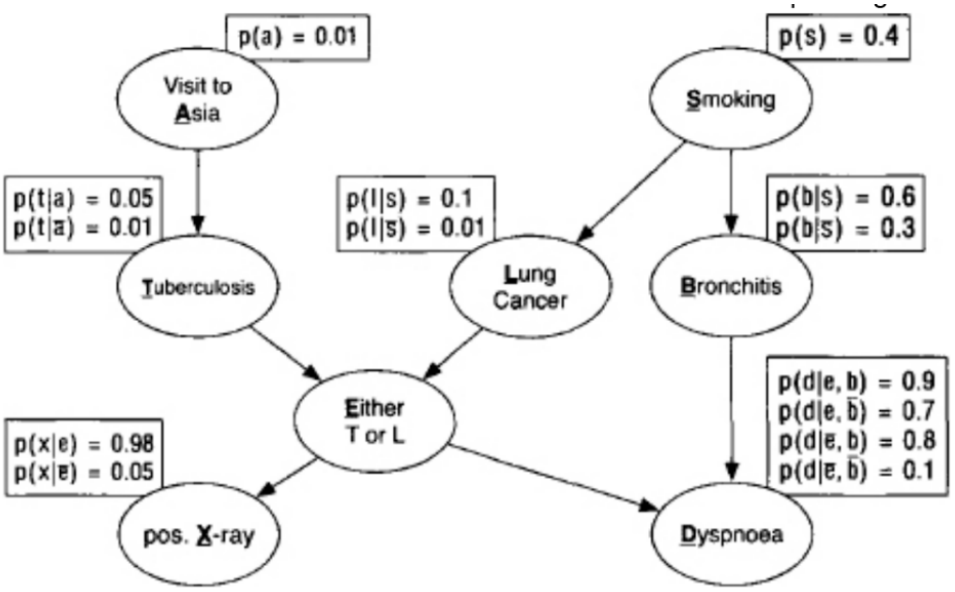


Figure 19: A Bayesian network which presents a distribution over medical knowledge adapted from Lauritzen and Spiegelhalter [1988]. Shortness of breath (dyspnoea) could be caused by tuberculosis, bronchitis, or lung-cancer. Visiting Asia increases ones changes of tuberculosis. Smoking could course bronchitis or lung cancer. X-ray results and indications of dyspnoea do not give us more information about whether the patient has lung cancer or tuberculosis.

Let us now consider the effect of using priors in Bayesian network learning. Figure 19 presents a Bayesian network used to calculate the probability of a patient having lung cancer [Lauritzen and Spiegelhalter 1988]. To measure the effect of priors on Bayesian learning we will sample instances from the network in Figure 19 and using these to relearn the parameters of the network with different priors. We will then measure the distance between the learned network and the true network using a probability distance measure called relative entropy as we increase the number of samples.

Figure 20 shows the results of using maximum likelihood estimation; and uniform Dirichlet priors. We used different imaginary sample counts of 50, 200, and, 5000 on the network in Figure 19. The samples illustrated here are all independent and identically distributed (IID) from the network in Figure 19. The x-axis shows the number of samples and the y-axis is the relative entropy which is the distance between the learned and the ground-truth distribution.

The blue line corresponds to MLE. We see that the blue line is jagged and starts off being higher than all the other learning cases. Although MLE does continue to get lower as we increase the number of samples, it remains relatively uncertain what the true parameters should be.

The orange, gray, and red lines represent Bayesian estimation, where we make use of a prior. All of these examples make use of a uniform prior but different equivalent

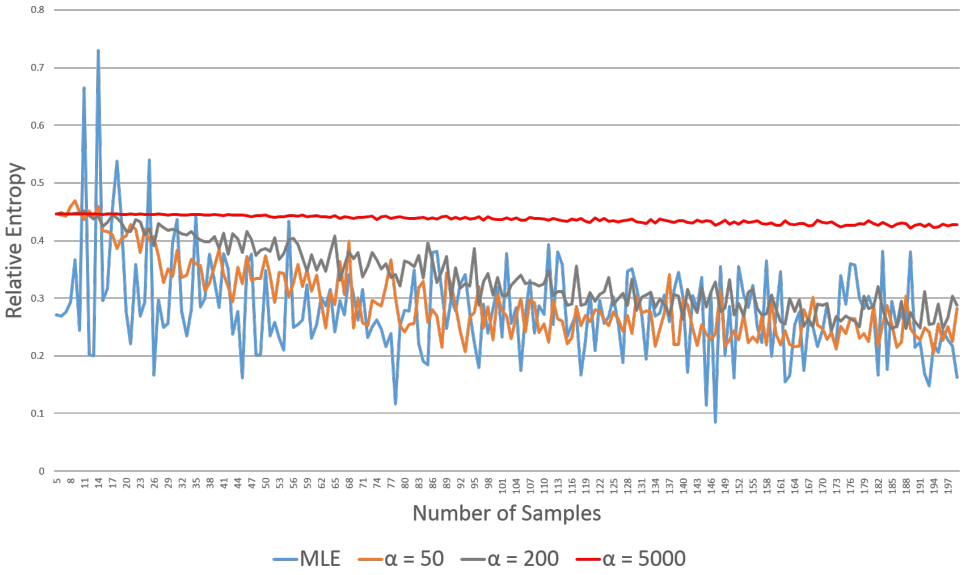


Figure 20: The results of using maximum likelihood estimation; and uniform Dirichlet priors of 50, 200, and, 5000 on the network in Figure 19. The x-axis represents the number of samples and the y-axis represents the relative entropy to the ground-truth distribution. The samples illustrated here are all independent and identically distributed (IID) from the network in Figure 19.

sample sizes. For $\alpha = 50$ (the orange line) and $\alpha = 200$ (the gray line), entropy values are almost overlapping.

As we increase the prior strength to $\alpha = 5000$ (the red line), that is, have stronger confidence that we expect a uniform distribution over factors in the model in Figure 20, we become too ambitious about the expected distribution described. Since the model in Figure 20 is not uniform over its factors the relative entropy increases indicating that the learned model is getting further away from the true model, thus decreasing the performance of the learned model. Thus with a large parameter prior we are more firm about our beliefs about the parameter setting even if we notice a strong signal of evidence in the data to do so.

In Section 2.3.1.1 we described MLE for temporal models. We note that although we represented the parameters using random variables in the provided examples, our discussions on Bayesian estimation should now also fit into the context of temporal models. We therefore omit the discussion of Bayesian estimation in the context of temporal models. In the next section we will discuss how to learn parameters when we have missing or incomplete data.

2.3.1.3 *Learning Latent Variables*

We have thus far explored various representations of Bayesian models and the problem of parameter estimation. We now consider a much more difficult problem of learning from missing or incomplete data. This occurs when values from data are missing or when variables are latent. We often consider latent variables because they

provide a sparse parameterisation of a distribution and can assist in aggregating other variables.

Consider Figure 21 which presents two network configurations. Network (a) shows a configuration without a latent variable and network (b) shows a network configuration with a latent variable L. If we assume that the network parametrisation is a Bernoulli distribution then network (a) can be expressed with 232 parameters (i.e. $(2 \times 4) + 2^5 + (2^6 \times 3)$), whereas network (b) can be expressed with 56 parameters (i.e. $(2 \times 4) + 2^5 + (2^2 \times 4)$). Therefore, although learning latent variables may be difficult it may be worth considering since it provides a sparser representation of the distribution.

Estimating missing data is a well defined problem in statistics presented by [Rubin 1976]. Little and Rubin [2014]; Little [1976] develop the notion as well as explain how to handle missing data at random and deliberately missing data values. Casella and Berger [2002]; Tanner [1991] address whether missing data values are identifiable, and Settimi and Smith [1998]; Garcia [2004] position the problem of missing parameters into the context of Bayesian networks.

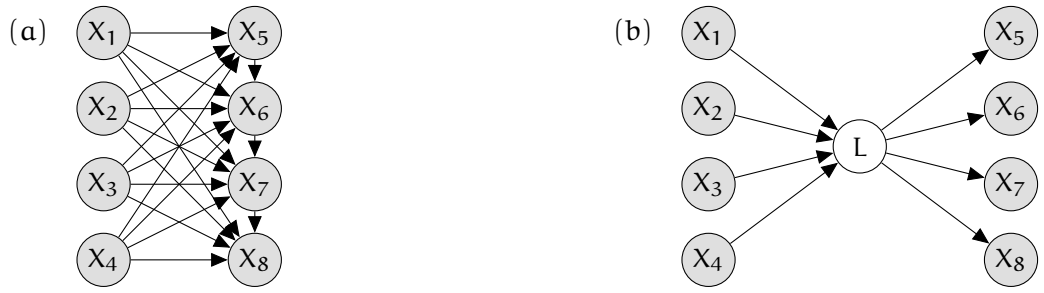


Figure 21: Using a latent variable to provide a sparse independency representation and parameterization between observable variables.

The main issue with partially observed data is its effect on the likelihood function. The likelihood function, for partially observed data, is multi-modal [Koller and Friedman 2009] and can not be easily optimised as we did with complete data in Section 2.3.1.1. The gradient of such multi-modal likelihood functions is explored by Buntine [1994]. Binder *et al.* [1997]; Thiesson [1995]; Greiner *et al.* [2005] propose a gradient ascent method to learning missing values in Bayesian networks.

Another common strategy to optimise the likelihood function is Expectation Maximisation (EM) which attempts to learn both the missing data and the parameters simultaneously. The EM algorithm was proposed by Dempster *et al.* [1977] who generalised several algorithms including the Baum-Welch algorithm for leaning HMMs [Rabiner and Juang 1986]. The general algorithm of EM is outlined in Algorithm 1. Several variants of the EM algorithm are listed by McLachlan and Krishnan [2007]. The discussion of EM in this background follows Neal and Hinton [1998]; Koller and Friedman [2009].

Algorithm 1 Expectation Maximisation

1: **procedure** EXPECTATION-MAXIMISATION(A BN with latent variables/data)
2: Pick a starting point for the parameters.

3: **for** until convergence **do**
4: Complete the data using the current parameters
5: Estimate the parameters relative to data completion
return Data and parameters;

Line 4 in Algorithm 1 is called the E-step. In this step we perform Bayesian inference to infer the data given the parameters. That is for each instance $\xi[m]$ and each family $X, \mathbf{U}$, where $\mathbf{U}$ is the parent set of X, we compute

$$P(X, \mathbf{U}|\xi[m], \theta^t),$$

where θ^t is the current setting of the parameters at iteration t. This is known as a soft completion of the data. We now can compute the expected sufficient statistics, $\hat{M}$, for each combination of values $(x, \mathbf{u})$ per family, where $\mathbf{u}$ is a possible assignment of the parent set to x. We can express the sufficient statistics as

$$\hat{M}_{\theta^t}[x, \mathbf{u}] = \sum_{m=1}^{M} P(x, \mathbf{u}|\xi[m], \theta^t).$$

Line 5 in Algorithm 1 is called the M-step. In the M-step we treat the expected sufficient statistics, $\hat{M}$, as real sufficient statistics and use MLE (as in Section 2.3.1.1) or Bayesian estimation (as in Section 2.3.1.2). The EM algorithm is guaranteed to monotonically improve the likelihood of the parameters relative to the data at each iteration [Neal and Hinton 1998; Koller and Friedman 2009].

EM is commonly used in the naïve Bayesian model to learn the parameters of the latent class variable, see Figure 9. In this situation we are interested in learning the class labels for the observable features presented in the data.

Let us consider the behaviour of EM with a baseline Bayesian network called the ICU alarm network. Figure 22 depicts the ICU alarm network which is a 37 variable network, by Beinlich *et al.* [1989]. The ICU alarm network is considered a baseline for many Bayesian learning problems.

EM is a local search procedure that approximates the likelihood function, or the log-likelihood in practice. Figure 23(a) depicts the behaviour of EM as a function over the number of iterations for the ICU alarm network as adapted from Koller and Friedman [2009]. The x-axis represents the number of iterations and the y-axis shows the log-likelihood of the average training instance. We notice that, as expected, the log-likelihood increases monotonically over the number of iterations.

In Figure 23(a), the log-likelihood function seems to converge at 10 iterations. Figure 23(b) shows the parameter values over the number of iterations for the same ICU Alarm network (Figure 22). In Figure 23(b), it is not clear that the system has converged at 10 iterations. Therefore, the convergence over the likelihood space is not the same as the parameter space and it might be better to judge convergence over the parameter space [Koller and Friedman 2009]. Accelerating the performance and convergence of EM has been investigated [Bauer *et al.* 1997; Ortiz and Kaelbling

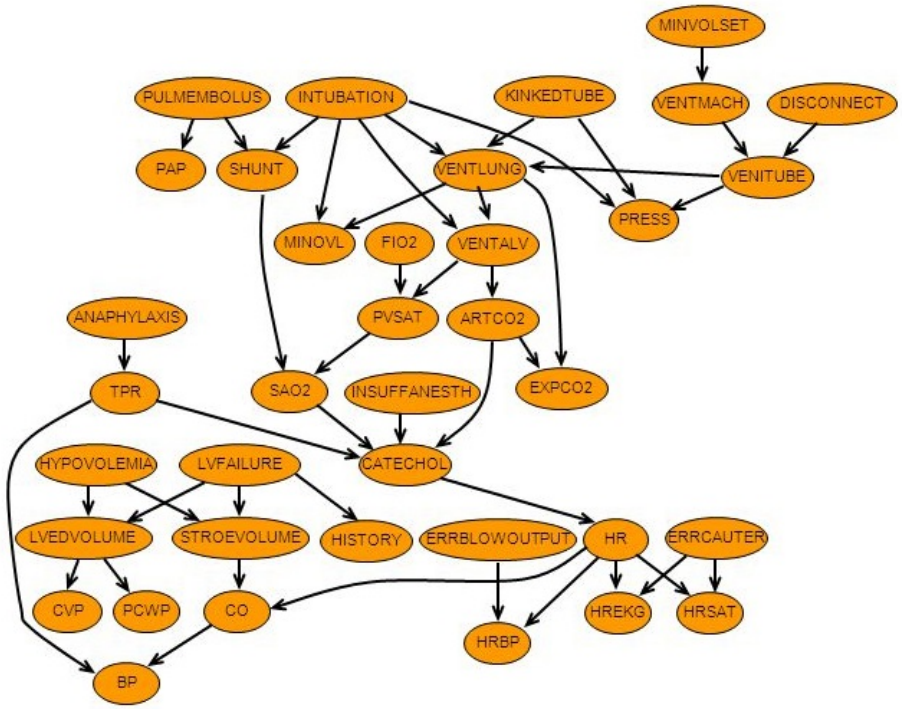


Figure 22: An illustration of the 37 variable ICU alarm network. Adapted from Beinlich *et al.* [1989]; Koller and Friedman [2009].

1999]. The earliest applications and theoretical developments of the EM algorithm include Cheeseman *et al.* [1988 1993]; Ghahramani and Jordan [1994]. Lauritzen [1995] proposed using EM for general probabilistic models.

Let us now consider how we can learn a set of latent variables constructed as a hierarchy which aggregates observations. Figure 24 shows an example of learning the latent variables melody, dynamics, and tone from Mozart's Symphony No. 41 in C major. This example sums up our discussion of learning a hierarchical structure of latent variables from observations. The observations magnitude flux, relative difference, and low energy tell us something about the music's dynamics; and the strongest frequency Fourier max; compactness, and general strongest frequency give us information about the tone [Ajoodha *et al.* 2015]. Finally, the dynamics and tone describe the general melody [Ajoodha 2014; Ajoodha *et al.* 2014]. Details of the observable features used can be found in Ajoodha *et al.* [2015]. This hierarchical structure allows us to aggregate observations into meaningful abstractions using h-trees. The figure also shows the effects of using various priors to learn the parameters of the network.

In this chapter we summarised important contributions and advancements made to the development of representing and learning Bayesian networks. In the next chapter we explore the relevant structure learning literature to aid in extending the language of DBNs for tracking influence between temporal processes.

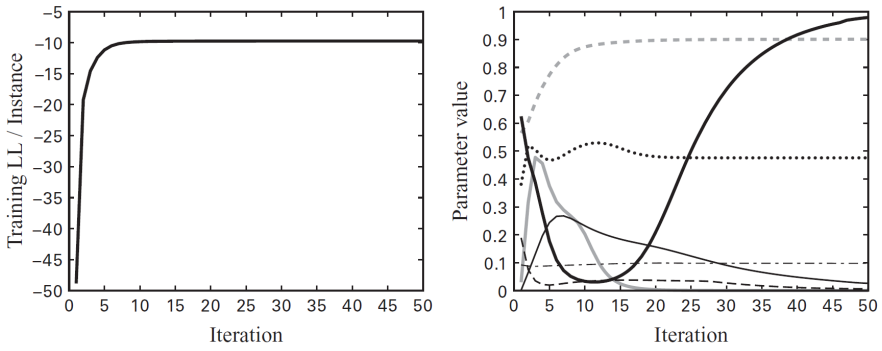


Figure 23: An illustration of the likelihood function improving over EM iterations (a) and the change in parameters (b). This figure was adapted from Koller and Friedman [2009] which is based on the ICU alarm network they presented.

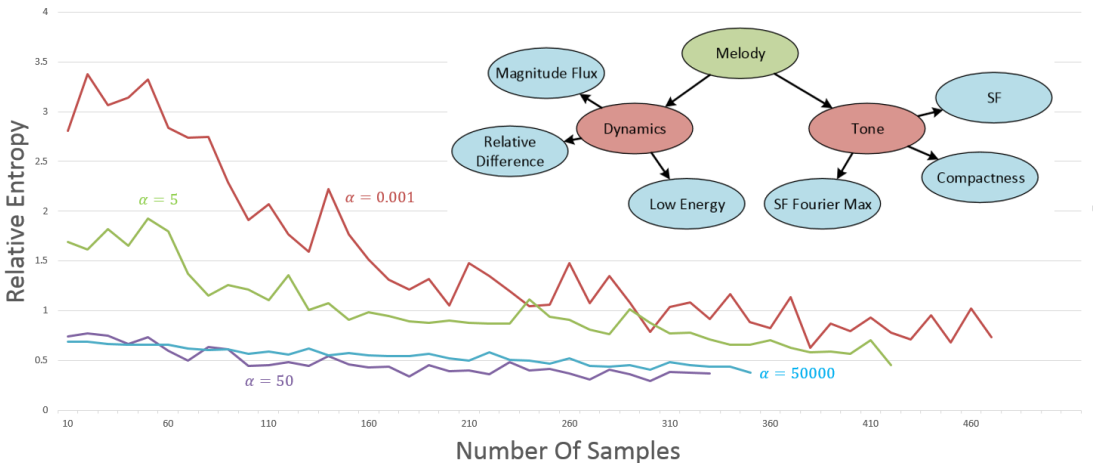


Figure 24: An illustration of effect of parameter priors when learning a hierarchical Bayesian network with latent variables on Mozart's 41^{st} Symphony in C major. There are six observable features (blue) and three latent variables including dynamics (maroon), tone (maroon), and melody (green).

3

RELATED WORK

3.1 INTRODUCTION

Influence relations, in observational data, are mainly deduced though statistical methods [Hatfield *et al.* 2006; Opgen-Rhein and Strimmer 2007; Grinthal and Berkeley Heights 2015; Commenges and Gégout-Petit 2009]. Discovering influence between stochastic processes using statistical methods has been explored with regard to Bayesian network structure learning between random variables which is of great importance to many sciences [Bunge 2017; Salmon 1984; Koller and Friedman 2009].

We may not always be given the structure of a Bayesian network in advance, and might want to learn the Bayesian network structure for either knowledge discovery or density estimation [Heckerman *et al.* 1995a; Mohammadi and Wit 2015; Madsen *et al.* 2017; Fan *et al.* 2014]. On the one hand, if we learn fewer edges than those in the true network structure, there is no possible setting of the parameters that will give us the joint distribution specified in the true network [Koller and Friedman 2009]. On the other hand, if we learn more edges than those in the true network, we might be able to capture the true distribution, but risk fragmentation given more edges than necessary [Koller and Friedman 2009]. Considering the pressures that fragmentation imposes on parameter estimation [D'Elia *et al.* 2003], it might be better to generalise the true distribution with a sparser structure even though we might not completely capture the true distribution [Koller and Friedman 2009]. There are two common approaches to structure learning in Bayesian networks: constraint-based structure learning [Colombo and Maathuis 2014] and score-based structure learning [Koller and Friedman 2009; Campos and Ji 2011].

In *constraint-based* structure learning [Campos and Ji 2011] we view the Bayesian network structure as a set of independence assumptions, for which we can discover a perfect map by performing multiple hypothesis tests for conditional independence between variables [Lehmann and Romano 2006; Cheng *et al.* 1997]. Statistical tests often can be wrong about certain independence assertions [Kanji 2006]. When learning the distribution of influence between processes we are required to test a large number of hypotheses (given the length of the processes involved). If we are wrong about certain hypotheses, this error could grow which reduces our ability to recover the true Bayesian network structure regardless of the tests significance level [Koller and Friedman 2009]. Thus declaring independence assertions which are incorrect will bias the learning procedure to an incorrect structure [Koller and Friedman 2009] which reduces our ability to recover the true set of independence assumptions. This is since every choice of adding an independence assumption restricts the type of assumptions which can be added in future. Therefore, constraint-based approaches are best suited to networks with a small amount of variables; a large amount of samples

with strong presence of dependencies; and is efficient to finding the true structure when faced with a structure which as already quite close to it [Koller and Friedman 2009]. Constrant-based approaches are therefore unsuitable for our task of tracking influence between partially observed multi-dimensional processes. Another promising approach, called score-based structure learning [Bouchaala *et al.* 2010], resolves these restrictions by considering the complete structure as a state in the search space.

Score-based structure learning requires the definition of a hypothesis space of potential network structures; defines a scoring function which gauges how well the model fits the observed training data; and finally, attempts to find the highest scoring network structure as an optimisation problem [Kok and Domingos 2005; Tenenbaum *et al.* 2011; Tsamardinos *et al.* 2006; Lee *et al.* 2007]. However, given that the search space is super-exponential in size (given the number of possible DAGs with respect to the number of variables considered), this proposes an NP-hard problem which we attempt to solve using heuristic techniques [Chickering *et al.* 1994; Chickering 1996; Chickering *et al.* 2004; Suzuki 2017].

Score-based structure learning is best suited to the task of learning influence networks since it (a) considers the complete influence structure as a state in the search space (Section 3.6); (b) preserves basic score properties to allow for feasible computations (Section 3.2 and Section 3.7); and (c) it provides a clear indication of the independence assertions between the concerned networks relative to the data (Section 3.6) [Koller and Friedman 2009; Campos and Ji 2011; Ellis and Wong 2008]. In this section we review related traditional score-based structure learning practices.

This chapter is structured as follows. In Section 3.2 we explore a score which provides the likelihood of a proposed structure relative to the data. In Section 3.3 we introduce an extension to the likelihood score called the Bayesian information criterion (BIC) which makes use of a penalty to manage structural complexity. In Section 3.4 we discuss the Bayesian score which uses the Bayesian paradigm. In Section 3.5 we review related structure search techniques to recover tree structures. In Section 3.6 we present practices that recover graph structures which is a much more difficult problem than recovering tree structures. Finally, in Section 3.7 we present the complexity of the general structure learning task.

3.2 THE LIKELIHOOD SCORE

Recall that score-based structure learning requires the definition of a scoring function which gauges how well the model fits the training data. There are several choices of scoring functions geared at evaluating the likelihood of a particular structure given the fit to data [Drton and Maathuis 2017]. A well-known choice is that of the likelihood score which maximises the likelihood (or log-likelihood in practice) of the structure to the data [Beretta *et al.* 2017]. We can express this using the maximum likelihood estimate, $\hat{\theta}$, given a particular graph structure, $\mathcal{G}$, relative to the data, $\mathcal{D}$. Thus we denote the likelihood score more formally as,

$$\text{score}_L = \ell((\hat{\theta}, \mathcal{G}) : \mathcal{D}). \tag{7}$$

In other words, if we are presented with a particular graph structure we find the MLE of that graph with respect to the data. This can be expressed more generally as the relative cost of adding an edge between two variables in an empty graph structure. More formally, we can define the likelihood score as follows:

Proposition 3.1. The likelihood score decomposes as the number of instances in the data multiplied by the mutual information, $\mathbf{I}_{\hat{P}}$, between each family of variables minus the entropy of each variable considered in isolation (i.e. independent of the structure). More formally,

$$\text{score}_L(\mathcal{G}, \mathcal{D}) = M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathbf{Pa}_{X_i}^{\mathcal{G}}) - M \sum_{i=1}^{n} \mathbf{H}_{\hat{P}}(X_i),$$

where

$$\mathbf{I}_{\hat{P}}(X; Y) = \sum_{x,y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)},$$

and

$$\mathbf{H}_{\hat{P}}(X) = -\sum_{x} P(x) \log P(x).$$

Intuitively, the mutual information informs us of the average cost of adding an edge between X and Y (ie. $P(x, y)$), over modelling these as mutually independent (ie. $P(x)P(y)$) [Koller and Friedman 2009]. Since the entropy term is independent of the graph structure, it is often ignored.

Proposition 3.1 intuitively tells us that the score will be higher if there is evidence in $\mathcal{D}$ which supports a high correlation of variables with their respective parents [Drton and Maathuis 2017; Liu *et al.* 1996]. However, the likelihood score has a strenuous consequence: maximising the likelihood score will always prefer the most connected network given that more edges always give more information about correlations between variables [Lee *et al.* 2007; Koller and Friedman 2009]. This is true unless variables are truly empirically independent, which is never the case in most datasets given statistical noise [Koller and Friedman 2009].

The fact that the most complicated network is always preferred poses a significant over-fitting problem. This is usually overcome by regularising the hypothesis space or penalizing structural complexity [Koller and Friedman 2009; Campos and Ji 2011]. In the next section we discuss a structure score which addresses this problem.

3.3 THE BAYESIAN INFORMATION CRITERION

The Bayesian information criterion (BIC), developed by Schwarz and others [1978], is a popular choice for trading off model complexity and fit to data. The BIC score consists of two terms: the first describes the fit of the hypothesised structure to the

data, usually the likelihood function in Equation 7; and the second is a penalty term for complex networks. More formally the BIC score is given as

$$\text{score}_{\text{BIC}} = \ell(\hat{\theta}_{\mathcal{G}} : \mathcal{D}) - \frac{\log M}{2}\text{DIM}[\mathcal{G}], \tag{8}$$

where M is the number of training instances and the $\text{DIM}[\mathcal{G}]$ is the number of independent parameters in the network. We note that the negation of this score is referred to as the minimal description length (MDL) score [Bouckaert 1993; Lam and Bacchus 1993; Suzuki 1993; Kullback 1997; Chow and Liu 1968].

Upon further investigation, we note that the entropy component (ie. $\mathbf{H}_{\hat{P}}(X)$) of the likelihood term in Equation 8 is negligible since it does not depend on the selected structure. This observation allows us to rewrite the BIC score as

$$\text{score}_{\text{BIC}} = M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathbf{Pa}_{X_i}^{\mathcal{G}}) - \frac{\log M}{2}\text{DIM}[\mathcal{G}]. \tag{9}$$

As we increase the number of samples (ie. M) in Equation 9 the emphasis moves from model complexity to the fit to data [Chen and Gopalakrishnan 1998]. In other words, as we obtain more data we are more likely to consider more complicated structures [Tamura *et al.* 1991]. This property is referred to as consistency. More formally [Koller and Friedman 2009],

Definition 3.2. A scoring function is said to be *consistent* if, as the amount of data provided increases to infinity, the true structure will maximise the score; and all structures that are not I-equivalent (Definition 2.6) to the true structure will have a strictly lower score.

Alongside the definition of consistency (Definition 3.2) is the notion of score equivalence which states that I-equivalent structures have the same structure score. More formally,

Definition 3.3. A scoring function is said to be *score equivalent* if for all I-equivalent networks $\mathcal{G}$ and $\mathcal{G}'$ we have that $\text{score}(\mathcal{G} : \mathcal{D}) = \text{score}(\mathcal{G}' : \mathcal{D})$ for all datasets $\mathcal{D}$.

Schwarz and others [1978]; Rissanen [1987]; Barron *et al.* [1998] provided much development to the basic properties of the BIC score in addition to establishing its consistency and score equivalence which are used for efficient and manageable searches [Geiger *et al.* 2001; Rusakov and Geiger 2005; Settimi and Smith 1998]. In the next section we explore the Bayesian score which considers the Bayesian paradigm.

3.4 THE BAYESIAN SCORE

The last score that we discuss in detail is the Bayesian score. The Bayesian score, with a Dirichlet prior, was introduced by Buntine [1991]; Cooper and Herskovits [1992]; Spiegelhalter *et al.* [1993]; Heckerman *et al.* [1995a]. The Bayesian score operates under the Bayesian paradigm which states that anything we have uncertainty over we

must maintain a probability distribution over. In this case we have uncertainty over parameters as well as network structure. More specifically,

$$P(\mathcal{G}|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{G})P(\mathcal{G})}{P(\mathcal{D})}, \tag{10}$$

where $P(\mathcal{D}|\mathcal{G})$ is the marginal likelihood (not the maximum likelihood); $P(\mathcal{G})$ is the prior over structures; and finally, $P(\mathcal{D})$ is a constant term which describes the prior over datasets. $P(\mathcal{D})$ gives us no information about the structure of the graph and can be ignored when being used to compare different structures. We can rewrite Equation 10 using a sum of logs as

$$\text{score}_B(\mathcal{G} : \mathcal{D}) = \log P(\mathcal{D}|\mathcal{G}) + \log P(\mathcal{G}). \tag{11}$$

The marginal likelihood term expands as follows

$$P(\mathcal{D}|\mathcal{G}) = \int_{\Theta_\mathcal{G}} P(\mathcal{D}|\theta_\mathcal{G}, \mathcal{G}) P(\theta_\mathcal{G}|\mathcal{G}) d\theta_\mathcal{G},$$

where $P(\mathcal{D}|\theta_\mathcal{G}, \mathcal{G})$ is the likelihood for data given the structure and parameters; and $P(\theta_\mathcal{G}|\mathcal{G})$ is the prior over the parameters. We note that the marginal likelihood integrates over all possible settings of the parameters (not just the maximum likelihood $\hat{\theta}$). Therefore in this case $\theta_\mathcal{G}$ is a setting of the parameters for a structure $\mathcal{G}$.

The idea of using the marginal likelihood, as an alternative to the maximum likelihood, significantly reduces over-fitting since it makes a prediction over an unseen instance given the previous instances. It is useful to think about marginal likelihood as a type of cross validation process for each new instance, which is opposed to maximum likelihood estimation which considers all the instances together.

Although the Bayesian score might seem complicated, if we consider multinomial table CPDs and no structure in $P(\theta_\mathcal{G}|\mathcal{G})$, then the marginal likelihood can be provided in closed form as

$$P(\mathcal{D}|\mathcal{G}) = \prod_i \prod_{\mathbf{u}_i \in Val(Pa^\mathcal{G}_{X_i})} \frac{\Gamma(\alpha^\mathcal{G}_{X_i|\mathbf{u}_i})}{\Gamma(\alpha^\mathcal{G}_{X_i|\mathbf{u}_i} + M[\mathbf{u}_i])} \prod_{x_i^j \in Val(X_i)} \frac{\Gamma(\alpha^\mathcal{G}_{x_i^j|\mathbf{u}_i} + M[x_i^j, \mathbf{u}_i])}{\Gamma(\alpha^\mathcal{G}_{x_i^j|\mathbf{u}_i})},$$

where M is a sufficient statistic (Definition 2.17) and the gamma function is given by

$$\Gamma(x) = x.\Gamma(x-1).$$

It is important to recognise that the above expression allows us to express the log marginal likelihood as a sum over Bayesian family scores. Turning our attention to the structure prior in Equation 11, the most common choice is a uniform structure prior. Other structure priors include priors that are non-uniform over Bayesian network structures [Buntine 1991] or deviations between prospective networks and recommended networks [Heckerman *et al.* 1995b]. Another score closely related to the Bayesian score is the BDe score. The BDe score was proposed by Heckerman *et al.* [1995a], and the authors showed that it possesses basic properties such as score-equivalence and local parameter independence. For a more detailed empirical analysis of these scores see Dawid [1984]; Kass and Raftery [1995].

A derivation of the BIC score is the AIC score [Akaike 1974] which only considers the dimension of the graph structure (ie. the number of independent parameters) as a penalty term. More specifically,

$$\text{score}_{AIC} = M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i; \mathbf{Pa}_{X_i}^{\mathcal{G}}) - \text{DIM}[\mathcal{G}].$$

Practically, the major difference between the AIC and BIC scores is the size of the penalty term. The BIC score penalizes complex structures more heavily, thus the AIC score may capture more independence assumptions which hypothesically provides a better likeness to the data.

In summary, the Bayesian score avoids over fitting by averaging over all possible network parameterisations and is asymptotically equivalent to the BIC score [Koller and Friedman 2009].

3.5 LEARNING TREE-STRUCTURED NETWORKS

Learning a tree structured network is perhaps the simplest structure learning problem. The first application of learning tree structured networks was proposed by Chow and Liu [1968]. Having selected a structure score, we turn our attention to an optimisation problem which attempts to maximise the selected score over potential structures. Decomposability of the score turns out to be an important property for computational savings in the structure search procedure. More specifically, decomposability is the property that the complete network score can be written as equal to the sum of family scores. For example, in Preposition 3.1 we see that the likelihood score decomposes as a sum of family scores (for every variable and its parent-set).

There are two important reasons why one would want to learn a tree structured network over a graph structured network. Firstly, there already exists powerful algorithms for efficient optimisation over high-dimensional tree structured networks; and secondly, trees provide sparse networks with manageable generalised parameterisation which reduces over-fitting. In summary, trees can be constructed by using polynomial time algorithms which summarise the most important dependencies that allow for better approximations [Koller and Friedman 2009; Bunge 2017]. Trees can also offer a suitable prior which can be used to learn more descriptive graph structures relative to the data [Koller and Friedman 2009].

In order to learn tree structures we need to calculate the score between different variables. If our score is decomposable we can express the weight between a variable, j, and its parent, i, as

$$w_{i\to j} = \text{score}(X_j|X_i) - \text{score}(X_j). \tag{12}$$

In the case of the likelihood score the expression of $w_{i\to j}$ in Equation 12 becomes

$$w_{i\to j} = M \sum_{x_i, x_j} P(x_i, x_j) \log \frac{P(x_i, x_j)}{P(x_i)P(x_j)}. \tag{13}$$

Intuitively, the weight between two variables is the average over the relative error of modelling the joint distribution between the variables and the product of them being marginally independent.

Now that we have a way to measure the score of two variables we can define an algorithm to obtain a tree structured network. A general algorithm to obtain a tree structured network is to compute the score of every pair of variables and then find the maximum weighted spanning tree (MWST) or forest between all of the variables. One could use any standard MWST algorithm such as Prim or Kruskal in $O(n^2)$ [Greenberg 1998; Huang *et al.* 2009], where n is the number of variables.

There are two important observations to note from Equation 13. The first is that the mutual information term can never be negative. However, in the BIC or BDe scores we could have a penalty term which causes $w_{i\to j}$ to be negative. If we use the likelihood score for $w_{i\to j}$ then the score between the two variables will be positive. Thus using the likelihood score with a MWST algorithm will result in a tree structured network as opposed to obtaining a forest when using penalty-based scores such as the AIC or BIC scores. This is the case since a penalty based score could a negative weight with is ignored by MWST algorithms such as in Prim and Kruskal [Greenberg 1998; Huang *et al.* 2009].

Secondly, score equivalent networks have the same score for $w_{i\to j}$ and $w_{j\to i}$, this implies that we will get a undirected tree or forest structure which we can then impose an orientation on the edges. It is important to note that in practical situations the mutual information term between various variable groupings often remain uninformative, this can severely limit the utility of the pair-wise approach mentioned here which is why we later consider using groups of variables in graph structures.

Once the score of all pairs of variables are found, one can use an arbitrary orientation on the edges, as long as the resulting graph is a DAG. In the next section we turn our attention to the much more difficult problem of learning graph structured networks.

3.6 LEARNING GENERAL GRAPH-STRUCTURED NETWORKS

In the case of learning general graph-structures the problem complexity increases greatly. More specifically [Koller and Friedman 2009],

Theorem 3.4. *For any dataset, $\mathcal{D}$, and decomposable structure score,* score, *the problem of finding the maximum scoring network, that is,*

$$\mathcal{G}^* = \arg\max_{\mathcal{G}\in\mathcal{G}_d} \text{score}(\mathcal{G} : \mathcal{D}),$$

is NP-Hard for any $d \geqslant 2$, where $\mathcal{G}_d = \{\mathcal{G} : \forall i, |Pa^{\mathcal{G}}_{X_i}| \leqslant d\}$.

In other words, finding the maximal scoring network structure with at most d parents for each variable is NP-hard for any d greater than or equal to 2 [Chickering *et al.* 1994; Chickering 1996; Chickering *et al.* 2004; Suzuki 2017]. This is because of the super-exponential search space that one has to traverse to obtain the maximal scoring

network. Koivisto and Sood [2004]; Singh and Moore [2005]; Silander and Myllymaki [2012] provide a detailed description and proof of NP-Hardness of this combinatorial problem.

There has been much work to further explore the repercussions of the result in Theorem 3.4. More specifically, learning a general graph structure is NP-hard for a bounded in-degree less than or equal to d for the Bayesian score [Chickering 1996; Chickering *et al.* 2004]; polytrees (graphs with underlying tree structures) [Dasgupta 1999]; finding a path in a graphical model [Meek 2001]; and even when certain properties of the problem are known (eg. perfect independence, generative, inference, or information oracle) [Chickering *et al.* 2004].

The result in Theorem 3.4 might be discouraging. However, due to the developments of local search procedures we are able to provide a solution using a heuristic hill-climbing search.

Chickering *et al.* [1995]; Buntine [1991] proposed a local search over graph structures. The intuition of the heuristic local search procedure is as follows. Suppose we have a arbitrary candidate network as depicted at the center of Figure 25 labeled (a). We can decide to perform local perturbations in an attempt to improve the network structure relative to the data and a particular score. Suppose that we are given the following options: to reverse the edge $X_2 \rightarrow X_5$ obtaining network (b) which gives us a score of 75; to delete the edge $X_6 \rightarrow X_5$ obtaining network (c) which gives us a score of 90; to add an edge $X_4 \rightarrow X_2$ obtaining network (d) which gives us a score of 50; or to reverse the edge $X_5 \rightarrow X_3$ obtaining network (e) which results in a cyclic network.

Obviously we do not consider network (e) since this is not a legal Bayesian network. The most favorable transition would be to delete the edge $X_6 \rightarrow X_5$. This option improves the current network score from 70 to 90. It is not clear whether this transition will be favorable in the long term, however, it does provide a better network structure from a single transition.

There are two main design choices that one needs to make when performing a local structure search: the choice of search operators and search procedure. Firstly, we must select a set of operators which are local steps to traverse the search space. Common choices for local search operators are edge addition, reversal and deletion.

The edge reversal search operator might seem counter intuitive since it can be achieved by a simple edge deletion and edge addition. When optimising that structure score over the search space, deleting an edge with the intention to perform a edge reversal would lower the overall structure score for the next step. Therefore, having an edge reversal transition allows us to explore the option of reversing edges without encountering this issue.

There are several other search spaces one could consider such as moving a variable to a new position in the network [Moore and Wong 2003]; searching over ordering and bounded in-degrees [Teyssier and Koller 2012]; or perturbing the data out of local optimum [Elidan *et al.* 2002]. Although methods which take larger steps in the search space may be expensive, they are empirically faster and more resistant to local optimum [Chickering 1996 2002; Elidan *et al.* 2002; Teyssier and Koller 2012].

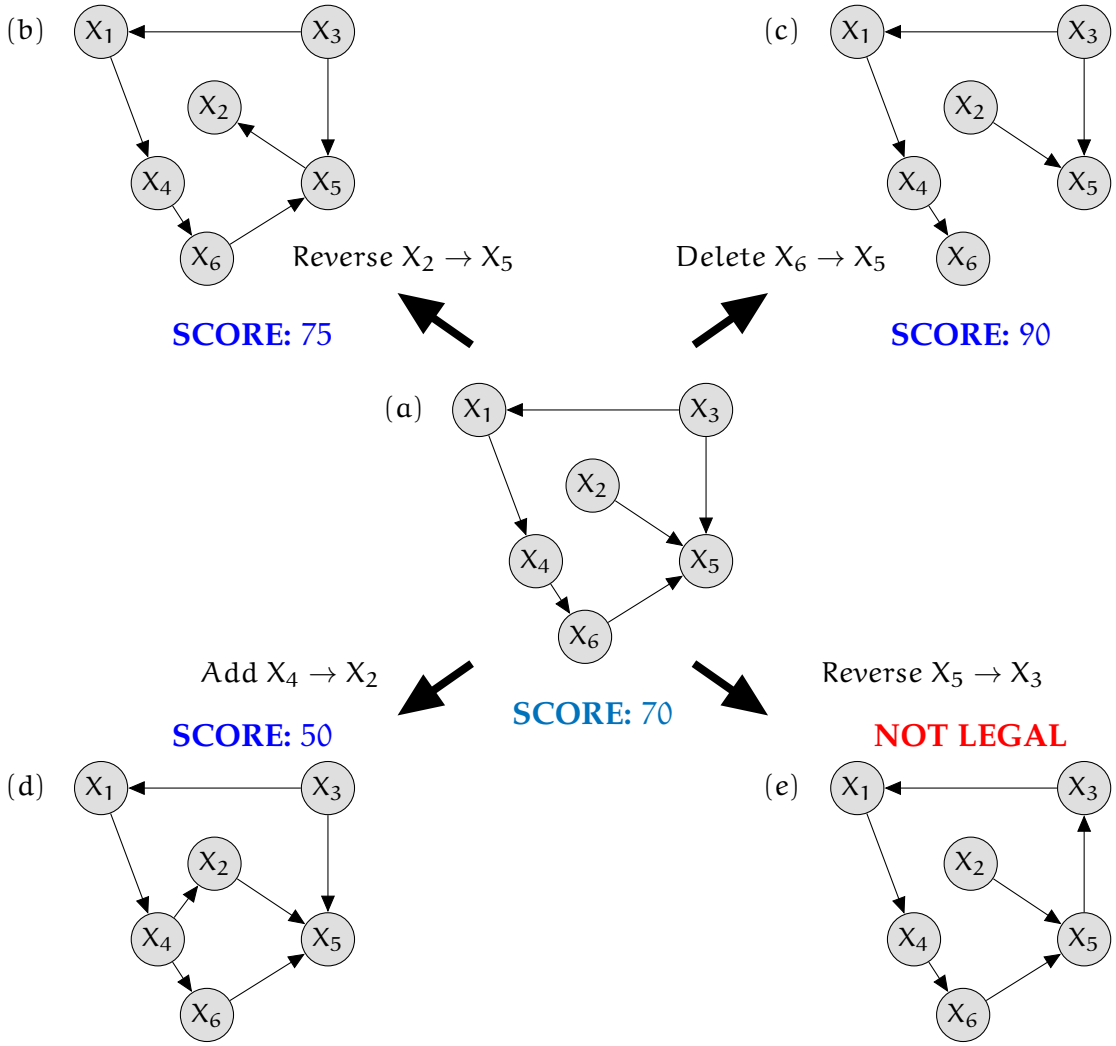


Figure 25: An illustration of the heuristic search procedure between different candidate Bayesian network models. This figure shows local perturbations in an attempt to improve the network structure relative to the data and a particular score. Network (a) provides the current network structure. Network (b) reverses the edge $X_2 \to X_5$ which increases the network score by 5. Network (c) deletes the edge $X_6 \to X_5$ which increases the score by 20. Network (d) adds an edge $X_4 \to X_2$ which decreases the score by 20. Finally, network (e) reverses the edge $X_5 \to X_3$ obtaining an illegal network structure.

The second design choice is to select a search technique that traverses the search space. Some choices include greedy hill-climbing, best first search, or simulated annealing. The most common choice is greedy hill-climbing (GESS) which starts with a prior network. The prior network could be an empty network; a best tree obtained from the procedure mentioned in Section 3.5; a random network; or one elicited by an expert. From this prior network we iteratively try to improve the network by utilising search operators. In greedy hill-climbing we always apply the change that improves the score until no improvement can be made. A comparison was done on various local search procedures by Chickering *et al.* [1995], such procedures included the K2 algorithm, local search, and simulated annealing. Chickering *et al.* [1995] note that local search provides the best time-accuracy trade-off to all other methods compared. Cooper and Herskovits [1992] were perhaps the first authors to propose a general

graph search called the K2 algorithm. The K2 algorithm used an ordering over variables which would permit families of variables to specify an orientation.

There are two major issues with the greedy local search procedure. The resulting network structure can be interpreted as a local optimum for which no operator can improve the network score. Suppose we started the local search procedure at position C in Figure 26. We might unfortunately arrive at a poor local maximum, position A, relative to the global maximum, position B, which is considerably better.

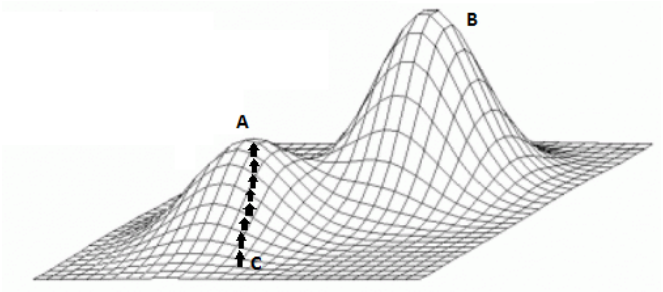


Figure 26: An illustration of a local search procedure which takes small steps in the search space and arrives at a local optimum [Tompkins and Lawley 2009].

The second problem arises when we encounter a plateau in the search space. Consider Figure 27, at position B where if we look around the current structure there may be a variety of possible network transitions which give the same score. In this case there is no way to know which direction to proceed towards.

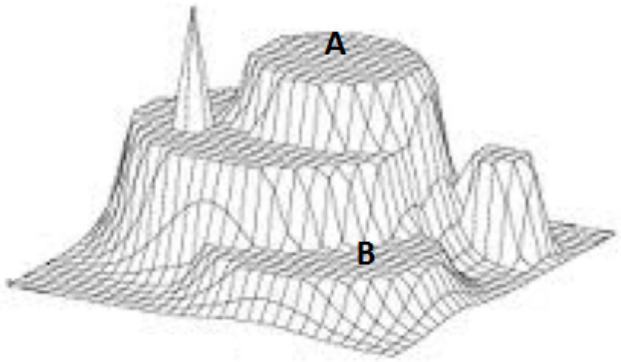


Figure 27: An illustration of a state space with many plateaux [Tompkins and Lawley 2013].

This occurs frequently in Bayesian network structure learning because of I-equivalent network structures yielding the same score given a score-equivalent structure score (eg. those in Figure 8). In order to avoid these issues in practice we augment greedy hill-climbing with two search techniques: random restarts and tabu lists.

RANDOM RESTARTS: In random restarts, when we reach a local optimum we take k random steps and then continue traversing the search space using the search procedure. The intuition is that if we are at a local maximum that is shallow then k random steps will set us up in a better position to explore a better optimum.

TABU LISTS: In tabu lists we try to avoid treading the same path over and over again by maintaining a list of the most recent k steps taken by the search procedure. Glover and Laguna [2013] provide a detailed discussion on the effects of tabu lists on structure search procedures.

Figure 28 presents the effects of learning parameters only and learning both parameters and structure on the ICU alarm network presented in Figure 22 [Koller and Friedman 2009]. The x-axis shows the number of samples (M) and the y-axis shows the distance measured as K-L divergence [Minka 2005] between the learned and true distribution. The solid line in the figure indicate both structure learning and parameter estimation and the dotted line indicates learning only parameters given the true structure.

The result in Figure 28 suggests that the structure learning problem is not intrinsically more difficult than the parameter estimation problem given observable data. However, learning from synthetic data has a much stronger signal for the correlation between variables than in real data where their exists statistical noise.

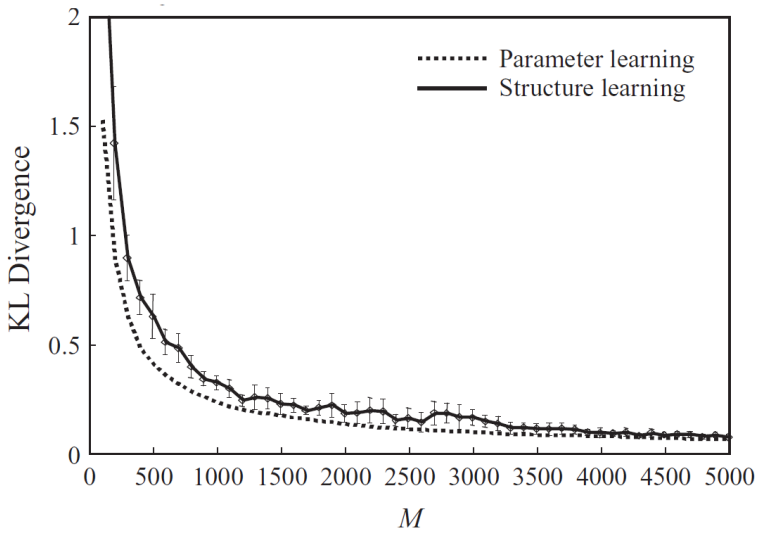


Figure 28: The performance of two learning tasks for Bayesian networks. Adapted from Koller and Friedman [2009]. The dotted line represents parameter estimation given the true structure, and the solid line represents structure and parameter learning.

Other examples of related work using local search procedures and traversal algorithms include a search over I-equivalent classes [Chickering 1996 2002]; and constraint-based algorithms which guarantee obtaining the correct network at a large sample limit, such as the SGS [Spirtes *et al.* 2000] and KES [Nielsen *et al.* 2002] algorithms. Höffgen [1993] attempted to assess the rate of learning over the number of samples for Bayesian network structures with bounded in-degree. Höffgen [1993] presents that relative entropy of the ground truth grows logarithmically with the number of samples. Similar work also explores the effects of using structure scores with penalties [Friedman and Yakhini 1996]. Abbeel *et al.* [2006] present a polynomial-time algorithm to learn a low-degree maximum-likelihood Bayesian network. Learning the structure for dynamic Bayesian networks [Friedman *et al.* 1998] and object-relational models [Friedman *et al.* 1999; Getoor *et al.* 2002] have also been proposed.

In the context of learning the structure of a dynamic Bayesian network, the problem is considered in the complete and incomplete data cases. On the one hand, in learning the dynamic Bayesian network structure for complete data, we simply learn the initial and unrolled states using algorithms to recover a static Bayesian network structure. Such algorithms include Cooper and Herskovits [1992]; Heckerman *et al.* [1995a]. On the other hand, in the case of incomplete data, we may consider learning the position of latent variables or knowing the internal structure of each dynamic Bayesian network and learning their missing values. Discovering the structure of latent values has been addressed by Friedman and others [1997]; Friedman [1998] using the Structural Expectation Maximization (EM) algorithm. The Structural EM algorithm perturbs the structure and parameters of the model iteratively. This method is proven to be effective to recover the structure of dynamic Bayesian networks when we do not know the position of the latent variables. Furthermore, since the Structural EM algorithm uses score-based structure learning, it is readily adaptable to using regularization techniques such as the BIC or BDe scores to control structural complexity [Schwarz and others 1978; Heckerman *et al.* 1995a]. Friedman *et al.* [1998] analyse the algorithmic complexity of this learning problem and present some literature which attempts to reduce the computational burden of local search procedures by a clever use of data structures. In this research we will assume that each individual process is represented by a dynamic Bayesian network with a known structure and some variables with missing values.

3.7 STRUCTURE LEARNING COMPLEXITY

Learning the structure of a Bayesian network can be viewed as a combinatorial optimisation problem where we attempt to select a structure through a search procedure using a scoring function over a search space. In this section we explore several techniques to reduce the total structure search complexity by taking advantage of score decomposability.

Recall Figure 25 where we evaluate several transitions from an initial network structure using various search operators. The computational cost of this structure search procedure is as follows. Suppose we have a Bayesian network with n nodes, then there exists $n(n-1)$ possible edges. Each edge is either present in the network or absent. Edges which are present can be deleted or modified using edge removal or reversal, and edges which are absent can be added using an edge addition. Therefore, there are asymptotically $O(n^2)$ total operators which we consider at each search step.

To evaluate the score of a network with respect to a selected scoring function we need to calculate the score of n families in the network. For each family we need to calculate the sufficient statistics for the scoring function which takes a traversal through the training data, $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$. This traversal will take $O(M)$ steps. Therefore, it would take $O(nM)$ to evaluate the score of a candidate network.

In order to avoid illegal networks during searching, such as network (e) in Figure 25, we need to perform an acyclicity check on every potential transition network structure. This can be achieved with a simple breadth-first search which requires $O(E)$, where E is the number of edges in the network.

Therefore in total, we require $O(Kn^2(Mn + E))$ to perform the structure search algorithm, where K is the total number of steps to the local optimum structure. For even moderately sized networks (eg. 30 to 60 variables) the computation becomes unmanageable.

The above presents a significant problem in terms of the general complexity of the structure search algorithm. However, we can exploit the decomposability property of the score function for significant computational savings.

Consider the transition from network (a) to (d) in Figure 25. Recall Proposition 3.1 which illustrates the decomposability of the likelihood score. The score of network (a) can be expressed using the score decomposability property as

$$\begin{aligned}\text{Score}(X_1, X_2, X_3, X_4, X_5, X_6) = &\text{Score}(X_1|X_3)\\ &+ \text{Score}(X_2)\\ &+ \text{Score}(X_3)\\ &+ \text{Score}(X_4|X_1)\\ &+ \text{Score}(X_5|X_2, X_3, X_6)\\ &+ \text{Score}(X_6|X_4).\end{aligned}$$

The score for network (d) in Figure 25, decomposes as

$$\begin{aligned}\text{Score}(X_1, X_2, X_3, X_4, X_5, X_6) = &\text{Score}(X_1|X_3)\\ &+ \text{Score}(X_2|X_4)\\ &+ \text{Score}(X_3)\\ &+ \text{Score}(X_4|X_1)\\ &+ \text{Score}(X_5|X_2, X_3, X_6)\\ &+ \text{Score}(X_6|X_4).\end{aligned}$$

The scores of networks (a) and (d) are exactly the same except for variables X_2's family. In network (a) we have $\text{Score}(X_2)$ and in network (d) we have $\text{Score}(X_2|X_4)$. Thus we need only recalculate the family score for X_2. Thus, each transition in the search space can be further expressed in terms of a $\triangle$score between each respective network given the search operators. This notion of a $\triangle$score can also be extended for edge reversal and deletion.

Exploiting the decomposability property allows us to compute one or two family scores at each transition, as opposed to recalculating every family score. Recall that to calculate a family score we need to calculate sufficient statistics for each variable in the family. This would take $O(nM)$ as opposed to $O(n^3M)$. Other methods that attempt to reduce the computational burden in structure learning literature include caching sufficient statistics and the use of data structures such as priority queues [Moore and Lee 1998; Deng and Moore 1995; Moore 2000; Komarek and Moore 2000; Indyk 2004].

In this chapter we reviewed various structure scores; a method to learn a tree-structured network; and local search techniques to recover graph structures. We also explored methods for computational saving given the intractability of the search procedure to traverse the search space. In the next chapter we use the contributions in

this and the previous chapter to introduce influence between stochastic processes. We then begin to develop procedures to recover influence relations between them.

4

THE REPRESENTATION OF DYNAMIC INFLUENCE

4.1 INTRODUCTION

Before developing algorithms to infer influence between stochastic processes, one needs to precisely define and specify the conditions of influence. In this chapter we will specify what is meant by influence in terms of its representation, distribution, and applicable inference. We leave the tasks of scoring and searching for influence networks to Chapter 5 and Chapter 6 respectively.

There are many practical applications where we can make use of influence networks, such as for knowledge discovery or density estimation. That is, to discover the structure of how influence flows from one process to another, or to estimate the probability of a process respectively. These learning tasks are useful for many applications such as for navigation systems [Papageorgiou 1990], where temporal influence between traffic conditions might flow between roads; predicting stock market trends [Cong *et al.* 2008], where temporal influence flows between market characteristics such as risk, volatility, selection, liquidity, regulation (etc.); and even for describing how the quantity of proteins in a cell influence each other as the cell's conditions may change [Sachs *et al.* 2005; Perriere and Thioulouse 2003; Durbin *et al.* 1998].

Temporal influence between processes can be modelled using a single dynamic Bayesian network (DBN), where the influence between processes flow in every time-slice in the DBN. However, overloading the representation of influence between processes, within a single temporal model, can significantly simplify our structure learning task if we model influence relations between processes as making use of conditional independence assumptions. This is especially useful since we can explicitly manage the extent of interactions between ensembles of processes by encoding it within the structure for knowledge discovery. Furthermore, this allows us to separate the structure of influence between processes from the structure of variables within each description of a process.

Therefore, we define a dynamic influence networks (DINs) as a DBN of a set of DBNs with an imposed high-level influence structure. By extending the representation of a DBN, rather then replacing it, we allow for the conservation of essential properties and learning algorithms such as independence maps (Section 2.2.1.2), I-equivalence classes, MLE (Section 2.3.1.1), and Bayesian estimation (Section 2.3.1.2) which we make use of when recovering influence structures between processes in later chapters.

The major contributions of this chapter are as follows:

1. the definition and properties of influence networks which include independency maps, the influence network factorisation, independency equivalence, and encoding the distribution of influence through a structure;

2. the definition of dynamic influence networks (DINs) in terms of two underlying assumptions (Markov and time-invariance), as well as its factorisation;
3. an algorithm to perform approximate particle-based inference on DINs;
4. and finally, an empirical analysis of using a set of independence assumptions between processes in DINs for modelling influence.

This chapter is structured as follows. In Section 4.2 we define an influence network between Bayesian networks; Section 4.3 defines DINs, which is a DBN that represents the distribution of interactions between processes; Section 4.4 discusses particle-based inference for influence networks; and finally, Section 4.5 provides a brief empirical justification for using DINs over modelling this problem using the general product rule over variables in a DBN.

4.2 INFLUENCE NETWORKS

Our primary contribution in this thesis is the notion of influence between processes and ways of recovering it from data. Recall that influence between processes is a set of independence assumptions (encoded as a structure) and a probability distribution. Intuitively, influence between processes means that the values that certain processes take in the observable data may effect the values that other processes take in the observable data.

We begin this chapter by discussing the non-dynamic case of influence. In this problem we are interested in modelling the influence between environments which are described by observable and latent variables in Bayesian networks [Ajoodha and Rosman 2017]. Intuitively, we are interested in recovering how variables in one environment can potentially effect the values of variables in other environments.

This section is structured as follows. Section 4.2.1 defines the structure of the influence network between Bayesian networks; Section 4.2.2 explores the use of I-maps for distributions; Section 4.2.3 explores how influence networks factorise as a probability distribution; Section 4.2.4 defines influence networks; and finally, Section 4.2.5 shows how the same set of independence assumptions can be represented by different structures.

4.2.1 *Influence Structure*

We begin by formally defining the semantics of an influence structure.

Definition 4.1. An *influence graph structure*, $\mathcal{G}^{\mathbb{I}}$, is a directed acyclic graph whose nodes represent Bayesian networks, $\mathbf{B} = \{\mathcal{B}^1, \ldots, \mathcal{B}^R\}$, where R is the number of networks. Each Bayesian network in $\mathbf{B}$ encodes the same set of independence assumptions $\mathcal{G}^{\mathbf{B}}$, but a different set of variables. Let $\mathbf{Pa}^{\mathcal{G}^{\mathbb{I}}}_{\mathcal{B}^i}$ denote a set of parent Bayesian networks for $\mathcal{B}^i$ with respect to the structure $\mathcal{G}^{\mathbb{I}}$, and $\mathbf{NonDescs}_{\mathcal{B}^i}$ denote the set of Bayesian networks that are not descendants of $\mathcal{B}^i$ in $\mathcal{G}^{\mathbb{I}}$. Then $\mathcal{G}^{\mathbb{I}}$ encodes the following set of conditional independence assumptions, called local independencies, denoted by $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$,

$$\forall\, \mathcal{B}^i \in \mathbf{B}\colon (\mathcal{B}^i \perp\!\!\!\perp \mathbf{NonDescs}_{\mathcal{B}^i|\mathbf{Pa}^{\mathcal{G}^{\mathbb{I}}}_{\mathcal{B}^i}}).$$

In Definition 4.1 we encode in the local independency set $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$ that any Bayesian network in this influence structure is conditionally independent of its non-decendents given its parents. The assignment of the dependency set can be based on relating latent variables, aggregating observations, or even a total assignment of all variables from one network to another. We will defer the discussion of how the assignment of the dependency set in $\mathcal{G}^{\mathbb{I}}$ is encoded as conditional independence assumptions to our discussion on structural assembles for influence networks in Chapter 5.

Figure 29 illustrates an example of an influence network structure, $\mathcal{G}^{\mathbb{I}}$, with four Bayesian networks, $\mathbf{B} = \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\}$, that encode the same set of independence assumptions $\mathcal{G}^{\mathbf{B}}$. Examples of members of the set $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$ include $(\mathcal{C} \perp\!\!\!\perp \mathcal{B} \mid \mathcal{A})$ and $(\mathcal{D} \perp\!\!\!\perp \mathcal{A} \mid \mathcal{B}, \mathcal{C})$.

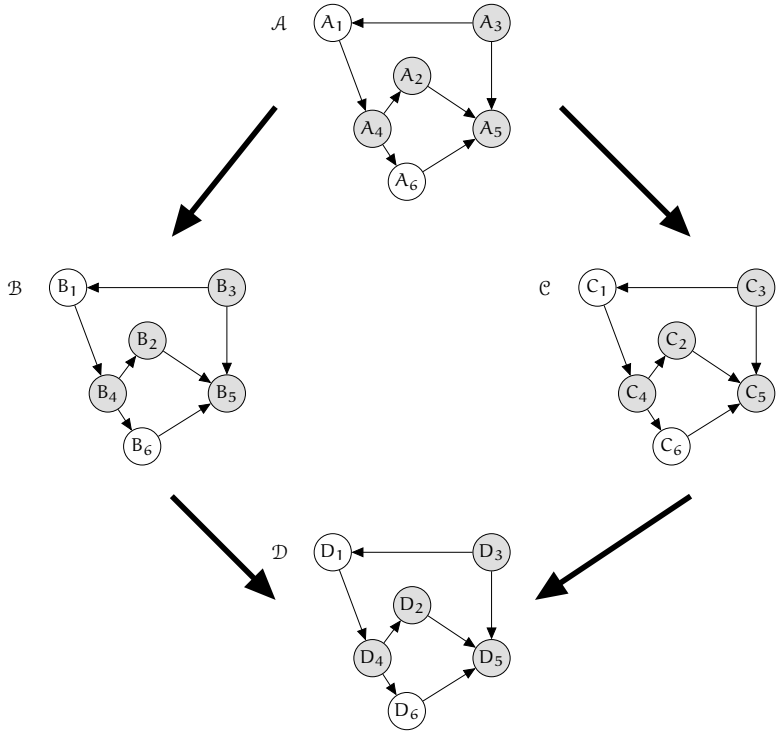


Figure 29: An illustration of an example of an influence network structure, $\mathcal{G}^{\mathbb{I}}$, with four Bayesian networks, $\mathbf{B} = \{\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}\}$, that encode the same set of independence assumptions $\mathcal{G}^{\mathbf{B}}$ (thin solid lines). The thick solid lines indicate the structure $\mathcal{G}^{\mathbb{I}}$. More specifically, the thick solid lines indicate independence assumptions between variables in different Bayesian networks. E.g. $P(C_i \mid A_i)$, where $i \in \{1, 2, 3, 4, 5, 6\}$ (since there can only be 6 variables in each network).

The set of independence assertions for the influence structure in Definition 4.1 can be equivalently expressed as a distribution, P, over independence assumptions. More specifically, as in the case of Bayesian networks, a distribution P satisfies $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$ if and only if P can be represented as a set of CPDs with respect to $\mathcal{G}^{\mathbb{I}}$. In the next section we develop this idea to reveal interesting properties and recognisable difficulties which present themselves when learning influence networks from data.

4.2.2 *Independency Maps*

The compact representation of the influence network structure, $\mathcal{G}^{\mathbb{I}}$, exploits the underlying independence assumptions for the distribution modelled, P. We write these assumptions in the set $\mathcal{I}(P)$ in the form $(X \perp\!\!\!\perp Y \mid Z)$, where X, Y and Z are Bayesian networks which encode their respective distributions. This allows us to precisely state that $\mathcal{G}^{\mathbb{I}}$ is an I-map for the distribution P since it is satisfied by the local independence assumptions for the influence network structure $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$. We denote this as $\mathcal{I}(\mathcal{G}^{\mathbb{I}}) \subseteq \mathcal{I}(P)$.

The main intuition behind the idea of an I-map is that an I-map does not misguide us about the independencies in P. That is, every independence assumption that $\mathcal{G}^{\mathbb{I}}$ holds must also hold in P. However, P may have additional independence assumptions that do not hold in $\mathcal{G}^{\mathbb{I}}$.

4.2.3 *Factorisation of Influence Networks*

The compact factored representation of the influence structure relies on producing a subset of independence assumptions specified in P. Furthermore, this factored representation allows us to compute entries of the joint distribution by a product of factors over each Bayesian network structure, $\mathcal{G}^{\mathcal{B}}$, and the influence graph structure, $\mathcal{G}^{\mathbb{I}}$. The same can be said for any I-map for the distribution P. More formally,

Definition 4.2. Let $\mathcal{G}^{\mathbb{I}}$ be an influence network structure over the Bayesian networks $\mathbf{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_R\}$, where R is the number of networks. Each Bayesian network in $\mathbf{B}$ factorises according to $\mathcal{G}^{\mathbf{B}}$. We say that a distribution P, over the same space, factorises according to $\mathcal{G}^{\mathbb{I}}$ if P can be expressed as a product

$$P(\mathcal{B}^1, \ldots, \mathcal{B}^R) = \prod_{i=1}^{R} \prod_{j=1}^{N} P(X_j^{\mathcal{B}_i} | Pa_{X_j^{\mathcal{B}_i}}^{\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}}}),$$

where $X_j^{\mathcal{B}_i}$ is the j^{th} variable in $\mathcal{B}_i$, $\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}}$ is the union of the two structure independence assumption sets; and N is the number of variables for network $\mathcal{B}_i$.

This result is an extension of the chain rule for Bayesian networks with respect to the structure imposed by the influence network. We will see later in Chapter 5 that the structure imposed depends on the application of the influence network.

4.2.4 *Influence Networks*

We are now ready to define an influence network.

Definition 4.3. Suppose you have a set of Bayesian networks $\mathbf{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_R\}$, where each Bayesian network in $\mathbf{B}$ factorises according to $\mathcal{G}^{\mathbf{B}}$. Further assume that you have a set of independence assumptions between these Bayesian networks, $\mathcal{G}^{\mathbb{I}}$. Then an *Influence network* is a pair $\mathbb{I} = (\mathcal{G}, P_{\mathbb{I}})$, where $\mathcal{G} = \mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}}$, and $P_{\mathbb{I}}$ factorises over $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}})$.

In the next section we explore independency equivalence which suggests that recovering the influence structure is a difficult problem given the ambiguity of structures which can be deduced from training data.

4.2.5 *Independency Equivalence*

We have discussed influence networks as a specification of conditional independence assumptions, $\mathcal{I}(\mathcal{G}^{\mathcal{I}})$, associated with a graph $\mathcal{G}^{\mathcal{I}}$, for a distribution, P. The members from $\mathcal{I}(\mathcal{G}^{\mathcal{I}})$ provide a sufficient specification of the independence properties of P (provided that $\mathcal{I}(\mathcal{G}^{\mathcal{I}}) \subseteq \mathcal{I}(P)$). Thus we can ignore the graph structure and just consider $\mathcal{I}(\mathcal{G}^{\mathcal{I}})$ to reconstruct P.

This observation has an important consequence since different graph structures can be constructed from the same set of local independence assumptions. This means that different influence structures can be equivalent if they are constructed from exactly the same conditional independence assertions. More formally:

Definition 4.4. Two influence network structures $\mathcal{G}_1$ and $\mathcal{G}_2$ over a set of variables $\mathcal{X}$ are I-equivalent if $\mathcal{I}(\mathcal{G}_1) = \mathcal{I}(\mathcal{G}_2)$.

Definition 4.4 has an important implication that the set of all influence network structures (over a set of random variables $\mathcal{X}$) is partitioned into a complete set of mutually exclusive I-equivalence classes.

Figure 30 shows an example of three influence structures that encode the same independence assumptions. Each node represents a Bayesian network and an edge represents the flow of influence between each respective network. Note that for the case where $A \rightarrow B \leftarrow C$ (not shown in the figure), and we know that B is given/observed, then influence can flow from A to C (i.e. inter-causal reasoning). That is, in the case of two causes that have a joint effect (also known as a v-structure), influence can not flow between the two causes.

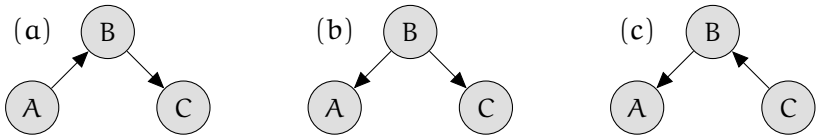


Figure 30: An example of an I-equivalent class encoded by the independence assertion: $(A \perp\!\!\!\perp C \mid B)$. Each node in the diagram represents a Bayesian network and edges represent the influence between them.

We will later see in Chapter 5 that I-equivalence plays an important role when selecting an influence network structure because of the following important restriction: There is no innate property of the distribution P which relates it to one network structure rather than another from the same I-equivalence class. Thus we can not decide on how to orient the direction of edges in an influence network.

More specifically, suppose we establish that two Bayesian networks $\mathcal{B}_1$ and $\mathcal{B}_2$ are correlated with respect to some set of observations. There is no information from the

observations that can determine whether the structure that gave raise to the ground truth was $\mathcal{B}_1 \to \mathcal{B}_2$ or $\mathcal{B}_1 \leftarrow \mathcal{B}_2$. We will revisit this essential implication when we discuss learning influence networks in Chapter 6. In the next section we will further develop this notion of influence for temporal models.

4.3 DYNAMIC INFLUENCE NETWORKS

An influence network specifies a joint distribution over a finite set of Bayesian networks. In many temporal domains, however, we require a probabilistic understanding of interactions between trajectories, which relate to a much more complex space then can be recorded as a finite set of Bayesian networks. Therefore, in temporal settings, we wish to represent a distribution of influence over systems whose states evolve over time. An influence state refers to the distribution of influence between a set of processes at a point in time.

Example 4.5 (Influence between proteins in a cell)**.** We may wish to model the relationship between quantities of proteins in a cell as its conditions change over time. In this setting we are given only feature information between several proteins at various times and we are expected to recover the inferred temporal influence relations between the proteins. The characteristic of each protein can be defined as observable and latent variables which is encoded by a local structure $\mathcal{G}^{\mathbf{B}}$ that persists through time. The structure of influence between each protein can be described as $\mathcal{G}^{\mathbb{I}}$ which also persists through time.

Example 4.6 (Influence between roads over time)**.** In road networks we may be given a set of observations for specific roads (e.g. light; number of cars, wind speed, temperature, number of collisions, e.t.c.) for which we may want to learn abstractions (e.g. traffic condition, weather, or the probability of an accident), these observable and latent features give us $\mathcal{G}^{\mathbf{B}}$ which persists though time. We then can infer influence between traffic activity on several roads by learning $\mathcal{G}^{\mathbb{I}}$ over time.

In Example 4.5 and 4.6 we may want to construct a single compact model that provides a template for the entire class of distributions from the same type, i.e. trajectories of traffic conditions or protein quantity over time.

In this section we extend the language of influence networks from the previous section to *dynamic influence networks* (DINs) which allow us to model influence between partially observed processes with an associated set of temporal observations. We begin by defining the context of temporal models as influence networks in Section 4.3.1; state the assumptions necessary for a compact and simplified representation in Section 4.3.2; and finally, delve into the definitions of DINs in Section 4.3.3.

4.3.1 *Context*

In a temporal setting we are mainly concerned with learning and reasoning about the influence states in a particular domain as it changes over time. Our main goal, in this section, is to describe influence between processes in a way which allows us to learn

the structure of influence networks effectively. Learning the structure of influence networks is addressed in Chapter 6, however, the way we define influence networks is critical to how we learn them.

The influence state, at time t, is modelled as an influence network which consists of several Bayesian networks (connected with a particular configuration - Chapter 5), such that, at a particular time we model the relationships between these Bayesian networks at time-point t as an influence network.

We assume that the influence state (at time t) is represented as an assignment of values (taken by random variables) to some set of Bayesian networks $\mathbf{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_R\}$. We denote $\mathcal{B}_i^{(t)}$ to be a Bayesian network $\mathcal{B}_i$ at time-point t. It is important to note that $\mathcal{B}_i^{(t)}$ is a template network whose structure may recur across time. The template network structure is initialised, at various time-points t, where each template network encodes the same set of factors and discrete values $Val(\mathcal{B}_i)$.

Our probability distribution between environments (from Section 4.2.4) is now a trajectory, that is, the distribution between environments is an assignment of network values for a particular time-point t. In this section we will present a DIN which is a joint distribution of influence over a set of processes.

Learning a joint distribution over processes presents a complex probability space. This complex probability space is discussed in the next section along with some simplifying assumptions to reduce its complexity.

4.3.2 *Assumptions*

In this section we present three assumptions used when learning DINs. The first is the discretisation of the continuous observations (Section 4.3.2.1); the second models future influence states as independent of the past given the present (Section 4.3.2.2); and finally, the third assumes that the system dynamics do not change (Section 4.3.2.3).

4.3.2.1 *Time Granularity*

Selecting a time-granularity, $\triangle$, to partition continuous observations of the system into time-slices at various intervals is essential to managing the space of observations. The selection of a time-granularity allows us to simplify the number of influence states that describe the trajectory to $t\triangle$, where t is the number of time-slices.

With this assumption we can (a) generalise aspects of the joint distribution that describes a trajectory over influence states, as well as, (b) simplify the distribution from a continuous system to one sampled from discrete intervals making the distribution more tractable. In the next subsection we discuss the Markov assumption.

4.3.2.2 *The Markov Assumption*

Suppose we have a distribution over a set of processes for $t = 0, \ldots, T$. That is $P(\mathbf{B}^{(0)}, \ldots, \mathbf{B}^{(T)}) = P(\mathbf{B}^{(0:T)})$ where $\mathbf{B}$ is a set of template Bayesian networks (as discussed in Section 4.3.1. Each $\mathbf{B}^{(t)}$ is referred to as an influence state). Then by the chain rule for probabilities (in the direction of time) we have that

$$P(\mathbf{B}^{(0:T)}) = P(\mathbf{B}^{(0)}) \prod_{t=0}^{T-1} P(\mathbf{B}^{(t+1)} | \mathbf{B}^{(0:t)}).$$

That is, the joint probability distribution over a set of processes is a product over all conditional distributions. This means that in order to calculate the probability of a single influence state we will need to calculate the product over all previous states. This is too expensive and discourages the use of lengthy processes.

Thus we would like to simplify the expression for trajectories using the Markov assumption (Section 2.2.2.1) in order to produce a more manageable distribution. The Markov assumption is a conditional independence assumption which models the next influence state as independent of past influence states given the present one. More formally,

Definition 4.7. A dynamic influence system over the template Bayesian networks, $\mathbf{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_R\}$, satisfies the Markov assumption if, for all $t \geqslant 0$,

$$(\mathbf{B}^{(t+1)} \perp\!\!\!\perp \mathbf{B}^{(0:(t-1))} | \mathbf{B}^{(t)}).$$

More specifically, independence in this case hold if and only if every variable in $\mathbf{B}^{(t+1)}$ is independent to every variable in $\mathbf{B}^{(0:(t-1))}$ given all variables from $\mathbf{B}^{(t)}$ (this is all relative to the arrangement of edges between these three Bayesian networks). The Markov assumption allows us to express that variables in the networks at influence states $\mathbf{B}^{(t+1)}$ are independent to influence states $\mathbf{B}^{(0:(t-1))}$ if we know the influence state $\mathbf{B}^{(t)}$. Thus we can now express the conditional distribution using the Markov assumption as

$$P(\mathbf{B}^{(0)}, \ldots, \mathbf{B}^{(T)}) = P(\mathbf{B}^{(0)}) \prod_{t=0}^{T-1} P(\mathbf{B}^{(t+1)} | \mathbf{B}^{(t)}).$$

The Markov assumption allows a compact representation of a joint distribution over a trajectory, however, the assumption might not always be practical in terms of the application. For example, there might be temporal dependencies which are left out due to being more than 2 influence states away from the any time t. In the case providing an all inclusive description of the world state will make the Markov assumption more reasonable [Koller and Friedman 2009].

In practical applications, where we are concerned with approximating a joint distribution, the Markov assumption is extensively used to simplify the distribution over trajectories. In influence networks, however, since the description of the world state depends on the temporal models which describe the processes, we rely on a rich

description of the state of the trajectory by the underlying temporal models for the assumption to be useful in the dynamic case of influence. In the next section we discuss the time-invariance assumption.

4.3.2.3 *The Time-Invariance Assumption*

The final simplifying assumption for dynamic influence systems is the time-invariance assumption where we assume that the system dynamics do not change. Since there is no limit to the length that a trajectory can take, it is useful for a template network to be used to describe influence states as t increases. More formally,

Definition 4.8. Suppose a dynamic influence system is Markovian. The dynamic influence system is said to be time invariant if $P(\mathbf{B}^{(t+1)} \mid \mathbf{B}^{(t)})$ is the same $\forall t$. In this case we can represent the process using a transition model $P(\mathbf{B}' \mid \mathbf{B})$, so that, for any $t \geqslant 0$,

$$P(\mathbf{B}^{(t+1)} = \xi' \mid \mathbf{B}^{(t)} = \xi) = P(\mathbf{B}' = \xi' \mid \mathbf{B} = \xi),$$

where ξ' is the next sample, ξ is the current sample, $\mathbf{B}'$ is the next Bayesian network, and $\mathbf{B}$ is the current Bayesian network.

With these simplifying assumptions we can now move on to define DINs in the next subsection.

4.3.3 *Dynamic Influence Networks*

Dynamic influence networks (DINs) are meant to represent dynamic influence between temporal models. The Markov (Section 4.3.2.2) and time-invariance (Section 4.3.2.3) assumptions allow us to model a joint distribution over a trajectory compactly since we only require a template influence network (Section 4.2.4) which recurs over time.

More specifically, we need to define a transition model, $P(\mathbb{I}^{(t)} \mid \mathbb{I}^{(t-1)})$, and initial state, $P(\mathbb{I}^{(0)})$. The initial influence network is simply the pair $\mathbb{I}^{(0)} = (\mathcal{G}, P_{\mathbb{I}})$, where $P_{\mathbb{I}}$ is a distribution which factorises over $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}})$ and $\mathcal{G}$ is any graph structure which is a perfect-map of $\mathcal{I}(\mathcal{G})$.

The transition model can be a 2-time-slice influence network. In a 2-time-slice influence network, the transition model can only have dependencies from the initial influence network. We refer to networks that are persistent through time as interface networks, denoted $\mathcal{B}_I$. Therefore, only interface networks can be parents of other network in the transition model. The observations are not interface variables. More formally, the transition model is a 2-time-slice influence network which is as follows:

Definition 4.9. A 2-time-slice influence network for a trajectory over a set of Bayesian networks $\mathbf{B}$ is a conditional influence network over $\mathbf{B}'$ given $\mathbf{B}_I$, where $\mathbf{B}_I \subset \mathbf{B}$ is a set of interface networks.

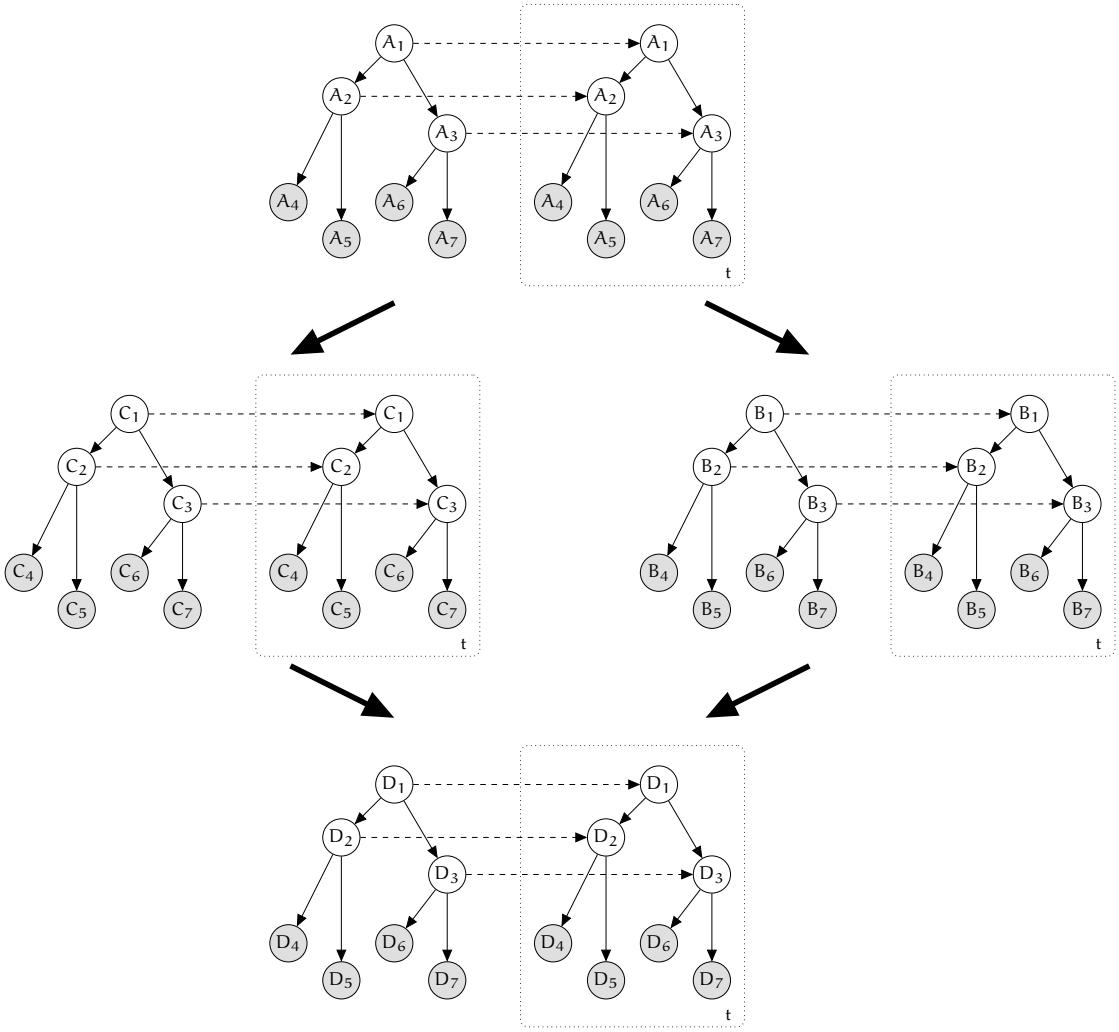


Figure 31: An illustration of a dynamic influence network between four hierarchical models. The initial influence states are omitted but follow the same structure as the first time-slice of the 2-time-slice influence networks in the image.

Figure 31 provides an example of an influence network between four temporal models. The unrolled networks are given in Figure 57 in Appendix A.

More specifically, the 2-time-slice influence network presents the following conditional distribution:

$$P(\mathbf{B}' \mid \mathbf{B}) = P(\mathbf{B}' \mid \mathbf{B}_I) = \prod_{i=1}^{R} P(\mathcal{B}_i' \mid Pa_{\mathcal{B}_i'}),$$

where R is the number of Bayesian networks.

Each template network, $\mathcal{B}_i$, contains a set of template variables $\mathcal{X}$ whose CPD values combine into a template factor, $P(X_j'|Pa_{X_j'})$ for all $j = \{1,,N\}$, where N is the number of variables in each Bayesian network. These factors are initialised many times as the influence network unrolls over time. The dependencies of these factors span from the previous influence state.

In a 2-time-slice influence network the inter-time-slice edges are between time-slices (dotted lines in Figure 31) and the intra-time-slice edges are between template

networks and their respective variables (thick and thin solid lines in Figure 31). Figure 31 also shows many examples of persistent variables and edges between variables in a DIN, which persist over time (Section 2.2.2.3).

Unrolling the transition model over any number of states allows us to define any trajectory size with the same dependency structure between variables (provided by the independency assumptions of each temporal networks internal structure for each process description).

The choice of how temporal networks relate, as well as the variables inside their time-slices, depends on how tightly coupled they are with respect to the data. That is, whether the exists strong statistical correlations between variables in the data. On the one hand, if the influence between networks is immediate (i.e shorter than the selected time granularity) then the effect between networks (and hence variables) will be indicated within the same time-slice using conditional dependence assumptions. On the other hand, if the influence between networks is gradual, then the effect between networks can be imposed between time-slices. We finally present the definition of a DIN.

Definition 4.10. A dynamic influence network (DIN) is a pair $\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle$, where $\mathbb{I}_0$ is a influence network over the set of Bayesian networks, $\mathbf{B}^{(0)} = \{\mathcal{B}_1, \ldots, \mathcal{B}_R\}$, representing the initial distribution and $\mathbb{I}_{\rightarrow}$ is a 2-time-slice influence network for the remainder of the influence process ($P(\mathbf{B}' \mid \mathbf{B}_I) = \prod_{i=1}^{R} P(\mathcal{B}'_i \mid Pa_{\mathcal{B}'_i})$). For any desired time span $T \geqslant 0$, the joint distribution over $\mathbf{B}^{(0:T)}$ is defined as an unrolled influence network, where, for any $i = 1, \ldots, n$:

- the structure and CPDs of $\mathcal{B}_i^{(0)}$ are the same as those for $\mathcal{B}_i$ in $\mathbb{I}_0$,
- the structure and CPD of $\mathcal{B}_i^{(t)}$ for $t > 0$ are the same as those for $\mathcal{B}'_i$ in $\mathbb{I}_{\rightarrow}$.

Thus, we can view a DIN as a compact representation of dynamic influence between processes from which we can generate an infinite set of influence states (one for every $T > 0$) by unrolling the DIN. On the one hand, since each processes is modelled using the same temporal structure, we have the convenience of using a single template structure for each processes since we don't have to specify different structures for each processes. On the other hand, we are in danger of using a non-representative temporal structure for a domain specified by the stochastic process.

The important difference between a DBN and DIN is that a DBN models a collection of variables over time, whereas DINs model groups of dynamic networks which describe processes. Each dynamic network that the DIN describes is made using a template dynamic network. Thus we assume that the description of all processes are the same. More specifically, we use the same features (ie. $\mathcal{X}$); dependencies between features (ie. $\mathcal{G}^{\mathbf{B}}$); and number of time-slices to describe each process.

In the next section we explore a simple algorithm to sample data from this dynamic model and finally analyse the difference between the DBN and DIN representations to model dynamic influence relations.

4.4 INFERENCE ON INFLUENCE NETWORKS

A DIN encodes independence assumptions and a distribution which we can generate samples from by using standard Bayesian sampling techniques. We can provide a good representation of the overall probability distribution of a DIN by using particle-based sampling methods. The simplest approach to generating samples is *forward sampling* (or ancestral sampling) [Murphy 1998], which we can use to generate samples, $\{\xi[1], ..., \xi[M]\}$, from the distribution $\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle$. That is, every ξ is a vector of values from every random variables in the DIN.

In Algorithm 2, we sample the nodes in the same order consistent with the partial order of the influence network, so that by the time we sample any node, we already have the values of its parents. Intuitively, we sample topologically such that for every directed edge $a \rightarrow b$, a will be sampled before b. This is since we need the value of a in order to index the correct parameter in b's corresponding factor.

Algorithm 2 Forward sampling in an influence network

procedure FORWARD-SAMPLE($\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow} \rangle$)
  Let $\mathbf{B} = \{\mathcal{B}_1, \ldots, \mathcal{B}_n\}$ be the topological ordering according to $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}})$.
  for all *timeslices* **do**
    for $i = 1, \ldots, n$ **do**
      Let $X_1, \ldots, X_k$ be the topological ordering according to $\mathcal{I}(\mathcal{B}_i)$
      for $j = 1, \ldots, k$ **do**
        $\mathbf{u}_k \leftarrow \mathbf{x}\langle Pa_{x_k} \rangle$ ▷ Assignment to Pa_{x_k} in $x_1, \ldots, x_{k-1}$
        Sample x_k from $P(X_k|\mathbf{u}_k)$
  return $(x_1, \ldots, x_k)$ $\forall$ $\mathcal{B}_i$;

Using basic convergence bounds (a method to select the correct choice of M), we know that from a set of samples, $\mathcal{D} = \{\xi[1], \ldots, \xi[M]\}$, generated by this sampling process, we can estimate the expectation of any function f as [Koller and Friedman 2009]:

$$\hat{\mathbf{E}}_{\mathcal{D}} = \frac{1}{M} \sum_{m=1}^{M} f(\xi[m]).$$

This convergence bound may serve as a useful guide to select the number of samples to train DINs.

Since a (dynamic) influence network is a Bayesian network, there are several inference algorithms available to the user of these influence networks. Such inference algorithms include the Sum-product message passing, where inference is performed by calculating the marginal distribution on each unobserved node conditioned on all observed nodes [Kschischang *et al.* 2001]; expectation propagation, which iteratively leverages the factorization structure of the target distribution [Minka 2001]; or variational inference, where one approximates probability densities through optimization

[Wainwright *et al.* 2008]. In the next section we will motivate empirically why knowing the influence network structure is better than modelling this problem of influence between processes using the general product rule.

4.5 IMPORTANCE OF INFLUENCE STRUCTURES

In this chapter we explored representing set of processes as a DIN. A DIN is defined as a structure ($\mathcal{G} = \mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}}$) and a distribution. $\mathcal{G}^{\mathbb{I}}$ is induced by a set of local independence assumption $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$. We now demonstrate that knowing $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$ enables us to learn the true dynamic influence structure better than structuring the processes using the general product rule [Schum 1994; Klugh 2013]. Learning $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$ is especially useful when learning for knowledge discovery since it explicitly outlines the flow of influence between temporal models.

Our definition of the DINs exploits the use of these independence assumptions by using it to decompose the joint distribution. We will use the general product rule as a baseline to demonstrate the effectiveness of knowing $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$ when recovering influence between temporal models.

Suppose we have a set of DBNs $\mathbf{D} = \{\langle \mathcal{B}_0^1, \mathcal{B}_\rightarrow^1 \rangle, \ldots, \langle \mathcal{B}_0^R, \mathcal{B}_\rightarrow^R \rangle\}$. We could model the joint distribution between temporal models using the general product rule. That is, $P(\mathbf{D}) = P(\langle \mathcal{B}_0^R, \mathcal{B}_\rightarrow^R \rangle | \langle \mathcal{B}_0^{R-1}, \mathcal{B}_\rightarrow^{R-1} \rangle, \ldots, \langle \mathcal{B}_0^1, \mathcal{B}_\rightarrow^1 \rangle) \times \ldots \times P(\langle \mathcal{B}_0^{R-1}, \mathcal{B}_\rightarrow^{R-1} \rangle | \langle \mathcal{B}_0^{R-2}, \mathcal{B}_\rightarrow^{R-2} \rangle, \ldots, \langle \mathcal{B}_0^1, \mathcal{B}_\rightarrow^1 \rangle)$. More generally,

$$P(\mathbf{D}) = P(\bigcap_{r=1}^{R} \langle \mathcal{B}_0^r, \mathcal{B}_\rightarrow^r \rangle) = \prod_{r=1}^{R} P(\langle \mathcal{B}_0^r, \mathcal{B}_\rightarrow^r \rangle \mid \bigcap_{j=1}^{r-1} \langle \mathcal{B}_0^j, \mathcal{B}_\rightarrow^j \rangle). \tag{14}$$

Decomposing the joint distribution using the general product rule allows us to consider simpler distributions by its factorization. This is perhaps the simplest case of attempting to capture an influence distribution between these temporal models.

In DINs we model the joint distribution ($P(\mathbf{D})$) as decomposed into a product of factors with respect to the independence assumptions presented in $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}})$. $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$ encodes the independence assumption which can take the form of several I-equivalent graph structures. We now compare the ability to recover a ground-truth distribution by using these two approaches to factorise the joint distribution between temporal models.

Figure 32 emphasises the importance of using DINs (blue line) with the true local independence assumptions, $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$, rather than modelling the processes using the general product rule in dynamic Bayesian networks (red line) as the number of samples increase. The y-axis shows the relative entropy to the ground truth distribution and the x-axis shows the increase in the number of samples.

In Figure 32, two dynamic influence parameter estimation tasks are shown. Each influence network models influence between 10 processes represented by HMMs with 5 time-slices. The form of the hidden Markov model is specified in Figure 15 and the

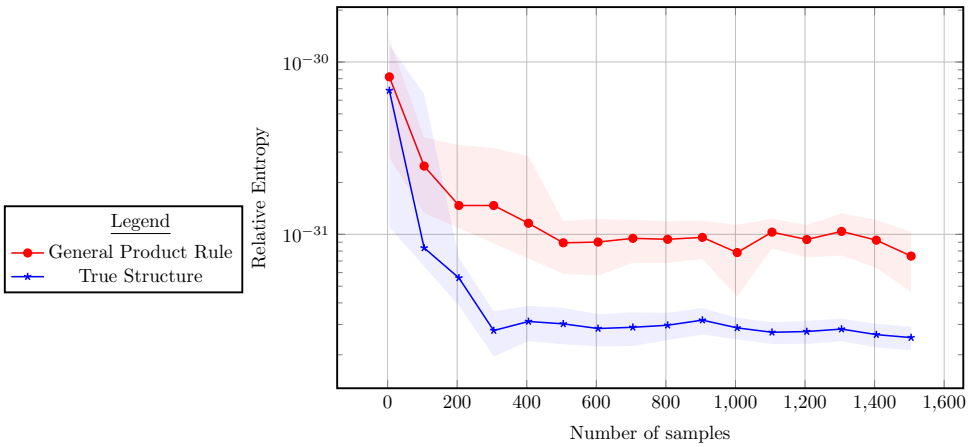


Figure 32: The average performance of modelling processes using the general product rule in a DBN (red) (p-value of 0.0059) and using a DIN (blue) (p-value of 0.0129) as the number of data samples increase. 10 processes were modelled using HMMs with 5 time-slices. The form of the hidden Markov model is specified in Figure 15 and the joint density function is given in Equation 5. The latent variables were learned using 50 EM iterations. The shaded regions represent the standard deviation as error bars over 10 trials.

joint density function is given in Equation 5. Latent variables were learned using 50 EM iterations (the EM algorithm is presented in Algorithm 1). The reason why the difference between the true structure and the ground truth distribution is because we are recovering the true parameters from the samples (indicated by the x-axis).

The first influence network was a standard DBN structure (red line) which modelled the HMMs using the general product rule as shown in Equation 14. The second influence network decomposed the joint distribution of all the HMMs using the independence assumptions presented in $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}})$. The generative ground truth distribution used an arbitrary graph structure induced by $\mathcal{I}(\mathcal{G})$. The shaded regions in Figure 32 represent the standard deviation as error bars over 10 trials. We notice that the variance shrinks as the numbers of samples grows for both influence modelling methods.

In Figure 32, we note that as the number of samples increase (generated from the ground-truth distribution using the sampling algorithm presented in Section 4.4), the DIN provides a smaller distance to the ground-truth distribution than modelling the set of variables using the general product rule. This is because of two reasons:

1. Using the local independence assumptions in $\mathcal{I}(\mathcal{G})$ allows us to learn the parameters of the ground-truth distribution, even if we do not know what the correct graph structure was (the one used by the ground-truth). Whereas, modelling the processes using the general product rule might restrict certain independence assumptions which limits our ability to learn the ground-truth distribution.

2. For a large number of processes, R, the general product rule offers a complete graph solution, since the first term will always be dependent on $R-1$ processes, the second on $R-2$ processes, and so on. Whereas, knowing $\mathcal{I}(\mathcal{G})$ offers a sparser structure which generalises better, translating to better performance for a small number of samples when compared to the general product rule.

We conclude this chapter by emphasising the importance of knowing the independence assumptions $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$ when learning influence relations. Thus, in Chapter 6, we attempt to reconstruct $\mathcal{I}(\mathcal{G}^{\mathbb{I}})$ by only observing the data. We also demonstrate that not only is using DINs useful to model the influence between processes for knowledge discovery, but it also allows for efficient learning procedures when recovering influence relations for density estimation due to the convenience of its representation.

In the next chapter we present a tool used to value DINs with respect to the data. We further present mechanisms to relate temporal models which are able to capture influence around richly structured spaces that consist of multiple processes which are interrelated in various ways. We continue to show that although it is possible to capture many types of influence in a single construction by using a setting of the parameters, complex influence constructions run into fragmentation issues.

5

STRUCTURE SCORES AND ASSEMBLES

5.1 INTRODUCTION

We attempt to track influence between stochastic processes which are represented as dynamic Bayesian networks (DBNs). Tracking influence between DBNs, using score-based structure learning, involves evaluating different structural configurations of influence which can be expressed by them. We have seen in the previous chapter that influence between DBNs can be expressed using dynamic influence network (DINs). Traditional scoring functions are equipped to evaluate structure over variables in a Bayesian network, and not structure between multiple unrolled temporal models as in a DIN. Evaluating the score between DINs involve the assessment of the data for each variables as the network unrolls over time. The machinery we provide in this chapter will enable us to measure the worth of a DIN which will assist in a pursuit to select an optimal one relative to the temporal training data.

Being able to evaluate the score of a DIN will enable us to select more appropriate networks relative to the training data by comparing scores between networks. In capturing the score of a network in the non-dynamic setting we can use scoring functions like the likelihood score (Section 3.2) and BIC score (Section 3.3) which ignore the temporal aspects of the data. However, in order to capture the score of a network in the dynamic setting we have to resort to approximating these measures over trajectories which are described by various DBNs. Aside from the scalability of structure scores to consider the temporal aspects of the data, we will see that another challenge arises when dealing with influence between dynamic models, like describing influence relations between them. We remedy this by introducing the notion of a structural assemble to encode influence relations between DBNs.

We make the following contributions in this chapter:

1. a formal generalisation of the log-likelihood score for influence and dynamic influence networks;
2. the notion of direct and delayed influence between temporal models;
3. the notion of a structural assemble to encode influence between sets of DBNs;
4. a generalisation of the BIC score for influence networks with respect to the proposed underlying structures (called the d-BIC);
5. proof of score decomposability in the proposed d-BIC score;
6. and empirical evidence showing the worth of using structural assembles to describe influence between multiple temporal models.

In Section 5.2, we explore structural scores in the context of influence models; and in Section 5.3, we declare structure scoring functions for influence networks using the notion of a *structural assemble* which assists in explaining influence between temporal models.

5.2 STRUCTURE SCORES

Recall in Section 3.6, that learning the structure of a Bayesian network can be viewed as an optimisation problem where we define a *search space*; a *structure score*; and a *search procedure*. The *search space* is a set of structural configurations; and the *search procedure* is an algorithm to traverse the search space in order to find a structure that maximises the *structure score*.

Evaluating the worth of a candidate structural configuration requires a mapping from the parameterisation of the Bayesian network to the likelihood of the training data. There have been several scores to enable such a mapping, these include the likelihood [Koller and Friedman 2009] and Bayesian information criterion (BIC) scores [Schwarz and others 1978]. However, in the context of influence structures in the temporal domain, which is the subject of this thesis, we are interested in measuring the capability of a structural configuration between temporal models to fit the training data. This poses the following major functional extension which scores that evaluate DINs need to adopt: A scoring function for DINs should be able to evaluate the score of a graphical structure between families of temporal models rather than families of random variables relative to the data.

This functional extension complicates traditional structure scoring functions since they need to be adjusted and extended to deal with this case. In Section 5.2.1 and Section 5.2.2, we explore and extend the capabilities of traditional scoring functions to handle this functional extension for DINs. Perhaps the simplest structure score is the likelihood score which relates the structure of a Bayesian network with the fit to data. We will begin by stating this score in the context of DINs.

5.2.1 *The Likelihood Score*

In this section we explore structural scoring functions for influence networks as defined in Chapter 4. The most natural way to approach this is from the perspective of measuring the likelihood of the structure relative to the training data. This section is structured as follows: Section 5.2.1.1 presents the simplest example of extending the traditional likelihood structure score between two naïve Bayes models; and then, Section 5.2.1.2 generalises the likelihood score for describing families of dynamic Bayesian networks.

5.2.1.1 *Scoring Influence Models*

Intuitively we would like to measure that if a relation that describes influence between two Bayesian networks is preferred by the data, then we would get more information about the general distribution from having this relation than without having it. Consider Figure 33 which illustrates two scenarios between two Bayesian networks, $\mathcal{B}_0 = (\mathcal{G}_{\mathcal{B}_0}, P_{\mathcal{B}_0})$ and $\mathcal{B}_1 = (\mathcal{G}_{\mathcal{B}_1}, P_{\mathcal{B}_1})$. $\mathcal{B}_0$ involves the variables $\{X_1, X_2, X_3\} \in \mathcal{X}$ and encodes the local independence assumption $\mathcal{I}(\mathcal{G}_{\mathcal{B}_0}) = \{X_2 \perp\!\!\!\perp X_3 | X_1\}$; while $\mathcal{B}_1$ involves the variables $\{Y_1, Y_2, Y_3\} \in \mathcal{Y}$ and encodes the local independence assumption $\mathcal{I}(\mathcal{G}_{\mathcal{B}_1}) = \{Y_2 \perp\!\!\!\perp Y_3 | Y_1\}$. As illustrated in Figure 33 the shaded variables X_2, X_3, Y_2, and Y_3 are observed and the non-shaded variables X_1 and Y_1 are latent.

Figure 33: Two influence structures between two naïve Bayes networks (Section 2.2.1.3). $\mathcal{B}_0$ involves the variables $\{X_1, X_2, X_3\} \in \mathcal{X}$ while $\mathcal{B}_1$ involves the variables $\{Y_1, Y_2, Y_3\} \in \mathcal{Y}$. The shaded variables are observable and the non-shaded variables are latent. The left scenario has no dependence between the Bayesian networks, and the right scenario encodes a dependence between the networks.

More formally, using Definition 4.3, each of these scenarios in Figure 33 can be described as an influence network, denoted $\mathbb{I}_0 = (\mathcal{G}_0, P_{\mathbb{I}_0})$ and $\mathbb{I}_1 = (\mathcal{G}_1, P_{\mathbb{I}_1})$ respectively, where $\mathcal{G}_i$ denotes the structure of the influence network $\mathbb{I}_i$ with probability distribution $P_{\mathbb{I}_i}$.

Intuitively, in scenario $\mathbb{I}_0$, both naïve Bayes structure $\mathcal{B}_0$ and $\mathcal{B}_1$ are independent. That is, the values of each of the models do not influence each other to any extent. in scenario $\mathbb{I}_1$, the values of the distribution in naïve Bayes model $\mathcal{B}_0$ can change the values of $\mathcal{B}_1$ given the dependency from X_1 to Y_1.

We assume that influence between networks, which may describe events, flows at a level of abstractions composed by observable variables. Thus the interaction of influence are not directly from the observations themselves (since the observations are dependent on the time granularity). Therefore we will adopt the hierarchical network structure described in Section 2.2.2.4 and represent influence as flowing between hierarchical models through latent variables.

Selecting an influence network, either $\mathbb{I}_0$ or $\mathbb{I}_1$, will require us to establish which structure, either $\mathcal{G}_0$ or $\mathcal{G}_1$, gives us a stronger likelihood to the data. Let us express the preferability of a particular structure more formally. The log-likelihood of $\mathcal{G}_0$ relative to the data can be expressed as:

$$\begin{aligned} \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = \sum_{m=1}^{M} (&\log \hat{\theta}_{x_1[m]} + \log \hat{\theta}_{x_2[m]|x_1[m]} + \log \hat{\theta}_{x_3[m]|x_1[m]} \\ &+ \log \hat{\theta}_{y_1[m]} + \log \hat{\theta}_{y_2[m]|y_1[m]} + \log \hat{\theta}_{y_3[m]|y_1[m]}), \end{aligned} \tag{15}$$

and the log-likelihood score of $\mathcal{G}_1$ relative to the data can be expressed as:

$$\begin{aligned}\text{score}_L(\mathcal{G}_1 : \mathcal{D}) = \sum_{m=1}^{M} (&\log \hat{\theta}_{x_1[m]} + \log \hat{\theta}_{x_2[m]|x_1[m]} + \log \hat{\theta}_{x_3[m]|x_1[m]} \\ &+ \log \hat{\theta}_{y_1[m]|x_1[m]} + \log \hat{\theta}_{y_2[m]|y_1[m]} + \log \hat{\theta}_{y_3[m]|y_1[m]}),\end{aligned} \tag{16}$$

where $\hat{\theta}_{x_i}$ is the maximum likelihood estimate for $P(x_i)$ and $\hat{\theta}_{y_j|x_i}$ is the maximum likelihood estimate for $P(y_j|x_i)$.

To intuitively express the trade-off of using one influence structure, between these naïve Bayes models, over the other, we would like to find which influence structure maximises the likelihood to the data. We can express this as the difference between the log-likelihood score of each model relative to the data as follows:

- if we have $\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) > 0$, then we would prefer the structure $\mathcal{G}_1$;
- if $\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) < 0$, then we would prefer the structure $\mathcal{G}_0$;
- finally, if $\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = 0$, then either structure will do since both give us the same likelihood relative to the data.

By subtracting Equation 15 from Equation 16 we can express the difference of computing the log-likelihood scores for either influence structure over the two naïve Bayes models as

$$\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = \sum_{m=1}^{M} (\log \hat{\theta}_{y_1[m]|x_1[m]} - \log \hat{\theta}_{y_1[m]}). \tag{17}$$

Recall the notion of a sufficient statistic from Definition 2.17. We can convert the summation, in Equation 17, to summing over values rather than over data instances. Thus we can represent each term by its respective sufficient statistic to obtain

$$\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = \sum_{x_1,y_1} M[x_1, y_1] \log \hat{\theta}_{y_1|x_1} - \sum_{y_1} M[y_1] \log \hat{\theta}_{y_1}. \tag{18}$$

The first summation in Equation 18 expresses the summation over all parameters of values $Val(Y_1)$ given $Val(X_1)$ multiplied by the number of times these values occur in the data. We can more clearly express this as an empirical distribution $\hat{P}(x_1, y_1)$ which is expressed in the training data $\mathcal{D}$. The sufficient statistic $M[x_1, y_1]$ is equal to the number of data instances multiplied by the empirical joint distribution, $M\hat{P}(x_1, y_1)$. Similarly we can state that $M[y_1] = M\hat{P}(y_1)$; $\hat{\theta}_{y_1|x_1} = \hat{P}(y_1|x_1)$; and $\hat{\theta}_{y_1} = \hat{P}(y_1)$.

If we express Equation 18 in terms of the empirical distribution then the difference in the score becomes

$$\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = \sum_{x_1,y_1} M\hat{P}(x_1, y_1) \log \hat{P}(y_1|x_1) - \sum_{y_1} M\hat{P}(y_1) \log \hat{P}(y_1). \tag{19}$$

Both summations in Equation 19 contain the number of samples M which is independent of the type of values found in the data and thus M can be pulled out of the summation.

Both summations could have been condensed into one if they were summed over the same values. We can artificially insert the sum over x_1 in the second summation of Equation 19 since $\sum_{x_1} \hat{P}(x_1, y_1) = \hat{P}(y_1)$. Thus pulling out M and artificially inserting a summation over x_1 we get

$$\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = M\Big(\sum_{x_1,y_1} \hat{P}(x_1, y_1) \log \hat{P}(y_1|x_1) - \sum_{x_1,y_1} \hat{P}(x_1, y_1) \log \hat{P}(y_1)\Big). \tag{20}$$

There are two more manipulations that we can exploit in Equation 20 to condense the difference of the scores further. Firstly, the term $\hat{P}(y_1|x_1)$ can be rewritten as $\frac{\hat{P}(x_1,y_1)}{\hat{P}(x_1)}$ using Bayes Law; and secondly, both summations in Equation 20 are of the same form and the term $\hat{P}(x_1, y_1)$ is common in each summation. Therefore, using the subtraction rule for logarithms we can condense the difference of the two scores as

$$\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = M \sum_{x_1,y_1} \hat{P}(x_1, y_1) \log \frac{\hat{P}(y_1, x_1)}{\hat{P}(y_1)\hat{P}(x_1)}. \tag{21}$$

The summation in Equation 21 is called the *mutual information* of $\mathcal{B}_0$ and $\mathcal{B}_1$ since it measures the average distance between the joint distribution, of $\mathcal{B}_0$ and $\mathcal{B}_1$, relative to if their distribution was a product of marginally independent models. We denote the mutual information of the two Bayesian networks as $\mathbf{I}_{\hat{P}}(\mathcal{B}_0; \mathcal{B}_1)$. More formally, we can express the difference between the two scores using the mutual information as:

$$\text{score}_L(\mathcal{G}_1 : \mathcal{D}) - \text{score}_L(\mathcal{G}_0 : \mathcal{D}) = M.\mathbf{I}_{\hat{P}}(\mathcal{B}_0; \mathcal{B}_1).$$

Note the difference between the traditional mutual information in Preposition 3.1 and the mutual information between Bayesian networks in Equation 21. The main difference is that the one is between variables and the other is between Bayesian networks. Practically, this difference allows us to evaluate influence networks (see Definition 4.3) which have a constrain on the edges described by $\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}}$. This constrain on the influence network lets us describe the flow of influence between models. By flow of influence, we are referring to the chain effect of influence between variables (see Section 2.2.1).

In this example the score over variables and Bayesian networks may seem fairly similar but depending on the number of latent variables in the model, and the structural properties which are expressed in the model (ie. $\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}}$), the mutual information over models aggregates structural correlations and similarities over all of the variables in each Bayesian network relative to the data.

5.2.1.2 *Scoring Dynamic Bayesian Networks*

We can now generalise this result for the log-likelihood score for DBNs. This small extension follows the same structural changes as in the previous section but spans the dynamic states as the Bayesian networks unroll with respect the Markov and time invariance assumptions (Section 2.2.2.1 and Section 2.2.2.2).

Proposition 5.1. Let T be the number of time-slices, then the log-likelihood score for a DBN decomposes as

$$\text{score}_L(\mathcal{G} : \mathcal{D}) = \sum_{t=0}^{T}(M \sum_{i=1}^{n} \mathbf{I}_{\hat{P}}(X_i^{(t)}; Pa_{X_i^{(t)}}^{\mathcal{G}}) - M \sum_{i=1}^{n} \mathbf{H}_{\hat{P}}(X_i^{(t)})),$$

where $\mathbf{H}_{\hat{P}}(X_i^{(t)})$ is the entropy of each variable X_i at time-slice t.

Proof. By the chain rule for Bayesian networks we can write the log-likelihood for DBNs as

$$\begin{aligned}
\text{score}_L(\mathcal{G} : \mathcal{D}) &= \sum_{t=0}^{T}(\sum_{i=1}^{n}(\frac{1}{M} \sum_{\mathbf{u}_i \in Val(Pa_{X_i^{(t)}}^{\mathcal{G}})} \sum_{x_i^{(t)}} M[x_i^{(t)}, \mathbf{u}_i] \log \hat{\theta}_{x_i^{(t)}|\mathbf{u}_i})), \\
&= \sum_{t=0}^{T}(\sum_{i=1}^{n}(\sum_{\mathbf{u}_i \in Val(Pa_{X_i^{(t)}}^{\mathcal{G}})} \sum_{x_i^{(t)}} \hat{P}(x_i^{(t)}, \mathbf{u}_i) \log \hat{P}(x_i^{(t)}|\mathbf{u}_i))), \\
&= \sum_{t=0}^{T}(\sum_{i=1}^{n}(\sum_{\mathbf{u}_i \in Val(Pa_{X_i^{(t)}}^{\mathcal{G}})} \sum_{x_i^{(t)}} \hat{P}(x_i^{(t)}, \mathbf{u}_i) \log(\frac{\hat{P}(x_i^{(t)}, \mathbf{u}_i)\hat{P}(x_i)}{\hat{P}(\mathbf{u}_i)\hat{P}(x_i)}))), \\
&= \sum_{t=0}^{T}(\sum_{i=1}^{n}(\sum_{\mathbf{u}_i \in Val(Pa_{X_i^{(t)}}^{\mathcal{G}})} \sum_{x_i^{(t)}} \hat{P}(x_i^{(t)}, \mathbf{u}_i) \log(\frac{\hat{P}(x_i^{(t)}, \mathbf{u}_i)}{\hat{P}(\mathbf{u}_i)\hat{P}(x_i)}) \\
&+ \sum_{x_i^{(t)}}(\sum_{\mathbf{u}_i \in Val(Pa_{X_i^{(t)}}^{\mathcal{G}})} \hat{P}(x_i^{(t)}, \mathbf{u}_i)) \log \hat{P}(x_i))), \\
&= \sum_{t=0}^{T}(\sum_{i=1}^{n}(\mathbf{I}_{\hat{P}}(X_i^{(t)}; Pa_{X_i^{(t)}}^{\mathcal{G}}) - \sum_{x_i^{(t)}} \hat{P}(x_i^{(t)}) \log \frac{1}{\hat{P}(x_i^{(t)})})), \\
&= \sum_{t=0}^{T}(\sum_{i=1}^{n}(\mathbf{I}_{\hat{P}}(X_i^{(t)}; Pa_{X_i^{(t)}}^{\mathcal{G}}) - \mathbf{H}_{\hat{P}}(X_i^{(t)}))).
\end{aligned}$$

□

We notice firstly that an entropy value per variable arises as $\mathbf{H}_{\hat{P}}(X_i^{(t)})$. Secondly, the log-likelihood score decomposes as a sum over family scores. That is, the mutual information of each variable, X_i, is based on a set of parent variables $Pa_{X_i^{(t)}}^{\mathcal{G}}$. The set, $Pa_{X_i^{(t)}}^{\mathcal{G}}$, is not explicitly defined in this context, neither is it defined in Figure 57 which shows an example of influence between unrolled DBNs. The selection of this parent

set is the basis of the discussion on assembles which we explore later in Section 5.3. For now we can assume that the parent set is derived from the internal structure of the DBN.

In the next section we continue to develop and analyse the log-likelihood score for temporal models by looking at scoring DINs. In particular we consider hierarchical dynamic Bayesian networks (HDBNs) to more explicitly express the aggregation over variables.

5.2.1.3 *Influence between Hierarchical Dynamic Bayesian Networks*

In the previous sections we investigated how influence between two Bayesian networks could be expressed using mutual information and how the structure score decomposes over DBNs. As an example consider Figure 57, which shows an influence model between four DBNs where we could use the score in Proposition 5.1 to calculate the log-likelihood of the DBN relative to some dataset.

Recall that we approach the goal of discovering dynamic influence between temporal models as an optimisation problem, where we define a scoring function that can evaluate each candidate DIN structure with respect to some dataset, $\mathcal{D}$. We then search for the highest scoring structure.

There have been many contributions to the establishment and development of scores alongside key properties. These include the likelihood score and the Bayesian information criterion score (BIC) [Schwarz and others 1978]. In this section we extend the likelihood score to one specifically geared to evaluating dynamic influence for DINs relative to temporal data, while being mindful of over-fitting. We further show that the derived score is decomposable for DINs.

We develop the likelihood score for temporal models by considering a hierarchical dynamic Bayesian network (HDBN). We intend that scoring functions and search procedures described here extend to all types of DBNs, however we consider an HDBN to be a special case of a DBN.

This discussion on HDBNs is inspired by Blei *et al.* [2003]; Blei and Lafferty [2006] who developed dynamic topic models. Dynamic topic models are the model of choice for document classification [Mcauliffe and Blei 2008]; semantic analysis on social media [Zhao *et al.* 2011]; and activity mining [Varadarajan *et al.* 2010].

Consider Figure 34 which presents an adaptation of a dynamic topic model, which we refer to as an HDBN, since we note that many layers can be added to this model to describe more high-level features.

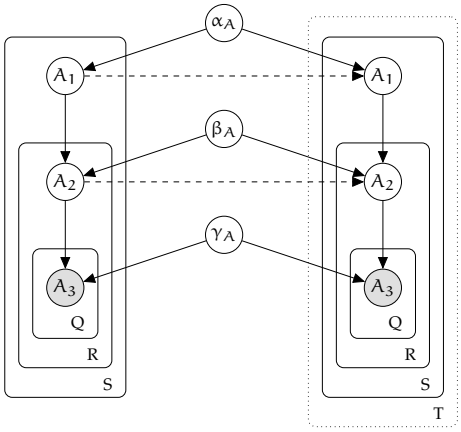


Figure 34: An illustration of a hierarchical dynamic Bayesian network (HDBN).

The model makes use of the following notation (refer to Chapter 2 for definition and descriptions):

- A_1, A_2 and A_3 are template variables;
- the parameters are explicitly indicated as the variables α_A, β_A and γ_A, and dependencies between parameters are indicated by solid lines;
- the template variable A_1 is a persistent latent variable, with a persistent edge between the initial state and the unrolled state encased in plate S;
- the template variable A_2 is a persistent latent variable, with a persistent edge between the initial state and the unrolled state encased in plate R;
- the template variable A_3 is an observable variable and encased in plate Q;
- persistent edges are indicated by dotted lines;
- the unrolled network is encased in a plate labelled T.

The joint density of the hierarchical dynamic Bayesian network (HDBN) graphical shown in Figure 34, denoted $\mathcal{H}$, is as follows:

$$P(\mathcal{H}) = P(A_3^{qrs}|A_2^{rs})P(A_2^{qrs}|A_1^{rs})P(A_1^{rs})P(A_3^{q'r's'}|A_2^{r's'})P(A_2^{q'r's'}|A_2^{qrs}A_1^{r's'})P(A_1^{r's'}|A_1^{rs})$$

The ability to consider a generalised and descriptive probability distribution between temporal models is a major advantage of the DIN representation (defined in Definition 4.10). Figure 35 expresses the DIN between four HDBNs using plate notation, dependencies between models are indicated by bold solid arrows.

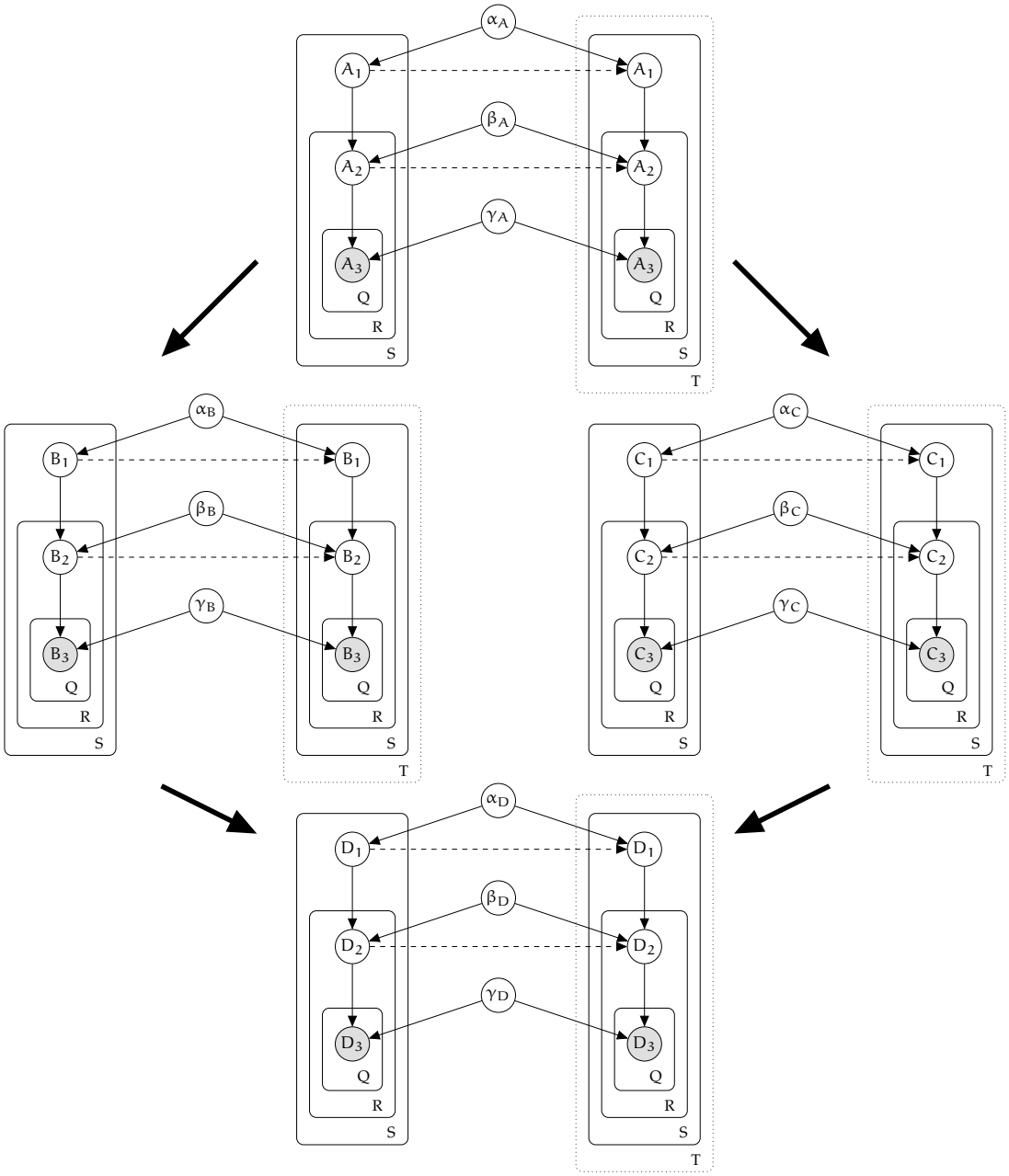


Figure 35: An illustration of a DIN between four HDBNs using plate notation. The thick lines in the figure correspond to the type of assemble related used discussed in Section 5.3.

Since a DIN is a dynamic Bayesian network with a restriction over its edges, we can compactly represent the joint distribution over HDBN networks (as shown in Figure 35) using conditional independence assumptions derived from the chain rule for Bayesian networks (Definition 2.2). This factorization simplifies the DIN between HDBNs significantly.

In order to search for an appropriate influence structure it is necessary to evaluate different DINs. Therefore, to extend these ideas for expressing general influence between multiple HDBNs we present the following proposition.

Proposition 5.2. The log-likelihood score of an influence structure $\mathcal{G}$ which consists of an HDBN, $\langle \mathcal{H}_0, \mathcal{H}_{\rightarrow} \rangle$, and a set of dependencies for $\langle \mathcal{H}_0, \mathcal{H}_{\rightarrow} \rangle$, $\mathbf{Pa} = \{\langle \mathcal{H}_1, \mathcal{H}_{\rightarrow} \rangle, \ldots, \langle \mathcal{H}_k, \mathcal{H}_{\rightarrow} \rangle\}$, decomposes as:

$$\text{score}_l(\mathcal{G} : \mathcal{D}^T) = M \sum_{k=1}^{K} \left(\sum_{t=1}^{T} \left(\sum_{i=1}^{N} \left(\mathbf{I}_{\hat{P}}(X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}} ; \mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}}) - \mathbf{H}_{\hat{P}}(X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}) \right) \right) \right),$$

where M is the number of instances in $\mathcal{D}^T$; K is the number of parent HDBNs, each HDBN is denoted as $\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle$; T is the number of time-slices in each HDBN; N is the number of variables in each HDBN's time-slice; $X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}$ is the variable X_i in model $\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle$ in time-slice t; and finally, $\mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}}$ is the set of all parent variables of X_i in model $\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle$ in time-slice t. The specification of $\mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}}$ will be discussed in Section 5.3.

Proof. Using the chain rule for Bayesian networks we get

$$\text{score}_l(\mathcal{G} : \mathcal{D}) = \sum_{k=1}^{K} \left(\sum_{t=1}^{T} \left(\sum_{i=1}^{N} \left(\sum_{m=1}^{M} \left(\log \hat{\theta}_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}[m] | \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}[m]} \right) \right) \right) \right)$$

By introducing sufficient statistics as parameter values we get

$$= \sum_{k=1}^{K} \left(\sum_{t=1}^{T} \left(\sum_{i=1}^{N} \left(\sum_{\mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}} \in Val(\mathbf{Pa}_{X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}})} \sum_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}} M[x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}, \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}] \right. \right. \right.$$

$$\left. \left. \left. \log \hat{\theta}_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}} | \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}} \right) \right) \right) \right)$$

By introducing the empirical distribution ($\hat{P}$) we get

$$= M \sum_{k=1}^{K} \left(\sum_{t=1}^{T} \left(\sum_{i=1}^{N} \left(\sum_{\mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}} \in Val(\mathbf{Pa}_{X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}})} \sum_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}} \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}, \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}) \right. \right. \right.$$

$$\left. \left. \left. \log \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}} | \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}) \right) \right) \right) \right)$$

Artificially inserting $\hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})$ to simplify the expression we get

$$= M \sum_{k=1}^{K} \left(\sum_{t=1}^{T} \left(\sum_{i=1}^{N} \left(\sum_{\mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}} \in Val(\mathbf{Pa}_{X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}})} \sum_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}} \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}, \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}) \right. \right. \right.$$

$$\left. \left. \left. \log \frac{\hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}, \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}) \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})}{\hat{P}(\mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}) \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})} \right) \right) \right) \right)$$

By separating terms into modelling the relative error and entropy of a temporal distribution we get

$$= M \sum_{k=1}^{K} (\sum_{t=1}^{T} (\sum_{i=1}^{N} (\sum_{\mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}} \in Val(\mathbf{Pa}_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}}) } \sum_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}} \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}, \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})$$

$$\log \frac{\hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}, \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})}{\hat{P}(\mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}) \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})}$$

$$+ \sum_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}} \sum_{\mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}} \in Val(\mathbf{Pa}_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}})} \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}, \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})$$

$$\log \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}))))$$

The first summation is called the mutual information (denoted $\mathbf{I}_{\hat{P}}$),

$$= M \sum_{k=1}^{K} (\sum_{t=1}^{T} (\sum_{i=1}^{N} (\mathbf{I}_{\hat{P}}(X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}; \mathbf{Pa}^{\mathcal{G}}_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}})$$

$$- \sum_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}} \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}) \log \frac{1}{\hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})})))$$

The second summation is called the entropy (denoted $\mathbf{H}_{\hat{P}}$) for each variable

$$= M \sum_{k=1}^{K} (\sum_{t=1}^{T} (\sum_{i=1}^{N} (\mathbf{I}_{\hat{P}}(X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}; \mathbf{Pa}^{\mathcal{G}}_{x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}}) - \mathbf{H}_{\hat{P}}(X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})))))$$

□

Now that we have developed a likelihood score for influence between HDBN families and so we can use this as a starting point to develop more sophisticated scores to evaluate DIN structures.

5.2.2 *The Dynamic Bayesian Information Criterion (d-BIC)*

Based on the preceding analysis, we see that the dynamic likelihood score measures the average distance between the joint distribution, $\hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}, \mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})$, relative to the product of marginals, $\hat{P}(\mathbf{Pa}_{x_i}^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}) \hat{P}(x_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}})$, which links the set $\mathcal{I}(\mathcal{G})$ to $\mathcal{D}$. However, unless two HDBNs are truly independent in $\mathcal{D}$, the dynamic likelihood score never prefers the simpler network over the more complicated one (i.e. one with more edges), since $\text{score}_l(\mathcal{G}_{\langle \mathcal{H}_0^1, \mathcal{H}_{\rightarrow}^1 \rangle \rightarrow \langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle} : \mathcal{D}) \geqslant \text{score}_l(\mathcal{G}_{\emptyset} : \mathcal{D})$. Which suggests that the dynamic likelihood score does not generalise well to instances from the true distribution (i.e P^*). This presents a significant over-fitting problem.

We therefore adopt an extension of the dynamic likelihood score called the dynamic Bayesian information criterion (d-BIC). The d-BIC score is a derivation of the

dynamic likelihood score that is biased to prefer simpler structures, but as it acquires more data it can prefer more complex structures to describe the distribution. In other words, it trades-off fit to data with model complexity, thereby reducing over-fitting. We therefore present the following extension of the dynamic likelihood score for influence networks in terms of the BIC score:

$$\text{score}_{\text{BIC}}(\mathcal{H}_0 : \mathcal{D}) = M \sum_{k=1}^{K} \Big(\sum_{t=1}^{T} \Big(\sum_{i=1}^{N} \big(\mathbf{I}_{\hat{P}} \big(X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}} ; \mathbf{Pa}_{X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle^{(t)}}}^{\mathcal{G}} \big) \big) \Big) - \frac{\log M}{c} \text{DIM}[\mathcal{G}], \tag{22}$$

where M is the number of samples; K is the number of dependency models; T is the number of time-slices for any dependency model; N is the number of variables in each time-slice; $\mathbf{I}_{\hat{P}}$ is the mutual information in terms of the empirical distribution defined in Preposition 5.2; and $\text{DIM}[\mathcal{G}]$ is the number of independent parameters in the entire influence network.

Equation 22 is simply the likelihood score in Preposition 5.2 with an added penalty term, and the following two observations:

- firstly the entropy term, $\mathbf{H}_{\hat{P}}$, does not influence the choice of structure and is ignored;
- secondly, the d-BIC score for HDBNs tends to trade-off the fit to $\mathcal{D}$ with model complexity. The mutual information term, $\mathbf{I}_{\hat{P}}$, grows linearly with the number of samples in $\mathcal{D}$, and the complexity term, $\frac{\log M}{c}\text{DIM}[\mathcal{G}]$, grows logarithmically with the size of $\mathcal{D}$.

Therefore, the larger the amount of data the more compelled the score will be to fit $\mathcal{D}$ and thus, with enough data, prefers the set $\mathcal{I}(\mathcal{G}^*)$. This property is called consistency.

In this section we introduced the dynamic likelihood and d-BIC score for DINs which intuitively weighted an influence network based on empirical correlations between temporal models which manifest in the data. Each variable in these temporal models was paired with a parent set whose members could span multiple temporal models in addition to variables in its own network. In the next section we introduce a mechanism to select this parent set.

5.3 STRUCTURE ASSEMBLES

The selection of the parent set for variables in a DIN introduces the notion of a structural assemble. A structural assemble is a configuration which connects temporal models and partly defines the parent sets for variables necessary to construct an influence network ('partly', since we may have variables as parents in the temporal model itself (ie. according to $\mathcal{G}^{\mathbf{B}}$) and in external models (ie. according to $\mathcal{G}^{\mathbb{I}}$)).

Consider Figure 36, which unrolls two HDBNs, $\langle \mathcal{A}_0, \mathcal{A}_\to \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_\to \rangle$, as represented in Figure 34 with $Q = 2$ (number of models); $R = 2$ (Bernoulli distribution per variable); and $T = 3$ (time-slices). One way to represent a structural assemble for the dependency assertion $P(\langle \mathcal{B}_0, \mathcal{B}_\to \rangle \mid \langle \mathcal{A}_0, \mathcal{A}_\to \rangle)$ is to connect every latent variable in $\langle \mathcal{A}_0, \mathcal{A}_\to \rangle$ to all latent variables in $\langle \mathcal{B}_0, \mathcal{B}_\to \rangle$. We call this the mesh assemble. The mesh assemble is a suitable representation mainly for two reasons:

1. we can express any parameterisation to capture the joint distribution given the knowledge about the model dependency;

2. and the edges between time-slices make us indifferent about the application of the model (ie. if influence moves quickly or slowly across time) since it precisely explains arbitrary interrelations between various time-slices.

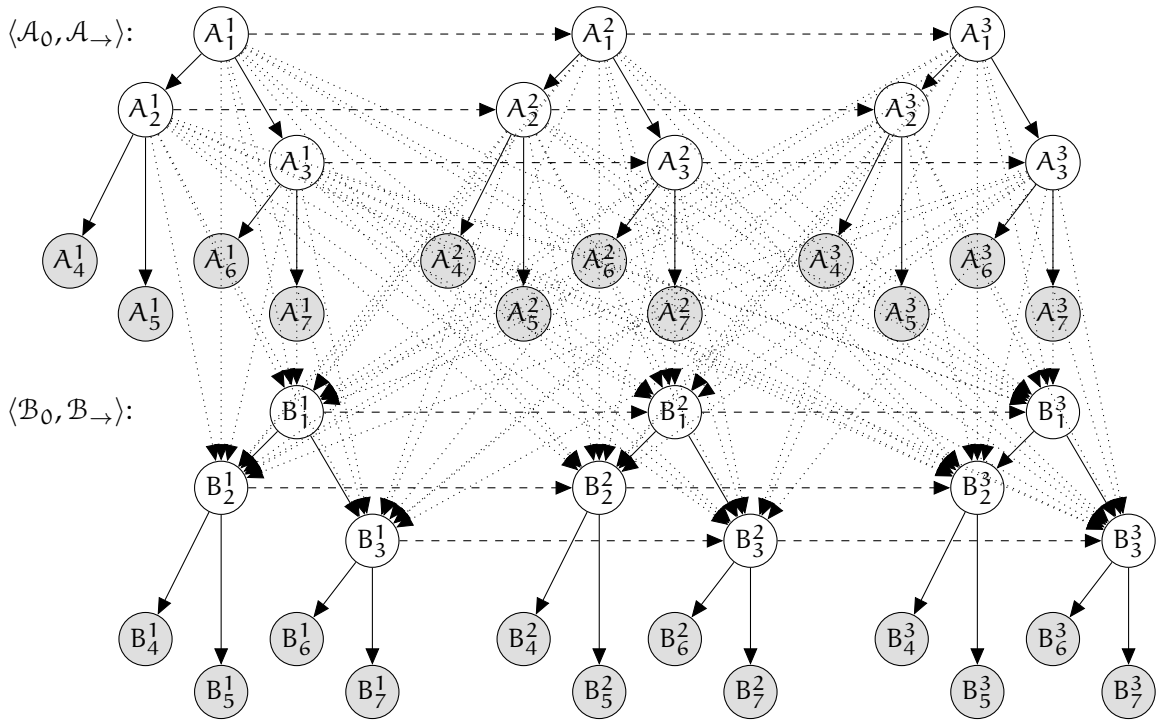


Figure 36: Two unrolled HDBNs, $\langle \mathcal{A}_0, \mathcal{A}_\to \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_\to \rangle$, as represented in Figure 34 with $Q = 2$; $R = 2$; and $T = 3$. The temporal models are connected with a mesh assemble.

The mesh assemble provides a complete graphical network with an expensive representation. Even taking advantage of the various independence assumptions within

the dynamic Bayesian network, we will be required to learn over 9 000 parameters[1] using table CPD factors just to describe a Bernoulli distribution between the two temporal models in Figure 36.

Of course, in practical application where we require DINs structures (such as for those in Example 4.5 and 4.6) we will require many dependencies per model which will increase the number of parameters super-exponentially, quickly making the representation unmanageable and parameterisation of this assemble intractable. Furthermore, this network may pose redundant interconnections between nodes since temporal processes move forward with time but some of these edges backtrack.

In this section we explore the space and capabilities of several alternative structural assembles which can be broken up into two main subgroups: direct (Section 5.3.1) and delayed (Section 5.3.2) assembles. Each of these assembles advocate various intuitive suitabilities for influence networks in different applications.

5.3.1 *The Direct Assemble Subgroup*

Direct influence between two stochastic processes, denoted $A_{\rightarrow}$ and $B_{\rightarrow}$, is defined by a *proportional* relationship between them. By proportional we mean that if we change a value at the time t_1 in $A_{\rightarrow}$ this will affect the value at exactly time t_1 in $B_{\rightarrow}$.

As a simple example of direct influence consider the illustration in Figure 37 of a pie chart representing five stochastic processes evolving by influencing each other over time. Each slice in the pie chart, denoted by the letter 'A' to 'E', represents the presence of a stochastic processes $A_{\rightarrow}$ to $E_{\rightarrow}$.

Although each stochastic process can be described by several features. In this example, we only wish to illustrate the extent to which processes can influence each other. We will explore richer temporal representations of processes later in this subsection.

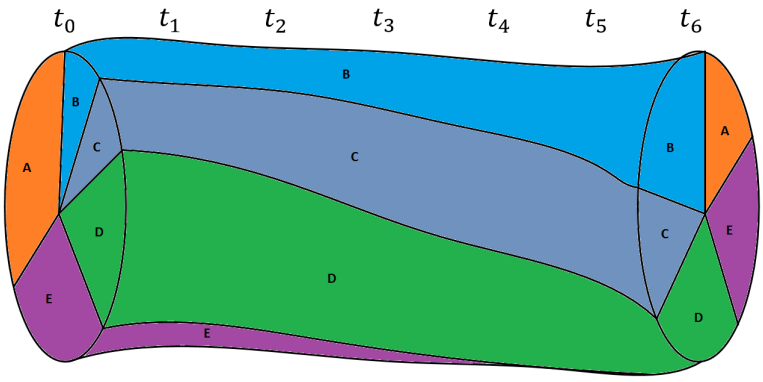


Figure 37: A simple example of direct influence between processes represented as a slice in a pie chart which represents the quantity of the processes as it evolves over time.

1 78 parameters for $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ (30 latent parameters and 48 observable parameters); and 9 264 parameters for $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$ (9 216 latent parameters and 48 observable parameters). Therefore, a total of 9 342 parameters.

Figure 37 shows that as the time elapses the presence of process $B_{\rightarrow}$ and $C_{\rightarrow}$ increases which influences process $A_{\rightarrow}$ to decrease, where process $A_{\rightarrow}$ began with a large presence in t_0. Processes $D_{\rightarrow}$ and $E_{\rightarrow}$ appear to be less influenced by the growth of $B_{\rightarrow}$ and $C_{\rightarrow}$, and shrinkage of process $A_{\rightarrow}$.

Perhaps the most obvious and intuitive way to arrange a direct assemble, using independence assertions between two HDBNs, is by adding a directed edge from every latent variable in $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ to every latent variable in $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, if we know that influence flows in this way. This is illustrated in Figure 38 which unrolls two HDBNs, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, using the direct assemble with $Q = 2$; $R = 2$; and $T = 3$.

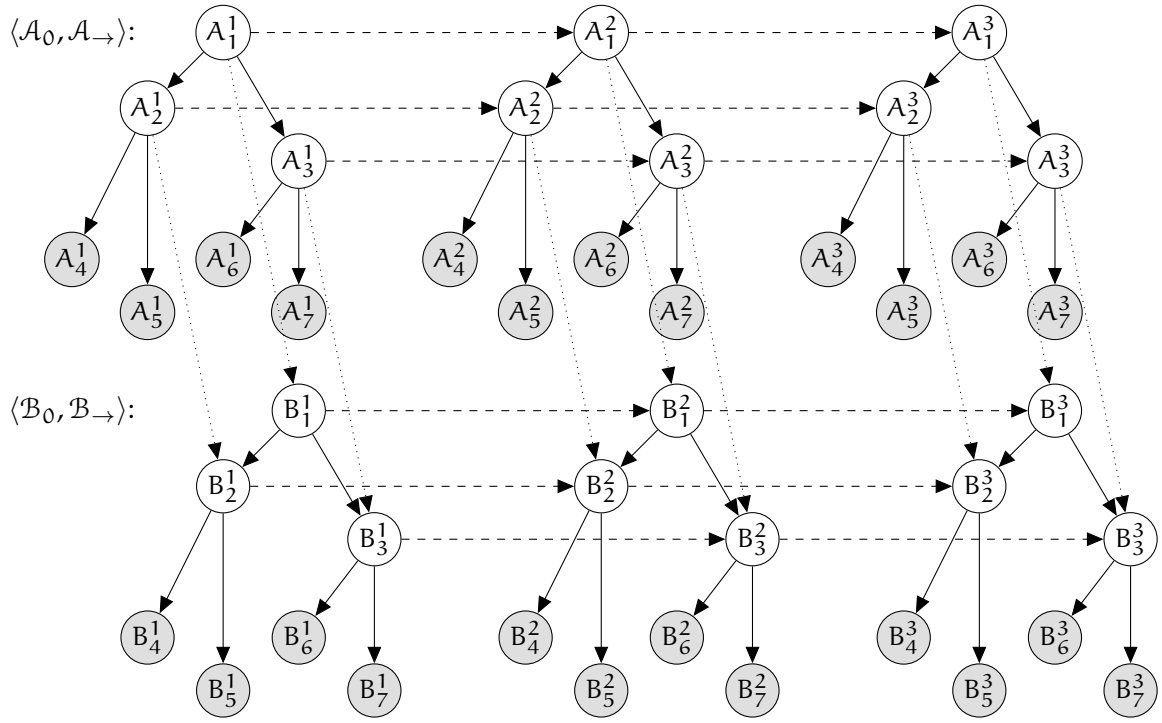


Figure 38: A direct dynamic influence network (DiDIN) modelling two unrolled HDBNs, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, as represented in Figure 34 with $Q = 2$; $R = 2$; and $T = 3$. The temporal models are connected with a direct assemble. Since the observations are relatively instantaneous (tightly coupled in the data) compared to our time granularity we represent them as intra-time-slices represented as solid lines; the persistent inter-time-slice edges are given by the broken lines; and finally, edges induced by the assemble are given by the dotted lines.

More generally, we now provide a definition of the direct influence assemble for a family of temporal models in Definition 5.3.

Definition 5.3. Consider a family of HDBNs, where $\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle$ represents the child with the parent set $\mathbf{Pa}^{\mathcal{G}}_{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle} = \{\langle \mathcal{H}_0^1, \mathcal{H}_{\rightarrow}^1 \rangle, \ldots, \langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle\}$. Further assume that $\mathcal{I}(\langle \mathcal{H}_0^j, \mathcal{H}_{\rightarrow}^j \rangle)$ is the same for all $j = 0, \ldots, k$. Then the *direct* dynamic influence network (DiDIN), denoted as a DIN $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$, satisfies all the independence assumptions $\mathcal{I}(\langle \mathcal{H}_0^i, \mathcal{H}_{\rightarrow}^i \rangle)$ $\forall\, i = 0, \ldots, k$. In addition, $\forall\, j$ and $\forall\, t$, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle^{(t)}$ also satisfies the following independence assumptions for each latent variable denoted L_i:

$$\forall\, L_i^{\langle \mathcal{H}_0^0, \mathcal{H}_\rightarrow^0 \rangle^{(t)}} : (L_i^{\langle \mathcal{H}_0^0, \mathcal{H}_\rightarrow^0 \rangle^{(t)}} \perp\!\!\!\perp \text{NonDescendants}_{L_i^{\langle \mathcal{H}_0^0, \mathcal{H}_\rightarrow^0 \rangle^{(t)}}} | L_i^{\langle \mathcal{H}_0^k, \mathcal{H}_\rightarrow^k \rangle^{(t)}}, Pa_{L_i}^{\langle \mathcal{H}_0^0, \mathcal{H}_\rightarrow^0 \rangle^{(t)}}).$$

We note that $\langle \mathcal{A}_0, \mathcal{A}_\rightarrow \rangle^{(t)}$ denotes the influence network at time-slice t. Definition 5.3 describes direct influence between families of dynamic Bayesian networks. The main benefits of the described direct influence assemble are the following:

1. unlike in the mesh assemble, the direct assemble provides model sparsity which means learning a much smaller set of parameters per model;
2. sparsity in the model also empirically provides better generalisation for density estimation [Koller and Friedman 2009; Krishnapuram *et al.* 2005; Friedman *et al.* 1999];
3. the assemble provides a representation for applications which require a direct influence between processes (since it ignores most inter-time-slice influences).

Implementing the direct influence assemble (defined in Definition 5.3) requires the partition of several conditional independence assumptions between variables in the DIN. This partitioning leads to the notion of internal and external dependencies with respect to a model $\langle \mathcal{H}_0^0, \mathcal{H}_\rightarrow^0 \rangle$ and its parent set $\mathbf{Pa}^{\mathcal{G}}_{\langle \mathcal{H}_0^0, \mathcal{H}_\rightarrow^0 \rangle} = \{\langle \mathcal{H}_0^1, \mathcal{H}_\rightarrow^1 \rangle, \ldots, \langle \mathcal{H}_0^k, \mathcal{H}_\rightarrow^k \rangle\}$.

Using a decomposable score function defined in Section 5.2, we can evaluate the complete structure score by computing the sum of family scores in a model. However, calculating even a single family score requires its compact compilation of *conditional independence assumptions* and efficient construction of its *parametrisation* for each involved factor. We describe these activities as the following:

COMPACT INDEPENDENCE ASSUMPTIONS: Our goal is to decompose the structure of a DIN with respect to $\mathcal{G} = \mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}}$. Each family of variables in a DIN is constructed with an assemble (direct or delayed encoded as $\mathcal{G}^{\mathbb{I}}$) which requires a specification of independence assumptions for that family. Dependencies for variables can span multiple models and (in the case of delayed influence) multiple time-slices. We identify those dependencies which exist within the model to be *internal dependencies* (ie. those of $\mathcal{G}^{\mathbf{B}}$), and those which span external models or time-slices to be *external dependencies* (ie. those of $\mathcal{G}^{\mathbb{I}}$). In order to determine the configuration of dependencies for a family within a network, one needs to partition the families of the model by the internal and external dependencies.

EFFICIENT PARAMETRISATION: Alongside the compact specification of conditional independence assumption of each family in the model, we need to learn the parametrisation of each factor with respect to the data. This can be done using Bayesian estimation as described in Section 2.3.1.2 with corresponding priors.

Consider Algorithm 3 which attempts to evaluate a family score (ie. score of a variable with respect to its parents) of a network using any decomposable score found in Section 5.2.

On Line 1, we declare that the procedure takes in a family which consists of an HDBN, $\langle \mathcal{H}_0^0, \mathcal{H}_\rightarrow^0 \rangle$, and the parent set for this model, denoted $\mathbf{Pa}^{\mathcal{G}}_{\langle \mathcal{H}_0^0, \mathcal{H}_\rightarrow^0 \rangle}$. On Line

2 we initialise the real value score to zero since we will accumulate the score as the algorithm proceeds. On Line 4 and Line 5 we traverse all of the variables in every time-slice of model $\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle$. For each variable we consider whether it is latent (Line 6) or observable (Line 13).

There are several scenarios that we need to explore. On the one hand, if the variable is latent then we know that the direct assemble has affected the variable and we may have an external dependency, however we may not have an internal dependencies if the variable is at the highest level of the hierarchy (Line 11). On the other hand, the variable may be observable, where it has to have an internal dependency, given the definition of an HDBN (Line 14), and there will not be any external dependency for this case. However, we reserve the case for general dynamic models where we may not have a dependency for observable variables (Line 18). Line 19 to 26 handle the remaining case where $\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle$ has no model dependencies to begin with.

Once we have identified each of these cases, we construct $\mathbf{Pa}^{\mathcal{G}}_{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle}$ to be a conjunction of the external and internal dependencies. This constructed parent set is used to score the family relative to the data as scores are tallied at each iteration of the for loop. Algorithm 3 provides a much more detailed implementation of the direct assemble construction.

Algorithm 3 The direct influence assemble

1: **procedure** FAMILYSCORE($\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle, \mathbf{Pa}^{\mathcal{G}}_{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle}$)
2: score = 0
3: **if** *!isEmpty*($\mathbf{Pa}^{\mathcal{G}}_{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle}$) **then**
4: **for** each timeslice, t, in $\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle$ **do**
5: **for** each variable, X_i, in $\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}$ **do**
6: **if** *isLatent*($X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}$) **then**
7: $\mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}} = \{\}$ ▷ No members in parent set
8: **if** *hasIntDep*($X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}$) **then**
9: $\mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}} = \{IntDep(X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}), ExtDep(X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}})\}$
10: **else**
11: $\mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}} = \{ExtDep(X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}})\}$
12: score += **score**($X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}, \mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}}$)
13: **else if** *isObs*($X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}$) **then**
14: **if** *hasIntDep*($X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}$) **then**
15: $\mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}} = \{IntDep(X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}})\}$
16: score += **score**($X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}, \mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}}$)
17: **else**
18: score += **score**($X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}, \{\}$)
19: **else**
20: **for** each timeslice, t, in $\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle$ **do**

21: **for** each variable, X_i, in $\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}$ **do**
22: **if** *hasIntDep*$(X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}})$ **then**
23: $\mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}} = \{IntDep(X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}})\}$
24: score += **score**$(X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}, \mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}})$
25: **else**
26: score += **score**$(X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}, \{\})$

THE PENALTY TERM: The penalty term, the second part of the summation in Equation 22, is expressed using the dimension (see Section 3.3) of the influence model. We can also calculate this component using decomposability since we are counting the number of independent parameters in the model. This can be practically achieved by replacing Lines 12, 16, 18, 24, and 26 in Algorithm 3 with Equation 23.

$$\mathbf{DIM}[\mathcal{G}] += \mathbf{DIM}[X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}, \mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^0, \mathcal{H}_{\rightarrow}^0 \rangle^{(t)}}}]. \tag{23}$$

The procedure for calculating the model dimension is provided in Algorithm 4. Notice that although the algorithm takes a family as input (ie. $X, \mathbf{Pa}_X^{\mathcal{G}}$), it only returns the number of independent parameters for a variable X and not for the whole family.

Algorithm 4 Model Dimension

1: **procedure** DIM($X, \mathbf{Pa}_X^{\mathcal{G}}$)
2: $d = \text{NumCPDs}(X) - 1$;
3: **for** all Y_i in $\mathbf{Pa}_X^{\mathcal{G}}$ **do**
4: $d \mathrel{*}= \text{NumCPDs}(Y_i) - 1$;
return d;

COMPLEXITY ANALYSIS: We outline the *rough* computational analysis of scoring a DIN relative to the provided training data. In order to provide an intuition for our analysis, suppose we have the DIN illustrated in Figure 39, where each node represents a HDBN. Suppose that all the parameters in each model have been learned and we need only evaluate its score. We can traverse the HDBNs in the DIN in any order and evaluate the score of each family of nodes using Algorithm 3. That is, for every model H; for every time-slice T in a model H; and for every variable X (in a time-slice T in a model H) we arrange a conjunction with its parent set (using Algorithm 3), which consists of I internal variables according to $\mathcal{G}^{\mathbf{B}}$ and 1 external variable according to $\mathcal{G}^{\mathbb{I}}$ and Definition 5.3, to form a family. For each family we compute the sufficient statistics by scanning though instances in the data $\mathcal{D} = \{\xi_1, \ldots, \xi_M\}$, which takes M units of time. We use the parameters to compute the score. A score with a penalty term, such as the d-BIC score, will use the same procedure to calculate the model dimension. Therefore asymptotically, our grand total time complexity is O(HTXIM).

In summary, the direct assemble learns influence in a straightforward implementation between temporal models and provides a sparse representation for DINs. Although the computational complexity seems expensive, the direct assemble provides

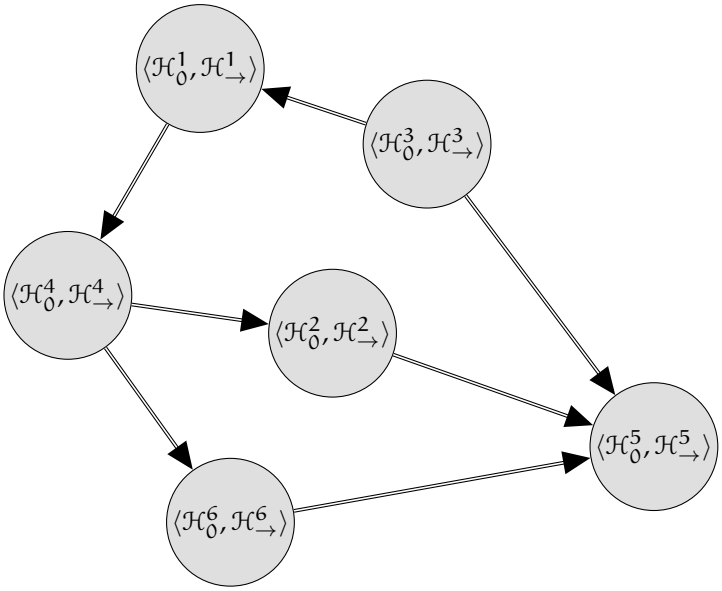


Figure 39: A DIN whose nodes represent six HDBNs. Each HDBN is denoted $\langle \mathcal{H}_0^i, \mathcal{H}_\rightarrow^i \rangle$, where $\mathcal{H}_0^i$ is the initial network and $\mathcal{H}_\rightarrow^i$ is the unrolled network. The double edges between each network represent the direct assemble used.

an intuitive representation of the expected flow of direct influence between various time-slices.

We now generalise the direct influence assemble for applications that require our model to explain influence between time-slices, since there can be a delay in the communication of information between variables. We call this phenomenon delayed influence where the choice of time granularity (Section 4.3.2) provides a trade-off between generalised and circumstantial applications of influence.

5.3.2 *The Delayed Assemble Subgroup*

Delayed influence between two stochastic processes, $A_\rightarrow$ and $B_\rightarrow$, is defined in contrast to direct influence where a change of a variable in process $A_\rightarrow$ at time t_1 will only result in a change in the variables at or after t_1 in $B_\rightarrow$.

As a simple example of delayed influence consider the illustration in Figure 40 which shows a modified map of Braamfontein outside the University of the Witwatersrand. Many motorists traverse the roads of Braamfontein from the star on Jorissen St to the Wits Theatre Complex on Bertha St. They can traverse the roads in Braamfontein either via De Korte St or Juta St.

Suppose that many motorists were to cross over from Jorissen St to the Wits Theatre Complex. Some motorists would travel via De Korte St and some would use Juta St creating a distribution of road usage which will translate to a particular traffic condition on each road in the network. Now suppose that there is an accident on De Korte St at time t_1 which causes bumper-to-bumper congestion resulting in no motorists being able to cross De Korte St. The influence of this event will force the

other motorists to use Juta St at time t_2 which will increase the number of cars on Juta thereby increasing congestion on Juta St.

In this example there was a delay from the time of the accident, t_1, to the time the conditions of De Korte St influenced the condition on Juta St, t_2. If we depict temporal models to describe the events of traffic conditions of each road over time we could not use the direct assemble since it ignores dependencies which could span multiple time-slices over multiple models. The need now arises for assembles that capture rich structure between time-slices from various temporal models.

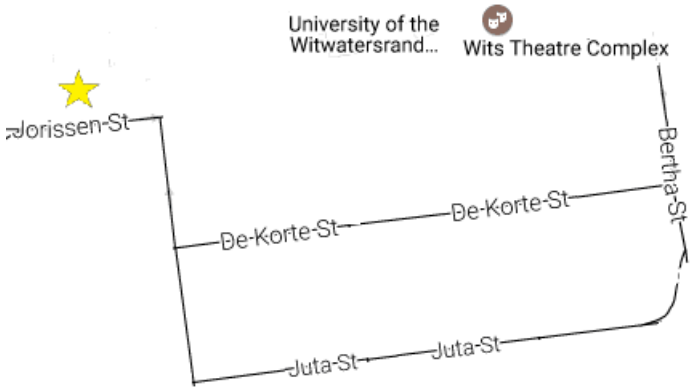


Figure 40: A modified map depicting a network of roads in Braamfontein outside the University of the Witwatersrand.

To intuitively capture this influence structure between temporal models we need to insert dependencies between different time-points that span various models. How far back the dependency between time-slices go depends on the influence structure of the distribution. Figure 41 depicts an example of delayed influence where we insert dependencies between two unrolled HDBNs, that is, from $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ to $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, where $Q = 2$; $R = 2$; and $T = 3$. In this example of influence, each time slice of $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$ has dependencies from the corresponding and previous time-slice in $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$.

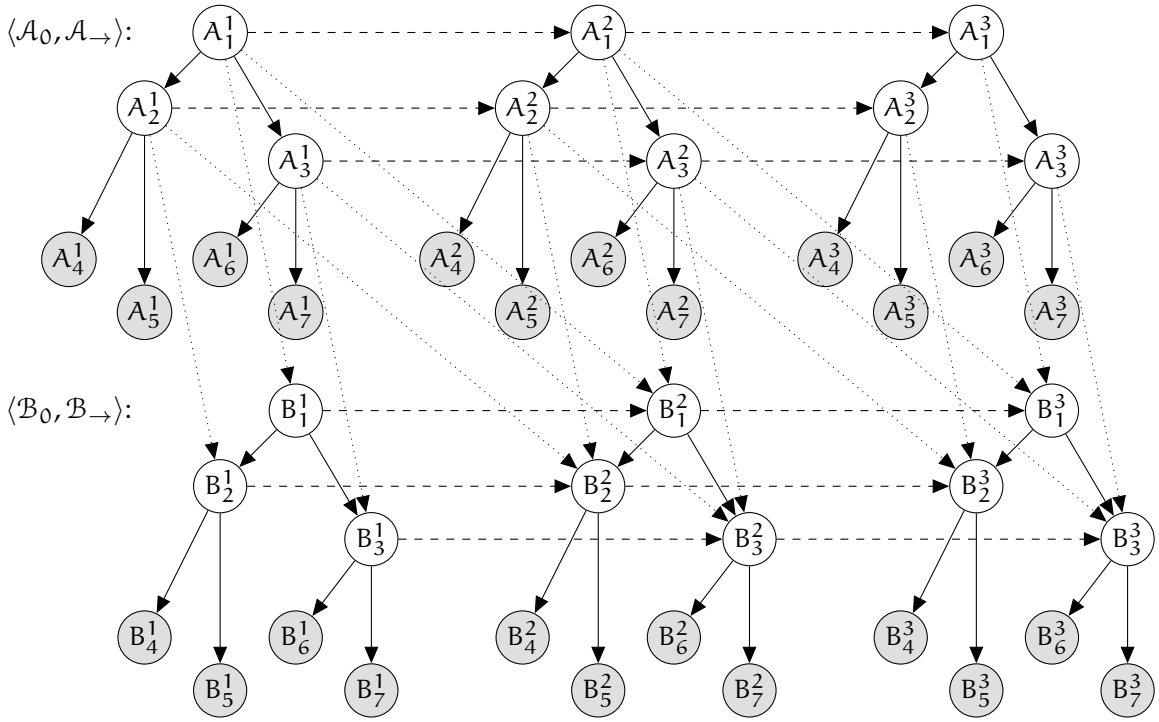


Figure 41: Two unrolled HDBNs, $\langle \mathcal{A}_0, \mathcal{A}_\to \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_\to \rangle$, as represented in Figure 34 with $Q = 2$; $R = 2$; and $T = 3$. The temporal models are connected with a delayed assemble. Since the observations are relatively instantaneous compared to our time granularity we represent them as intra-time-slices represented as solid lines; the persistent inter-time-slice edges are given by the broken lines; and finally, edges induced by the assemble are given by the dotted lines.

In Figure 41, we captured delayed influence by inserting dependencies between previous time-slices. More generally in Definition 5.4, we describe delayed influence with respect to α many previous time-slices for a family of temporal models.

Definition 5.4. Consider a family of HDBNs, where $\langle \mathcal{H}_0^0, \mathcal{H}_\to^0 \rangle$ represents the child with the parent set $\mathbf{Pa}^{\mathcal{G}}_{\langle \mathcal{H}_0^0, \mathcal{H}_\to^0 \rangle} = \{\langle \mathcal{H}_0^1, \mathcal{H}_\to^1 \rangle, \ldots, \langle \mathcal{H}_0^k, \mathcal{H}_\to^k \rangle\}$. Further assume that $\mathcal{I}(\langle \mathcal{H}_0^j, \mathcal{H}_\to^j \rangle)$ is the same for all $j = 0, \ldots, k$. Then the *delayed* dynamic influence network (DeDIN, denoted as a DIN $\langle \mathcal{A}_0, \mathcal{A}_\to \rangle$, satisfies all the independence assumptions $\mathcal{I}(\langle \mathcal{H}_0^i, \mathcal{H}_\to^i \rangle)$ $\forall\, i = 0, \ldots, k$. In addition, $\forall\, j$ and $\forall\, t$, $\langle \mathcal{A}_0, \mathcal{A}_\to \rangle^{(t)}$ also satisfies the following independence assumptions for each latent variable denoted L_i and some $t > \alpha \in \mathbb{Z}^+$:

$$\forall\, L_i^{\langle \mathcal{H}_0^0, \mathcal{H}_\to^0 \rangle^{(t)}} : (L_i^{\langle \mathcal{H}_0^0, \mathcal{H}_\to^0 \rangle^{(t)}} \perp\!\!\!\perp \text{NonDescendants}_{L_i^{\langle \mathcal{H}_0^0, \mathcal{H}_\to^0 \rangle^{(t)}}} | L_i^{\langle \mathcal{H}_0^k, \mathcal{H}_\to^k \rangle^{(t)}}, L_i^{\langle \mathcal{H}_0^k, \mathcal{H}_\to^k \rangle^{(t)-1}}, \ldots, L_i^{\langle \mathcal{H}_0^k, \mathcal{H}_\to^k \rangle^{(t)-\alpha}}, \mathrm{Pa}_{L_i}^{\langle \mathcal{H}_0^0, \mathcal{H}_\to^0 \rangle^{(t)}}).$$

The generalisation in Definition 5.4, which describes delayed influence between families of dynamic Bayesian networks, submits that direct influence (Definition 5.3) is a special case where $\alpha = 0$. This generalisation provides the following position:

1. delayed influence offers a less dense ensemble of dependencies than the mesh assemble since it prohibits the edges from later time-slices to previous ones;

2. the delayed assemble provides more sparse structure as α approaches 0;

3. large values of α may capture a richer distribution but sparsity in the model also empirically provides a better generalisation for density estimation (see Figure 42);

4. the assemble provides a representation for applications which require delayed influence between processes (since it utilises information between time-slices from different models).

Similarly to direct influence, implementing the delayed influence assemble (as defined in Definition 5.4) requires the partition of several conditional independence assumptions between variables as discussed in the previous section. We can also use a decomposable score to evaluate the complete structure score by computing the sum of family scores in a model. Specifying the compact independence assumptions for each family in delayed influence requires traversing $\alpha * K * T$ time-slices to construct the parent set of a temporal model and learned parameters for each parent factor.

This means that although we can use Algorithm 3 to compute the score of a DIN using a delayed assemble, constructing the parent set using *ExtDep()* may require an extraction of factors from various time-slices in all temporal models included in the parent set.

Perhaps the most compelling way to demonstrate the increase in factor sizes for delayed influence with a selection of α, is by realising the potential of the penalty term to exponentially increase as a function of the number of independent parameters.

COMPLEXITY ANALYSIS: The computational complexity is much like in Section 5.3.1, however we must incorporate α as a principal feature in the computation of the penalty term. For every model H; for every time-slice T in a model H; and for every variable X (in a time-slice T in a model H) we arrange a conjunction with its parent set, which consists of internal (I) and external (E) variables, to form a family. The number of external variables depends on the value for α. This is because large values of α may cause a large number of external dependencies which cause the number of parameters to express a variables to increase exponentially. For each family we compute the sufficient statistics by scanning though instances/samples in the data $\mathcal{D} = \{\xi_1, \ldots, \xi_M\}$, which takes M units of time. Therefore asymptotically, our total time complexity is $O(HTX(I + \alpha E)M)$.

In summary, although the delayed assemble defines a rich and expressive representation, it also offers a strong risk of fragmentation; a larger computational burden to deduce a score; and a less sparse representation than direct influence. It however extends and maintains our intuitive understanding of influence between temporal processes and is able to handle more complex distributions between processes.

In the next section we will empirically show the effect of selecting different assembles to describe influence in a DIN.

5.3.3 *Empirical Analysis of Structure Assembles*

In this section we empirically demonstrate the effects of using different structural assembles to reconstruct a ground-truth DIN. Figure 42 shows five learning tasks of several assembles to reconstruct a ground-truth DIN structure over 10 trials. The ground-truth was constructed using a delayed assemble with 10 HDBN models with 5 time-slices each; 1 latent layer; 3 observations per latent variable; 4 bins per variable (i.e. each variable has a distribution over four values); a Dirichlet prior of 5; $\alpha = 3$; and 10 EM iterations to learn each latent variable. Each of the learning tasks were given the true structure of influence between HDBNs.

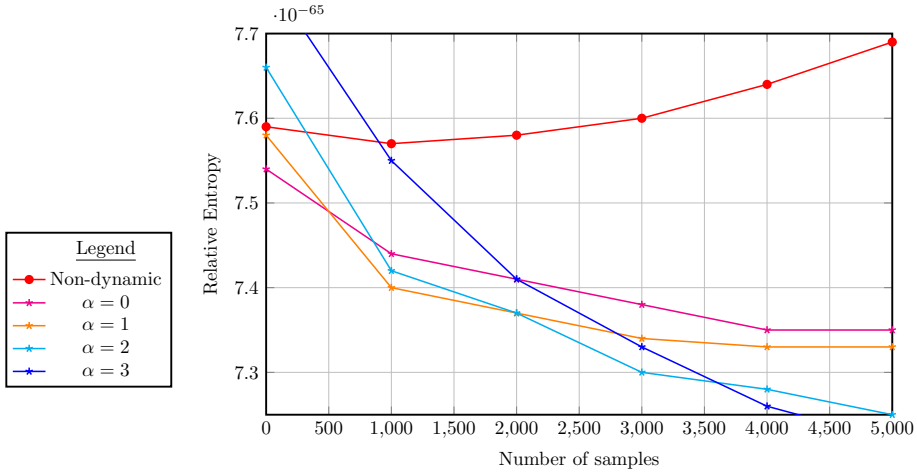


Figure 42: The average performance of five learning tasks to reconstruct a ground-truth DIN over 10 trials. The DIN consists of 10 HDBN models (which takes the form described in Figure 34), with 5 time-slices each; 1 latent layer; 3 observations per latent variable; 4 bins per variable; a Dirichlet prior of 5 (i.e. a uniform prior with 5 imaginary samples); and 10 EM iterations to learn each latent variable.

The five learning tasks are as follows: "Non-dynamic" (red) models influence between HDBNs without any assemble, this means that the temporal aspect of the data was ignored; "$\alpha = 0$" where the direct influence is modelled to describe influence between HDBNs; and "$\alpha = 1,\ 2,\ 3$", which models delayed influence with different values of α as defined in Definition 5.4.

In Figure 42, as indicated by the red line, modelling dynamic influence using a non-dynamic model moves further away from the ground-truth DIN. This is because samples from the ground-truth DIN are generated from adjacent time-steps which can not be captured in a non-dynamic parametrization. Thus the probability of new data instances/samples become less likely.

Using an assemble relation over the HDBN models seems to cope better with describing the influence relations between models than in the non-dynamic case. We also note that although the higher values of α start off in a worse position relative to the other learning tasks due to having more parameters to learn, they cope better by generalising to the true distribution given more data. Therefore, higher values of α are better when we have more data, and lower values provide a sparser representation to generalise better with fewer samples. In the next chapter we explore an algorithm to recover the structure for DINs.

6

INFLUENCE STRUCTURE SEARCH

6.1 INTRODUCTION

The previous chapter analysed various scores and assembles to evaluate the quality of influence networks with respect to some dataset. These scores included the dynamic likelihood and the d-BIC scores; and the assembles included the mesh, direct, and delayed assemble. In this chapter we focus on finding the highest scoring influence network given a set of candidate structures.

More formally, we have a well-defined optimisation problem where our input to a search procedure is as follows:

1. A training set $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}} = \{\mathcal{D}_{\langle \mathcal{H}_0^1, \mathcal{H}_\rightarrow^1 \rangle}, \ldots, \mathcal{D}_{\langle \mathcal{H}_0^k, \mathcal{H}_\rightarrow^k \rangle}\}$, where $\mathcal{D}_{\langle \mathcal{H}_0^i, \mathcal{H}_\rightarrow^i \rangle} = \{\xi_1, \ldots, \xi_M\}$ is a set of M instances from a ground-truth DBN $\langle \mathcal{H}_0^i, \mathcal{H}_\rightarrow^i \rangle$;

2. scoring function, $\text{score}(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle : \mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}})$;

3. and a set of distinct candidate structures, $\mathcal{G} = \{\mathcal{G}^1, \ldots, \mathcal{G}^L\}$, where L is the number of candidate structures and each structure $\mathcal{G}^l$ encodes a unique set of local independence assumptions $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{G}^{\mathbb{I}} \cup \mathcal{G}^{\mathbf{B}})$.

The output of the search procedure is the highest scoring influence network. In this chapter, we ignore the choice of structure score and assemble. However, we do recommend that the score is decomposable and score equivalent so as to take advantage of special search properties for computational savings (eg. reducing the number of computations to calculate a structure score, see Section 6.5.3).

Note that the scoring function and candidate structures can incorporate any necessary prior knowledge. This prior knowledge can be elicited as high-level structures or as distributions over parameters. The ability to include prior knowledge easily is an advantage of using probabilistic graphical models in our representation of influence.

Traditional search techniques traverse a set of structures between variables rather than structures between multiple temporal models. The machinery we provide in this chapter will enable us to pursue an optimal influence network by traversing a set of candidate DINs. The main contributions of this chapter are as follows:

- a complete algorithm to recover influence between temporal models with latent variables. This is an extension of the greedy structure search algorithm to select an optimal influence network;

- an adaptation of the EM algorithm to learn missing parameters in influence networks;

- several techniques for computational saving in recovering influence distributions.

The theoretical developments on dynamic structure scores (Section 5.2) in the previous chapter and as well as the search procedure provided in this chapter is empirically evaluated in Chapter 7.

In the next section we provide an algorithm to perform a structure search for DINs. Section 6.2 outlines the general algorithm to recover influence between temporal models; Section 6.3 discusses learning mutually independent temporal models; Section 6.4 discusses learning tree-structured DINs; and finally, Section 6.5 discusses learning general graph-structured DINs.

6.2 INFLUENCE STRUCTURE SELECTION

Our high-level algorithm to learn influence between stochastic processes are as follows. We firstly describe each process separately by a temporal model, and secondly, try to infer influence between these temporal models. The high-level algorithm is presented in Algorithm 5, where $\mathcal{S} = \{\mathcal{S}^1_{\rightarrow}, \ldots, \mathcal{S}^P_{\rightarrow}\}$ is a set of stochastic processes; $\mathtt{assemble}$, is a choice for an assemble; and $\mathtt{score}$, is any scoring function.

Algorithm 5 Influence structure search

1: **procedure** StrucSearch($\mathcal{S} = \{\mathcal{S}^1_{\rightarrow}, \ldots, \mathcal{S}^P_{\rightarrow}\}$, $\mathtt{assemble}$, $\mathtt{score}$)
2: Learn a temporal model for each stochastic process ($H = \{\langle \mathcal{H}^1_0, \mathcal{H}^1_{\rightarrow}\rangle, \ldots, \langle \mathcal{H}^P_0, \mathcal{H}^P_{\rightarrow}\rangle\}$)
3: Generate a search space over the models in H (ie. $\mathbf{G} = \{\mathcal{G}_1, \ldots, \mathcal{G}_n\}$)
4: Search for the structure $\mathcal{G}_i$ which maximises $\mathtt{score}$ in $\mathbf{G}$ w.r.t. $\mathtt{assemble}$
5: **return** $\mathcal{G}_i$

In this section we outline the general algorithm to recover influence between temporal processes. We assume that the data generated from the ground-truth influence structure has the following form: $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}}} = \{\mathcal{D}_{\langle \mathcal{H}^1_0, \mathcal{H}^1_{\rightarrow}\rangle}, \ldots, \mathcal{D}_{\langle \mathcal{H}^k_0, \mathcal{H}^k_{\rightarrow}\rangle}\}$, where $\mathcal{D}_{\langle \mathcal{H}^i_0, \mathcal{H}^i_{\rightarrow}\rangle} = \{\xi_1, \ldots, \xi_M\}$ is a set of M instances from HDBN $\langle \mathcal{H}^i_0, \mathcal{H}^i_{\rightarrow}\rangle$. Each ξ_m is a vector containing N features. All of the data in $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}}}$ are all generated IID (Section 2.3.1.1) over time from an underlying temporal distribution, $P^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}})$, where $\mathbb{I}_0$ is an initial network and $\mathbb{I}_{\rightarrow}$ is an unrolled network with respect to structure $\mathcal{G}$.

$\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}}$ contains a distribution between a set of HDBN models, $\langle \mathcal{H}^1_0, \mathcal{H}^1_{\rightarrow}\rangle, \ldots, \langle \mathcal{H}^k_0, \mathcal{H}^k_{\rightarrow}\rangle$, with the independence assumptions specified by $\mathcal{I}(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}})$. We further assume that $P^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}})$ is induced by another model, $\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}})$, which we refer to as the *ground-truth* structure. We evaluate our model by attempting to recover the local independence assertions in $\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}})$, denoted $\mathcal{I}(\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}}))$, by only observing $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_{\rightarrow}\rangle^{\mathcal{G}}}$.

The architecture of the proposed algorithm is given by Figure 43. We (ii) learn an HDBN for each process independently (using Expectation Maximisation (EM)); (iii)

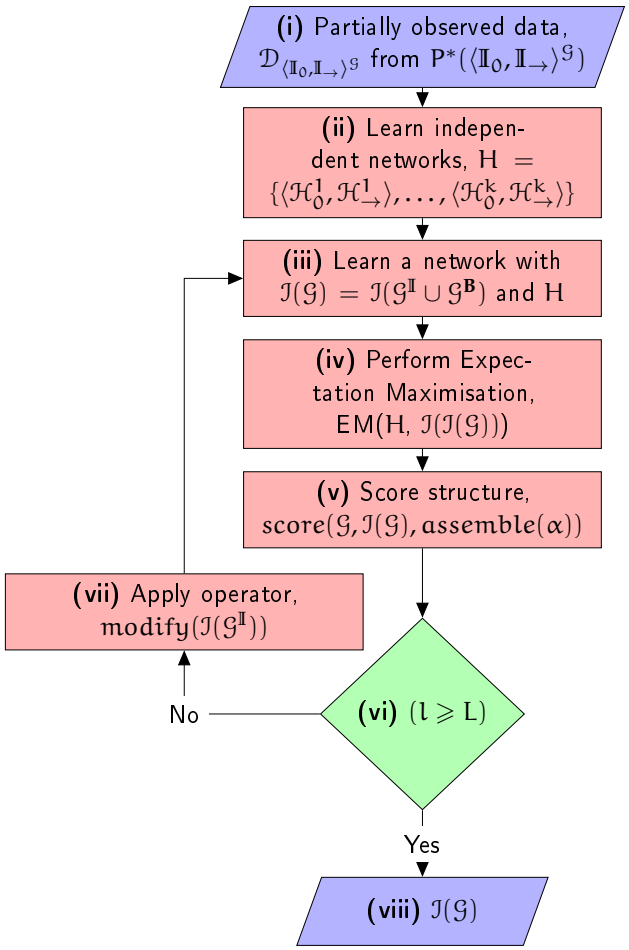


Figure 43: A more detailed architecture of the proposed algorithm to recover influence between stochastic processes represented as temporal networks.

set the independence assumptions with respect to each temporal model (ie. induce a model from $\mathcal{I}(\mathcal{G}) = \mathcal{I}(\mathcal{G}^{\mathbf{I}} \cup \mathcal{G}^{\mathbf{B}})$) and learn the resulting dynamic influence network; (v) compute the structure score of the model (using a scoring function and an assemble (Section 5.3) for influence networks); (vi) see if we converge or if the number of iterations (l) exceeds the iteration threshold (i.e. the number of iterations, denoted by L); (vii) apply the operator which results in best improvement of the score with respect to the data. Steps (iii), (iv), (v), and (vii) are repeated until we can not improve the score for the structure with respect to the data or if we exceed the specified number of iterations. We then select the best network (viii). This algorithm is referred to as the greedy structure search algorithm (GESS).

We separate our goal objective into (a) learning mutually independent dynamic Bayesian models (Section 6.3) and then (b) learning the structure between these processes (Section 6.4 and Section 6.5).

6.3 LEARNING MUTUALLY INDEPENDENT MODELS

Suppose that we learn a stochastic process, $\mathcal{S}_{\rightarrow}$, using a hierarchical dynamic Bayesian network, $\langle \mathcal{H}_0, \mathcal{H}_{\rightarrow} \rangle$. Then we must learn the latent variables at leach level of the hierarchy for $\langle \mathcal{H}_0, \mathcal{H}_{\rightarrow} \rangle$. For example, in Figure 44, we need to learn the latent variables

$H_1^{(t)}$, $H_2^{(t)}$, and $H_3^{(t)}$ for all time-slices t. In Section 6.3.1 we extend the traditional EM algorithm to learn the latent variables in DINs.

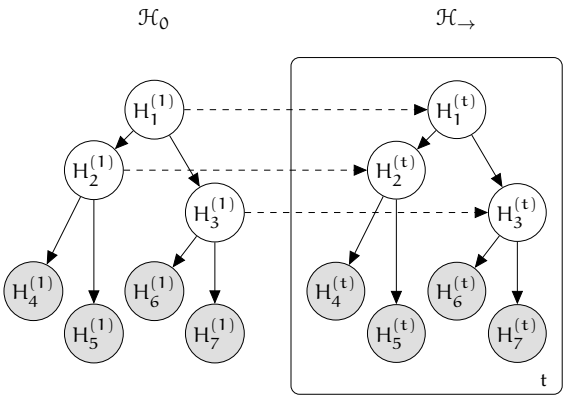


Figure 44: A HDBN for a stochastic process illustrated using plate notation. Each variable is denoted $H_i^{(t)}$, where H_i is the template variable in time-slice t.

6.3.1 *Expectation Maximisation*

The latent variables in Figure 44 can be learned using Expectation Maximisation (EM) outlined in Algorithm 1. The EM algorithm attempts to learn both the missing data and the parameters simultaneously [Rabiner and Juang 1986]. We provide an adaptation of the EM algorithm for DINs. The proposed EM algorithms takes in seven arguments outlined in Table 1.

$\mathcal{O} = (O_1, \ldots, O_j)$	Set of observable variables
$\mathcal{L} = (L_1, \ldots, L_k)$	Set of latent variables
$\text{Var} = (O_1, \ldots, O_j, L_1, \ldots, L_k)$	Set of all variables
$\mathcal{I}(\mathcal{H})$	Independence assertions for $\mathcal{H}$
EMit	The number of EM iterations
$\text{Dir}(\alpha)$	Array specifying the Dirichlet prior used for each variable corresponding to Var
NumBins	Array specifying the number of bins used for each variable corresponding to Var.

Table 1: Argument definition for EM algorithm

We present the general adaptation of the EM algorithm in Algorithm 6. The procedure outlines an implementation of EM for DINs and returns a set of factors with the learned parametrisation given seven input arguments. Line 2 initialises our latent dataset, denoted $\mathcal{D}_\mathcal{L}$, with random values. These random values will be replaced with new data values at each iteration of the EM procedure. At each iteration the likelihood of the parameters to the generated data monotonically increases [Minka and Lafferty 2002; Caffo *et al.* 2005].

There are $\mathcal{O}.length$ instances in this dataset, each consisting of $\mathcal{L}.length$ values for all latent variable in $\mathcal{L}$. Line 3 initialises a set of factors, $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_j}]$ using the **M-step**^{0}, which corresponds to the set of variables, Var. Each factor $\mathcal{F}$ contains all possible combinations of values that a variable can take with an probability value. Each factor is a probability distribution which means that each probability is positive and all probabilities in the factor sum to 1. This step serves to initialise the parameters as independent factors. Lines 5 and 6 perform the **E-step** and **M-step** iteratively until convergence. A detailed description of the **E-step** and **M-step** is given in the appendix of this document (Appendix A).

The **M-step**^{0}, on Line 2 of Algorithm 6, initialises the set of factors $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_j}]$. Algorithm 10 provides a high-level procedure to initialise this set of factors.

Algorithm 6 Expectation Maximisation for influence Networks

1: **procedure** EXPECTATIONMAXIMISATION(NumBins, $\mathcal{O} = (O_1, \ldots, O_j)$, $\mathcal{L} = (L_1, \ldots, L_k)$, Var $= (O_1, \ldots, O_j, L_1, \ldots, L_k)$, $\mathcal{I}(\mathcal{H})$, EMit, ,)
2: $\mathcal{D}_{\mathcal{L}} =$ **Rand**$(\mathcal{O}.length, \mathcal{L}.length)$
3: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_j}] =$ **M-step**$^0(\mathcal{D}_{\mathcal{L}}, \mathcal{D}_{\mathcal{O}}, \mathcal{L}, \mathcal{O}, \text{Var}, \mathcal{I}(\mathcal{H}), ,, \text{NumBins})$
4: **for** $i = 0$ to EMit **do**
5: $\mathcal{D}_{\mathcal{L}} =$ **E-step**$([\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_j}], \mathcal{D}_{\mathcal{O}}, \mathcal{O}, \mathcal{L}, \mathcal{I}(\mathcal{H}), \text{Var}, \text{NumBins})$
6: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_j}] =$ **M-step**$(\mathcal{D}_{\mathcal{L}}, \mathcal{D}_{\mathcal{O}}, \mathcal{O}, \mathcal{L}, \text{Var}, \mathcal{I}(\mathcal{H}), ,, [\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_j}])$
return $[\mathcal{F}_{O1}, \ldots, \mathcal{F}_{L_j}]$

There are two ways of implementing the EM algorithm for learning latent parameters: the soft-assignment and hard-assignment. The relationship between soft-assignment and hard-assignment EM was discussed by Kearns *et al.* [1998]. We use the hard-assignment EM, which traverses the likelihood discretely, to learn the initial temporal model. This is discrete since the hard-assignment EM uses the most likely assignment to the data (ie. MAP estimate). The hard-assignment EM is considered less accurate but converges faster than the soft-assignment EM [Kearns *et al.* 1998]. We use the soft EM, which traverses the likelihood continuously (ie. samples data instances using the complete distribution over all factors), to learn the complete DIN structure. We use soft-assignment EM to learn the complete model. Even though the soft-assignment EM takes longer to converge, it produces generally better results since it traverses the data continuously [Kearns *et al.* 1998; Koller and Friedman 2009].

Leaning temporal models for processes individually does not tell us anything about the influence between them. It does, however, provide us with a representation for each process to make meaningful comparisons with other processes. In the next section we use these temporal models to build tree structured DINs.

6.4 LEARNING TREE-STRUCTURED INFLUENCE NETWORKS

Learning a tree structured network is perhaps the simplest structure learning problem. Most tree structure learning procedures require a computation complexity with is usually polynomial [Chow and Liu 1968]. There are several reasons that one would

learn tree structured networks. Firstly, there already exists powerful algorithms for efficient optimisation over high-dimensional tree structured networks; and secondly, trees provide sparse networks which reduces over-fitting to data.

After selecting a dynamic structure score, we turn our attention to an optimisation problem which attempts to maximise the selected score over potential tree DIN structures. Decomposability of the score, that is the complete structure score of a DIN is equal to a sum of family scores, turns out to be an important property for decreasing the computational burden in the structure search procedure. Suppose we want to calculate the score over a particular network structure. If our score is decomposable we can express the weight between a variable, j, and its parent, i, as the improvement to the likelihood that a dependency between variable j and i yields, over not having the dependency.

More formally,

$$w_{i\to j} = \mathrm{score}(X_j|X_i) - \mathrm{score}(X_j). \tag{24}$$

The second term ($\mathrm{score}(X_j)$) does not depend on the influence structure and can be ignored. In the case of the likelihood score the expressions of $w_{i\to j}$ in Equation 12 becomes

$$w_{i\to j} = M \sum_{x_i,x_j} P(x_i,x_j) \log \frac{P(x_i,x_j)}{P(x_i)P(x_j)}.$$

The sumation is called the mutual information (see Equation 21). Although the mutual information can never be negative, when using other scores, such as the d-BIC score, we may have a penalty term which causes the resulting score to be negative. This implies that using the likelihood score will result in a tree structure as opposed to obtaining a forest when using the d-BIC score. This usually happens when there is no set of positively weighted edges which span every temporal model (ie. a path from every temporal model to every other temporal model).

A second important observation is that score equivalent networks, those that satisfy Definition 3.3, have the same dynamic likelihood score for $w_{i\to j}$ and $w_{j\to i}$. This is because of the symmetry of the mutual information term which consequently results in a undirected tree structure. The consequence of this symmetry results in there being no way to determine the orientation between two variables from the observable data alone.

A general algorithm to obtain a tree/forest structured network is to compute the score of every pair of variables. Each score between every pair of variables will be a weight from one variable to another resulting in a undirected graph. We then use an algorithm to find the maximum weighted spanning tree (MWST) or forest. One could use any standard MWST algorithm such as Prim or Kruskal in $O(n^2)$, where n is the number of temporal models. Finally, we impose an acyclic orientation of the edges between variables in the DIN using any method of choice.

6.5 LEARNING GRAPH-STRUCTURED INFLUENCE NETWORKS

In Chapter 5, we discussed several dynamic scores and structural assembles that can be used to evaluate the quality of DINs for a set of temporal processes. We now consider the task of searching through different DIN structures and selecting one that optimises the given dynamic score through some given structural assemble. There are many notable structure search procedures, these include attempts to recover tree-structured networks [Chow and Liu 1968] and Bayesian network structures [Cooper and Herskovits 1992]. Local search procedures over general networks have also received much attention [Chickering *et al.* 1995; Buntine 1991]. However, no search procedure has been proposed to optimise a dynamic structure score over networks with latent variables.

Decomposability is an important aspect of a dynamic score for our structure search procedures, meaning that we can write the complete dynamic score as a sum of family dynamic scores. Another property is that of score-equivalence which states that I-equivalent structures will have the same dynamic score.

As stated in Theorem 3.4 learning the graph structure for a Bayesian network is NP-hard for any restriction on the in-degree greater than or equal to 2. We can similarly state an extension of this theorem as learning an influence graph structure between temporal models is NP-hard for any restriction on the in-degree greater than or equal to 2. In the case of learning general graphical structures, the problem's complexity increases. More formally, for any dataset, $\mathcal{D}$; decomposable structure score $\mathtt{score}$; and any structural assemble, the problem of finding the maximum scoring network, that is,

$$\mathcal{G}^* = \arg\max_{\mathcal{G}\in\mathcal{G}_{\mathbf{d}}} \mathtt{score}(\mathcal{G} : \mathcal{D}), \tag{25}$$

is NP-Hard for any $d \geqslant 2$, where $\mathcal{G}_{\mathbf{d}} = \{\mathcal{G} : \forall\ i,\ k,\ t, |\mathbf{Pa}^{\mathcal{G}}_{X_i^{\langle \mathcal{H}_0^k, \mathcal{H}_{\rightarrow}^k \rangle(t)}}| \leqslant d\}$.

In other words, finding the maximal scoring influence structure with at most d parents for each variable in any temporal model, k, at any time-slice, t, is NP-hard for any d greater than or equal to 2. This is because of the super-exponential search space [Pólya 1937] that one has to traverse to obtain the maximal network. Thus, we resort to approximating the DIN for indegrees greater than or equal to 2. We are faced with a combinatorial optimisation problem to detect the optimal DIN. We solve this problem by utilising a local search procedure.

More formally, we define a search space which defines: the set of candidate network structures; a dynamic scoring function, that we aim to maximise over DINs; the structure assemble, which associates our temporal models; and finally, a search procedure which explores the search space of possible DINs. We have already discussed the dynamic structure score and structural assemble in Chapter 5, which leaves us to discuss the search space in Section 6.5.1; the search procedure in Section 6.5.2; and complexity of our proposed algorithm in Section 6.5.3.

6.5.1 *The Search Space*

Equation 25 suggests that exploring all possible DIN structures is computationally futile for a large amount of processes, and so we develop a heuristic solution to solve this problem. We present a heuristic solution in the form of a greedy hill-climbing search.

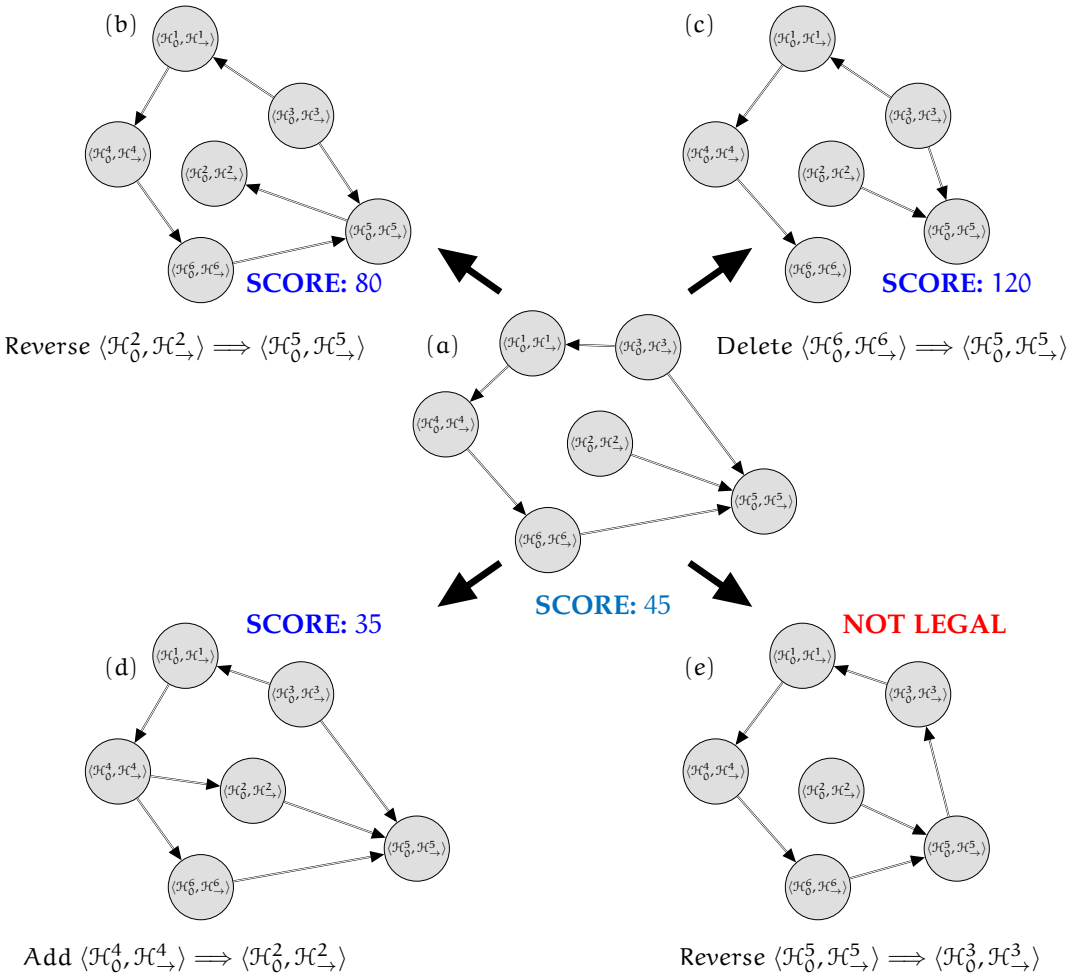


Figure 45: An illustration of possible transitions from a candidate influence networks. This figure shows local perturbations in an attempt to improve the network structure relative to the data, a particular score, and an assemble. Each node represents a temporal model and double arrow edges represent an assemble. Network (a) provides the current network structure. Network (b) reverses the edge $\langle \mathcal{H}_0^2, \mathcal{H}_\rightarrow^2 \rangle \Longrightarrow \langle \mathcal{H}_0^5, \mathcal{H}_\rightarrow^5 \rangle$ which increases the network score by 35. Network (c) deletes the edge $\langle \mathcal{H}_0^6, \mathcal{H}_\rightarrow^6 \rangle \Longrightarrow \langle \mathcal{H}_0^5, \mathcal{H}_\rightarrow^5 \rangle$ which increases score by 75. Network (d) adds an edge $\langle \mathcal{H}_0^4, \mathcal{H}_\rightarrow^4 \rangle \Longrightarrow \langle \mathcal{H}_0^2, \mathcal{H}_\rightarrow^2 \rangle$) which decreases the score by 10. Finally, network (e) reverses the edge $\langle \mathcal{H}_0^5, \mathcal{H}_\rightarrow^5 \rangle \Longrightarrow \langle \mathcal{H}_0^3, \mathcal{H}_\rightarrow^3 \rangle$ obtaining an illegal network structure.

Suppose that we have an arbitrary candidate network as depicted at the center of Figure 45 labeled (a). We may perform local perturbations in an attempt to improve the network structure relative to the data and selected dynamic score with respect to an assemble. We consider the common choices for local search operators with

respect to the selected assemble. Such operators include the edge addition, reversal, and edge deletion. These computationally cheap operators ensure that the diameter of the search space is small (K^2, where K is the number of models) and manageable (ie. can be explored, unlike in the case when we had to define a super-exponential number of networks). Suppose that we are given the following options:

1. to reverse the edge $\langle \mathcal{H}_0^2, \mathcal{H}_\rightarrow^2 \rangle \Longrightarrow \langle \mathcal{H}_0^5, \mathcal{H}_\rightarrow^5 \rangle$ obtaining network (b) which gives us a score of 80;
2. to delete the edge $\langle \mathcal{H}_0^6, \mathcal{H}_\rightarrow^6 \rangle \Longrightarrow \langle \mathcal{H}_0^5, \mathcal{H}_\rightarrow^5 \rangle$ obtaining network (c) which gives us a score of 120;
3. to add an edge $\langle \mathcal{H}_0^4, \mathcal{H}_\rightarrow^4 \rangle \Longrightarrow \langle \mathcal{H}_0^2, \mathcal{H}_\rightarrow^2 \rangle$ obtaining network (d) which gives us a score of 35;
4. or to reverse the edge $\langle \mathcal{H}_0^5, \mathcal{H}_\rightarrow^5 \rangle \Longrightarrow \langle \mathcal{H}_0^3, \mathcal{H}_\rightarrow^3 \rangle$ obtaining network (e) which results in a cyclic network. The edge reversal search operator is necessary since deleting an edge, with the intention to perform a edge reversal, might lower the overall structural score for the next step. Thus, having an edge reversal transition allows us to explore the option of reversing edges in one search step.

We do not consider selecting the operation which leads to an illegal DIN structure and so we do not consider option (e). The most favorable transition would be to delete the edge $\langle \mathcal{H}_0^6, \mathcal{H}_\rightarrow^6 \rangle \Longrightarrow \langle \mathcal{H}_0^5, \mathcal{H}_\rightarrow^5 \rangle$. This option improves the current network score from 45 to 120, perhaps because the dynamic scoring function prefers more sparse networks (this might not necessarily mean that the selected graph is closer to the ground-truth).

In local search procedure it is not clear whether transitions are favorable in the long term, however, it does provide a better candidate networks from a local perspective. Several search techniques (eg. random restarts and tabu lists) are discussed later which assist the local search procedure to consider more suitable networks.

6.5.2 *Local Search Procedure*

The second design choice is to select a search technique to traverse the search space. Chickering *et al.* [1995] compared various search procedures to learn the structure for Bayesian networks, including the K2 algorithm, local search procedures, and simulated annealing. Since a DIN is a Bayesian network with a restriction on the edges, we expect that the comparisons made by Chickering *et al.* [1995] are relevant to our task of learning the structure of a DIN. Chickering *et al.* [1995] show that for learning Bayesian networks, local search procedures offers the best time-accuracy trade-off. We therefore employ a greedy hill-climbing local search procedure to discover influence between temporal processes. The technique follows the following general algorithm:

1. pick an initial starting point $\mathcal{G}$ (or prior network) and compute its score using the assemble.
2. consider all neighbours of $\mathcal{G}$ which are possible transitions given the search operators, and compute their scores.
3. apply the change which leads to the best improvement of the score.

PRIOR STRUCTURES The prior network could be an empty network; a best tree obtained from the procedure mentioned in Section 3.5; a random network; or one elicited from an expert. From this prior network we iteratively try to improve the network by utilising search operators. In greedy hill-climbing we always apply the change that improves the score until no improvement can be made.

The returned structure from the greedy hill-climbing local search procedure above can either have reached a local optimum or a plateau. The resulting network structure can be interpreted as a local optimum for which no operator can improve the network score. The second problem arises when we encounter a plateau in the search space. If we look around at candidate structures there may be a verity of possible DINs transitions which give the same score. In this case we have no information to tell us in which direction to proceed.

This occurs frequently in Bayesian network structure learning due to I-equivalent networks. Since some scores are also score-equivalent we get the same score for any network in the same I-equivalence class (eg. those in Figure 8). To increase our chances of escaping local optima and avoiding plateaus we use random restarts and tabu lists.

RANDOM RESTARTS: In random restarts, when we reach a local optimum we take n random steps and then continue traversing the search space using the search procedure. The intuition is that if we are at a local optimum then perhaps n random steps will set us up in a better position to explore the search space for a better optimum.

TABU LISTS: In Tabu lists we try to avoid treading the same path over and over again by maintaining a list of the most recent n steps taken by the search procedure. Glover and Laguna [2013] provide a detailed discussion on the effects of Tabu lists on local structure searches.

6.5.3 *Computational Complexity and Savings*

We outline the *rough* computational analysis of our algorithm to recover dynamic influence between processes. We perform Bayesian estimation and EM for each HDBN prior to the structure learning task which takes $O(TNM)$-time. Suppose we have e operators and K HDBN networks, if we take D steps before convergence then we perform $O(DK^2)$ operators. Each applied operator involves an acyclicity check ($O(K)$, or $O(dK)$ if we know the in-degree d of the DIN). To evaluate an acyclic network we calculate the score which takes $O(KTNM)$ plus the cost to perform EM with i iterations. This gives us a total complexity of $O(DK^2(KTNM + dK))$. To provide significant computational saving we use a score cache to store our sufficient statistics for speedy score computations; and we use a priority queue to manage candidate structures and their respective scores.

In the next chapter we demonstrate the performance of our complete learning algorithm with respect to several baselines.

7

EXPERIMENTAL RESULTS

7.1 INTRODUCTION

In this chapter we provide an empirical evaluation of several learning methods to recover a ground-truth dynamic influence network (DIN). More specifically, we learn the structure and parameters of a probability distribution that describes the influence between a set of multi-dimensional partially observable stochastic processes. In order to achieve this, we separate our goal into (a) learning the structure of non-dynamic influence networks between latent variables; (b) learning the structure of direct dynamic influence networks (DiDINs) and delayed dynamic influence networks (DeDINs) between hidden Markov models (HMMs); (c) learning DeDINs between hierarchical dynamic Bayesian networks (HDBNs); and (d) demonstrating the scalability of our structure learning methods as we increase the difficulty of this structure learning problem.

In (a) we attempt to learn the non-dynamic influence between a set of latent variables which describe a set of observable features. We describe the relationship between the latent and observable variables using naïve Bayes models (NBMs). The purpose of this experiment is to demonstrate the capability of our greedy structure search (GESS) method to recover the structure of non-dynamic influence between latent variables subject to the accuracy of recovering the latent class labels compared to several baselines.

Having recovered the influence between latent variables in naïve Bayes models in (a), we turn our attention to solving this problem for the dynamic setting. In (b) we are interested in assessing the recovery of a direct dynamic influence network (DiDIN) between HMMs. The purpose of this experiment is to demonstrate that, as we increase the number of samples, the use of our dynamic structure scores (Section 5.2), paired with either the direct (Section 5.3.1) or delayed assemble, is empirically able to recover the ground-truth direct and delayed dynamic influence network between partially observable processes better than selected baselines.

In (c) we further attempt to empirically capture the DeDIN between HDBNs. In HDBNs we may have more latent variables than in HMMs. Although we are still able to recover the ground-truth DIN, we notice that the inability to recover the original cluster assignments for the latent variables leads to poorer performance for a small number of samples than in case (b). The purpose of this experiment is to demonstrate empirically the limitations and capabilities to recover a DeDIN with a more descriptive representation of each stochastic process. However, having more latent variables complicates the problem substantially given the increased number of local optima in the likelihood function to the data.

Having established structure learning methods for the non-dynamic setting in (a); extended the description of the structure learning problem for HMMs that describe stochastic processes in (b); and demonstrated the effectiveness of our structure learning methods for influence with more descriptive temporal models (HDBNs) in (c); we then attempt to demonstrate the scalability of our methods to recover the ground-truth DeDIN for extended difficulty of the structure learning task in (d). These learning tasks include changing the number of time-slices that describe each stochastic process; increasing the number of bins in the discrete description of the random variables; increasing the number of observations; and increasing the size of the ground-truth DeDIN for dense and sparse graphs.

We present learning non-dynamic and dynamic influence structures aside several baselines. The following five baseline structures are considered in this study: using no structure; a random structure; learning with some prior knowledge of the ground-truth structure; using a tree structure; and learning with complete knowledge of the ground-truth structure.

EMPTY STRUCTURE: Learning with no structure assumes that all internal networks (those that describe environments or processes) are mutually independent of each other. In other words, the set $\mathcal{G}^{\mathbb{I}}$ is empty. Hypothetically, this could produce a good approximation for new instances given fewer data since there would be less parameters to learn.

RANDOM STRUCTURE: Using a random structure could result in two scenarios for large amounts of data. On the one hand, dense random influence structures could potentially explain more of the correlations in the data between internal networks. However, we may never be able to recover the ground-truth distribution since increasing the number of independence assumptions between internal networks may constrain our ability of capturing the ground-truth distribution. On the other hand, sparser random influence structures may generalize better to new instances than using no structure since having at least one more dependency (rather than none) will have a higher-likelihood to the data than having an empty influence structure.

DOMAIN KNOWLEDGE: We also can learn using prior domain knowledge about the influence graph structure. In particular, knowing the max in-degree of the ground-truth structure and the number of edges used. Having this prior knowledge can allow us to avoid over-fitting the likelihood of families of internal networks relative to the data. We consider learning with prior knowledge as a penalty-based score. This is implemented by using the likelihood score with an infinite penalty for networks with a max in-degree and number of edges greater than that of the ground-truth structure. Learning with this prior domain knowledge of the ground-truth is denoted as 'GESS with PK'.

TREE STRUCTURES: We may also decide to learn a tree structured network which summarises the most important score-based dependencies between any two internal networks. Trees provide a sparse influence structure which hypothetically generalises better than using no structure since the most important dependencies between internal networks are preserved.

TRUE STRUCTURE: Finally, we may have the true influence structure. That is, the influence structure which generated the data. In this case we would expect that knowing the true influence structure between internal networks should produce the highest likelihood to the data (since it generated it). However, we only have access to the true structure, not the true parameters, which means we have to relearn the observable and latent parameters manually from the observable data.

We provide the following contributions in this chapter: an empirical comparison of several parameter and structure learning methods (alongside the aforementioned baselines) for recovering the underlying influence network for:

1. latent variables learned from a set of observations in naïve Bayes models [Ajoodha and Rosman 2017] (a);
2. direct influence between HMMs (b);
3. delayed influence between HMMs [Ajoodha and Rosman 2018] (b);
4. general influence between HDBNs (c);
5. changing the condition of influence, such as increasing the number of time-slices, bin-values, observations, and the sizes of dense and sparse influence structures (d).

We firstly present the results of the non-dynamic case of influence in Section 7.2, where we attempt to learn the structure between latent variables; in Section 7.3 we present the results of discovering direct and delayed influence between HMMs; in Section 7.4 we present the results of learning general influence between HDBNs; and finally, in Section 7.5, we interpret all the results presented based on various learning criteria.

7.2 LEARNING IN THE NON-DYNAMIC CASE

Current structure learning practices in Bayesian networks have been developed to learn the structure between observable variables and learning latent parameters independently. However, no method has demonstrated learning the influence structure (as defined in Definition 4.2) between latent variables that describe (or are learned from) a number of observations. In this section we present a method that learns a set of naïve Bayes models (NBMs) (Section 2.2.1) independently given a set of observations, and then attempts to track the high-level influence structure between every NBM. The latent parameters of each model are then relearned to fine-tune the influence distribution between models for density estimation.

Applications of this method include knowledge discovery and density estimation in situations where we do not fully observe characteristics of the environment. For example, we may know the features which describe particular market characteristics (eg. risk, volatility, selection, liquidity, regulation (etc.)) and may want to describe how different markets influence each other.

Figure 46 illustrates an example of an influence structure (as defined in Definition 4.2) between several NBMs. The dotted lines between the models indicate the

high-level influence between the latent variables. Our approach to recover the underlying distribution attempts to learn the optimal influence structure by firstly learning a set of independent naïve Bayes models, and thereafter, optimising a structure score over possible structural configurations of these models.

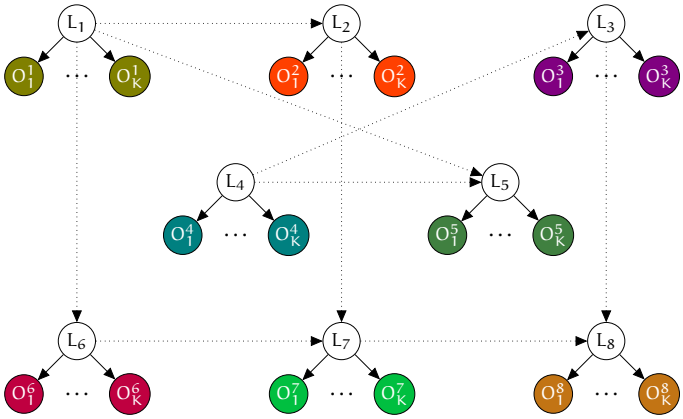


Figure 46: A graphical depiction of the influence structure between several NBMs. Each set of observations for latent variable L_i is denoted as $O_1^i, \ldots, O_K^i$. The solid lines indicate the conditional independence assumptions of each NBM, and the dotted lines indicate the high-level structure of influence between each NBM.

Throughout the experiments in this chapter our sampled dataset $\mathcal{D}$ has been organised as follows: $\mathcal{D} = \mathcal{D}_t$, where $t = 1, \ldots, T$ and each $\mathcal{D}_t$ contains a training, test, and validation set. The training-set contains 80% of the samples and is used to train the DIN; the test-set contains 10% of the samples and is used to assess the predictive performance; and finally, the validation-set contains 10% of the samples and is used for selecting structure priors (eg. tree structures), parameter priors and hyper-parameters (e.g. Dirichlet priors) where necessary. The recovered model cannot be allowed to use the sampled latent values from the ground-truth generative network. Thus we provide the set from $\mathcal{D}_t$ with only observable variables to our structure learning methods.

Figure 47 shows the performance of four parameter or structure learning methods to recover the underlying ground-truth distribution. The error bars show the standard deviation of the relative entropy over 10 trials and are given by the shaded regions. Relative entropy is a measure of how one probability distribution diverges from a second probability distribution [Joyce 2011]. Intuitively, the larger the relative entropy - the further apart the distribution are. The y-axis is the log-scale relative entropy to the true distribution, $P^*(\mathbb{I}^{\mathcal{G}})$, and the x-axis represents the increase in the number of training samples.

The set-up of parameters for the ground-truth non-dynamic influence network was as follows. Each influence ground-truth network had: 10 naïve Bayes models; 3 values per random variable; 15 edges in the influence structure; a max in-degree of 2 in the influence network structure; and 5 observable variables per latent variable. Examples of data from this ground-truth non-dynamic influence network is given in Section A.1.

The four learning methods, in Figure 47, are: 'random structure', where a random structure is generated and we learn the (latent and observable) parameters from the

observable data; 'No structure', where no conditional independence assertions are present between models and we learn the (latent and observable) parameters from the data; 'Learned structure', where we simultaneously estimate the (latent and observable) parameters and structure between models using the greedy structure search (GESS) (Section 6.2) described in the previous chapter with the standard BIC score (Section 3.3); and finally, 'True structure', where we are given the true structure between models and attempt to learn the (latent and observable) parameters. For reproducibility, all of the parameters for each learning method are outlined in Table 2.

	Selection	Random structure	No structure	GESS with BIC	True structure
5	No. edges	-	-	-	15
7	No. observable var	5	5	5	5
8	Dirichlet prior	5	5	5	5
9	Parameter threshold	-	-	2000	-
14	EM iterations	20	20	20	20
15	EM accuracy $(\mu\%, \sigma)$			$(70\%, 9.3)$	
16	Likelihood score	-	-	Log-Like	-
17	Penalty score	-	-	BIC-pen	-
18	Search iterations	-	-	20	-
19	No. random restarts	-	-	5	-
20	Tabu-list length	-	-	5	-

Table 2: A summary of the parameters used by the parameters and structure learning methods for recovering the influence between naïve Bayes models.

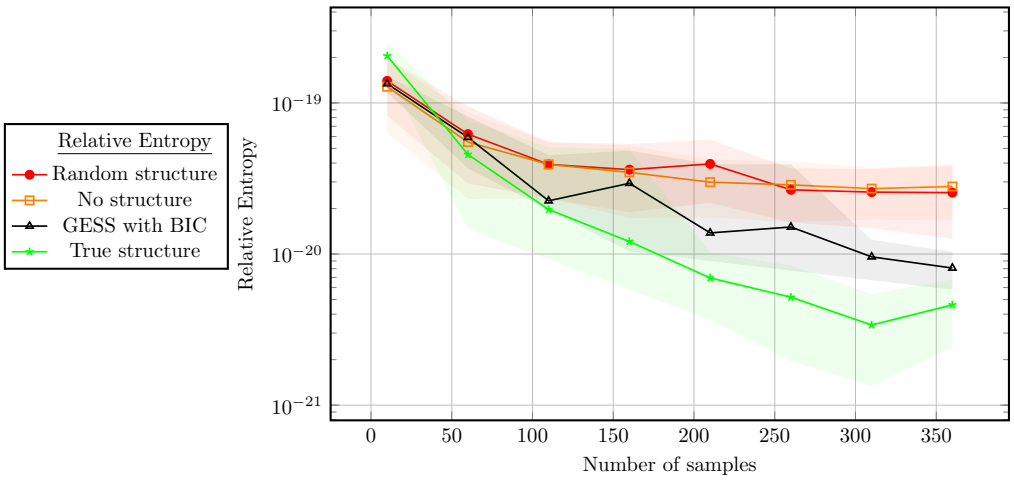


Figure 47: The performance of parameter and structure learning methods to recover the influence structure between naïve Bayes models. The error bars show the standard deviation of the relative entropy over 10 trials and are given by the shaded regions.

Figure 47 suggests that learning the (latent and observable) parameters of mutually independent models (orange) or using a random structure (red) with learned

parameters performs worse on average when compared to learning the (latent and observable) parameters of the true (green) or learned structure (black) using the standard BIC score with the GESS algorithm (Figure 43).

We can use a t-test to assess the significance of the difference between two sample sets. When performing a t-test we calculate a t-value and p-value. The t-value is a ratio of the difference between two sample sets and the difference within the sample sets. The p-value is calculated from the t-value and is the probability that the results of the sampled data used to calculate the t-value occurred by chance. For more information about t-tests see Cohen [1992]. Generally a p-value less than or equal to 0.05 rejects the null hypothesis that the samples from the two distributions are not significantly different. We will use the t-test to demonstrate statistical significance within our experiments.

Referring back to Figure 47, we find that using no structure performs significantly worse (p-value of $0.028 < 0.05$) than using GESS with BIC. This is since using no structure provides no possible way to encode a parameterization in the influence network which describes the relationships between the naïve Bayes models. Using a random influence structure also performs significantly worse than using GESS with BIC (p-value of $0.0018 < 0.05$). This is because a random structure may not be able to describe the ground-truth distribution since every incorrect independence assertion constrains the networks possible parametrization. However, at early iterations (5-70 samples), the relative entropy to the ground-truth distribution is similar for all learning methods. As expected, knowing the true structure as the number of samples increases learns the closest distribution to the ground-truth compared to the other learning methods in Figure 47.

In this section we discussed the empirical results of learning the influence structure between naïve Bayes models. For a large number of samples (> 300) our results show that learning using the GESS with the standard BIC score recovers the ground-truth structure better than using no influence structure or a random influence structure. According to the presented results in this section, learning the structure using the GESS algorithm provides a promising approach, in the remaining parts of this chapter we explore parameter and structure learning in the dynamic setting using GESS. In the next section, we firstly consider direct (Section 5.3.1) and delayed influence (Section 5.3.2) in HMMs.

7.3 LEARNING INFLUENCE BETWEEN HMMS

In this section we attempt to recover the structure of influence between processes. More specifically, we wish to reconstruct a probability distribution which describes the influence between a set of processes, each represented as a hidden Markov model (HMM). We firstly learn a set of independent HMMs to describe each process, and thereafter, optimise a structure score over possible structural configurations between these HMMs. The purpose of this section is to demonstrate empirically the recovery of influence between processes with respect to the direct and delayed structural assemble. Section 7.3.1 explores learning direct influence between HMMs, and Section 7.3.2 explores learning delayed influence between HMMs.

7.3.1 *Learning Direct Influence Between HMMs*

In Section 5.3.1, we described the implementation and applications of the direct influence assemble. We can implement the direct influence assemble between processes using independence assertions between two HMMs by adding a directed edge from every latent variable in $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ to every latent variable in $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, if the influence flows in this way. This is illustrated in Figure 48 which unrolls two HMMs, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, using the direct structural assemble from Definition 5.3.

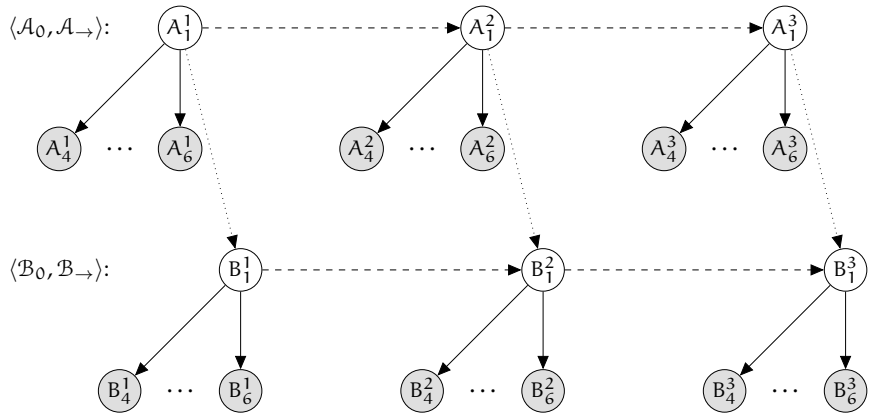


Figure 48: Two unrolled HMMs, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, as represented with 3 time-slices. The HMMs are connected with a direct structural assemble (dotted lines); the intra-time-slice edges are given by the solid lines; and the inter-time-slice edges are given by the broken lines. The unshaded variables are latent and the shaded variables are observable.

We present the performance of 6 direct dynamic influence networks (DiDIN) parameter and structure learning methods with respect to the generative ground-truth DiDIN's distribution. The performance is summarised in Figure 49, which shows the relative entropy to the generative ground-truth DiDIN (log-scale) over the number of training samples. The error bars show the standard deviation of the relative entropy over 10 trials and are given by the shaded regions.

The set-up of parameters for the ground-truth dynamic influence network was as follows. Each influence ground-truth network had: 10 hidden Markov models; 3 values per random variable; 5 time-slices per HMM; 15 edges in the influence structure; a max in-degree of 2 in the influence network structure; 5 observable variables per latent variable; and $\alpha = 1$.

Figure 49 shows the performance of the following parameters and structure learning methods. 'Random structure', which used a randomly generated structure for a DiDIN and learned the missing and observable parameters; 'No structure', which modelled each HMM as mutually independent to others and learned parameters; 'd-BIC with GESS', which used the d-BIC score with GESS and learned missing (using EM) and observable parameters (using MLE); 'd-AIC with GESS', which is the dynamic likelihood minus the total model dimension using GESS; 'GESS with PK', which used prior knowledge of the ground-truth distribution with GESS, this includes knowledge about the max in-degree and number of edges in the ground-truth; and finally, 'True structure', which used the ground-truth influence structure,

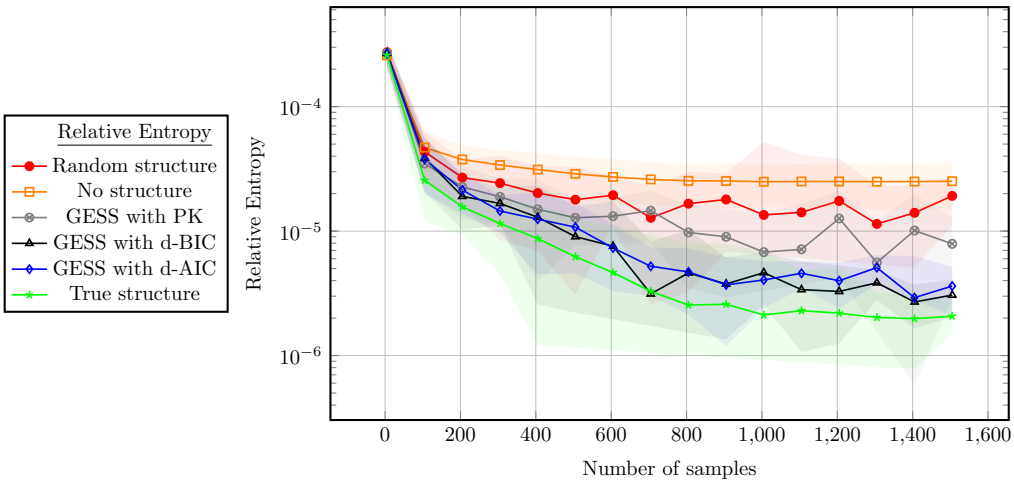


Figure 49: The performance of parameter and structure learning methods to recover direct influence between HMMs. The error bars show the standard deviation of the relative entropy over 10 trials and are given by the shaded regions.

but relearned missing and observable parameters. The setup for these parameter and structure learning methods are summarised in Table 3.

	Selection	Random structure	No structure	GESS with PK	GESS with BIC	GESS with AIC	True structure
1	α	1	1	1	1	1	1
5	No. edges	-	-	15	-	-	15
6	Max in-degree	3	-	3	-	-	-
7	No. observable var	5	5	5	5	5	5
8	Dirichlet prior	5	5	5	5	5	5
9	Parameter threshold	-	-	-	5000	5000	-
14	EM iterations	20	20	20	20	20	20
15	EM accuracy $(\mu\%, \sigma)$				(78%, 7.3)		
16	Likelihood score	-	-	Log-Like	Log-Like	Log-Like	-
17	Penalty score	-	-	-	BIC	AIC	-
18	Search iterations	-	-	50	50	50	-
19	No. random restarts	-	-	5	5	5	-
20	Tabu-list length	-	-	10	10	10	-

Table 3: A summary of the parameters used by the parameters and structure learning methods for recovering the direct influence between HMMs.

In Figure 49 modelling stochastic processes as mutually independent of all other processes, that is using no influence structure, performs worse on average compared to all other methods as the number of samples grows with respect to all other learning methods. This is caused by the inability to encode the influence between processes due to no conditional independence assumptions between processes that are expressed in each factor. Learning the parameters of a random structure performs worse on average than learning with prior knowledge of the true structure. This is intuitive since knowledge of the max in-degree and number of edges prevents us from over-fitting correlations in the data between variables. Although knowing the true structure generally performs better on average than all methods to recover the

ground-truth distribution, the two penalty-based scores (d-BIC and d-AIC) are similar in performance to fit to the training data.

Our results suggest that learning with prior knowledge increases our chances than learning with a random or no structure. Learning using the d-BIC and d-AIC perform better on average compared to all other methods, except with learning with the true structure. In the next subsection we explore delayed influence between HMMs.

7.3.2 *Learning Delayed Influence between HMMs*

A delayed assemble is a configuration which connects a family of temporal models (Section 5.3.2). It partly defines the parent sets for variables necessary to construct a delayed dynamic influence network (DeDIN). Recall from Definition 5.4, that in order to capture delayed influence structure between temporal models we need to insert dependencies between different time-points that span various models. How far back the dependencies between time-slices go depends on the influence structure of the distribution. More generally, we can describe delayed influence with respect to α-many previous time-slices for a family of temporal models. As a simple example, Figure 50 illustrates delayed influence between two unrolled HMMs, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, with $\alpha = 1$.

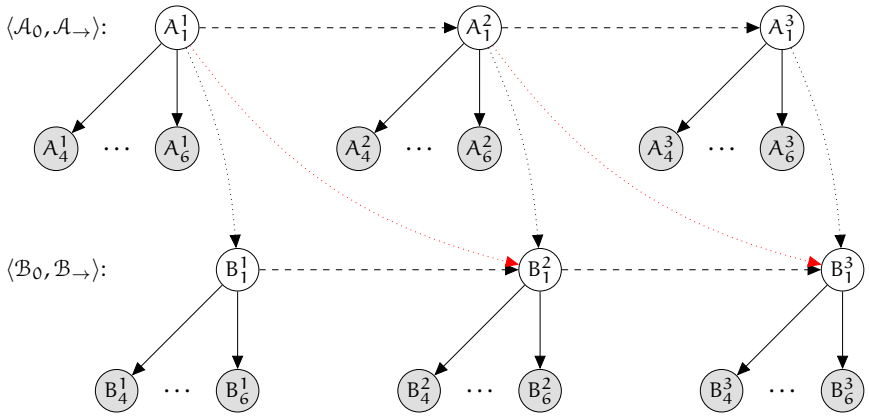


Figure 50: Two unrolled HMMs, $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ and $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, as represented with 3 time-slices. The HMMs are connected with a structural assemble ($\alpha = 1$) indicated by the dotted lines; the inter-time-slice edges are given by the broken lines; and the intra-time-slice edges are indicated by the solid lines. The unshaded variables are latent and the shaded variables are observable.

In this section we demonstrate the effects of learning DeDINs using GESS. We will show that the GESS algorithm can be used to learn the ground-truth distribution better than several baselines. We will also explore the time taken by our structure learning and parameter estimation methods.

Figure 51 shows the relative entropy to the generative ground-truth DeDIN (log-scale) and execution times over the number of training samples. The left log-scale y-axis shows the relative entropy to the ground-truth generative DeDIN; the x-axis shows the increase in sample size; and the right log-scale y-axis shows the execution

time of each learning method. The error bars show the standard deviation of the relative entropy over 10 trials and are given by the shaded regions.

The set-up of parameters for the ground-truth dynamic influence network was as follows. Each influence ground-truth network had: 10 hidden Markov models; 3 values per random variable; 5 time-slices per HMM; 15 edges in the influence structure; a max in-degree of 2 in the influence network structure; 5 observable variables per latent variable; and $\alpha = 2$.

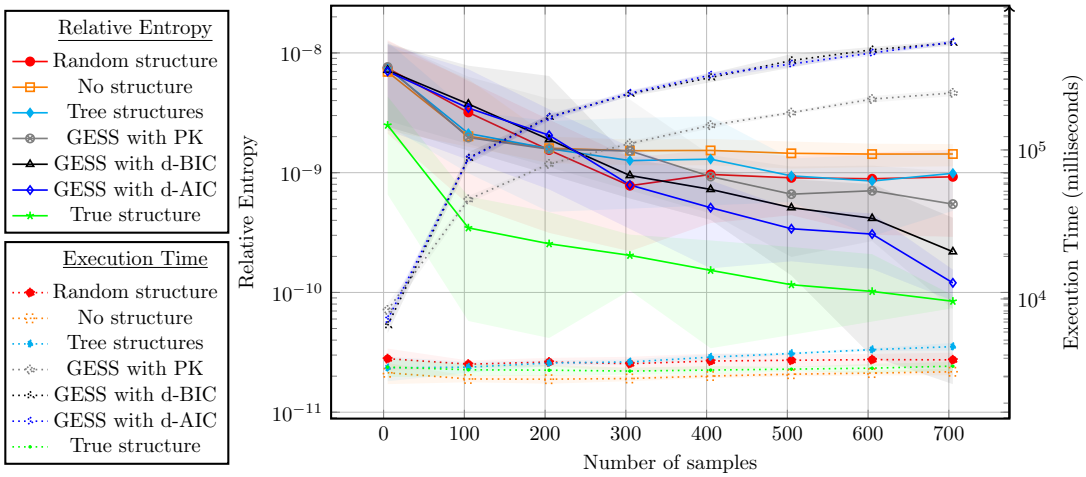


Figure 51: The performance of seven parameter and structure learning methods to learn delayed influence between HMMs.

Figure 51 presents the performance of seven DeDIN parameter and structure learning methods with respect to the generative ground-truth DeDIN's distribution. The seven learning methods are as follows: 'Random structure', which used a randomly generated structure for a DeDIN and learned the missing and observable parameters; 'No structure', which modelled each HMM as mutually independent to others and learned parameters; 'Tree structures', which learns a tree structure between the HMMs as well as learned missing (EM) and observable parameters; 'GESS with PK', which used the dynamic likelihood score with prior knowledge of the ground-truth structure (the max in-degree and number of edges in the ground-truth) and also learned parameters (latent and observable); 'd-BIC with GESS', which used the d-BIC score with GESS and learned both types of parameters; 'd-AIC with GESS', which used the d-AIC score with GESS and learned both types of parameters; and finally, 'True structure', which used the ground-truth structure, but relearned both types of parameters. The second y-axis shows the execution times for each learning method. The parameters of the seven learning methods are summarised in Table 4.

In Figure 51, with the exception of knowing the true structure, all learning procedures have similar performance for a small number of samples ($<$ 250 samples). This is since with fewer samples the uniform Dirichlet prior strength is still unchanged in each CPD value describing the model. For samples greater than 250, we note that using no structure (orange) yields the lowest performance on average since, no matter how much of data is provided, the model does not include the conditional independence assumptions which are capable of encoding the dynamic influence structure.

Using tree structures (blue) and random structures (red) have similar performance, although we can guarantee that tree structures are sparse whereas random structure

	Selection	Random structure	No structure	Tree structure	GESS with PK	GESS with BIC	GESS with AIC	True structure
1	α	2	2	2	2	2	2	2
5	No. edges	-	-	-	15	-	-	15
6	Max in-degree	3	-	-	3	-	-	-
7	No. observable var	5	5	5	5	5	5	5
8	Dirichlet prior	5	5	5	5	5	5	5
9	Parameter threshold	-	-	-	-	5000	5000	-
14	EM iterations	20	20	20	20	20	20	20
15	EM accuracy $(\mu\%, \sigma)$				$(75\%, 6.7)$			
16	Likelihood score	-	-	Log-Like	Log-Like	Log-Like	Log-Like	-
17	Penalty score	-	-	-	-	BIC	AIC	-
18	Search iterations	-	-	-	50	50	50	-
19	No. random restarts	-	-	-	5	5	5	-
20	Tabu-list length	-	-	-	10	10	10	-
23	α	2	2	2	2	2	2	2

Table 4: A summary of the parameters used by the parameters and structure learning methods for recovering the delayed influence between HMMs.

may be dense. After 400 samples, learning with prior knowledge of the true structure from the ground-truth structure performs better than random, no structure and tree structures. All the three penalty-based learning procedures ('d-BIC with GESS', 'd-AIC with GESS', and 'GESS with PK') out-perform our baselines after 400 samples. The d-AIC performs better on average than the d-BIC penalty score after 250 samples.

With regard to the execution time, all three penalty-based procedures provide the highest computational burden, whereas, learning a tree-structure, selecting a random structure, using no structure, or being given the true structure can be done in relatively constant time. Learning with some knowledge of the ground-truth DIN can be learned faster than using d-AIC or d-BIC, which relatively have the same average execution time.

In this section we presented the results of learning the structure of DeDINs. We noticed that although the penalty-based procedures provide better density to the ground-truth distribution, these procedures take a longer average execution time than the other methods tested. In the next section we move to generalise the performance of these learning methods to recover general influence networks between HDBNs in more difficult learning problems.

7.4 LEARNING GENERAL HIERARCHICAL DYNAMIC BAYESIAN NETWORKS

We have thus-far demonstrated the effectiveness of the proposed GESS algorithm compared to several baselines on HMMs. In some scenarios, we deal with complex stochastic processes with more than one latent variable each described by a set of observations. In these scenarios we need to consider a more suitable description for each process. In this section we consider recovering influence between processes represented by hierarchical dynamic Bayesian networks (HDBNs).

Recall from Chapter 5 that we can capture delayed influence between general HDBNs. As a reminder, consider Figure 34 which depicts an example of general (delayed) influence where we insert dependencies between two unrolled HDBNs, that is, from $\langle \mathcal{A}_0, \mathcal{A}_{\rightarrow} \rangle$ to $\langle \mathcal{B}_0, \mathcal{B}_{\rightarrow} \rangle$, where $Q = 2$; $R = 2$; and $T = 3$.

Figure 52 shows the relative entropy to the generative ground-truth DeDIN (log-scale) and execution times over the number of training samples for recovering delayed influence between HDBNs with two latent layers. The left log-scale y-axis shows the relative entropy to the ground-truth generative DeDIN; the x-axis shows the increase in sample size; and the right log-scale y-axis shows the execution time of each learning method. The error bars show the standard deviation of the relative entropy over 10 trials and are given by the shaded regions.

The set-up of parameters for the ground-truth dynamic influence network was as follows. Each influence ground-truth network had: 10 hierarchical dynamic Bayesian networks (HDBNs); 3 values per random variable; 5 time-slices per 2-layered HDBN; 15 edges in the influence structure; a max in-degree of 2 in the influence network structure; and 5 observable variables per latent variable.

Figure 52 presents the performance of seven DeDIN parameter and structure learning methods with respect to the generative ground-truths DeDIN distribution. The first three learning methods are as follows. 'Random structure', which used a randomly generated structure for a DeDIN and learned the missing and observable parameters; 'No structure', which modelled each HDBN as mutually independent to others and learned parameters; and 'Tree structures', which learned a tree structure between the HDBNs as well as learned missing (EM) and observable parameters (BE).

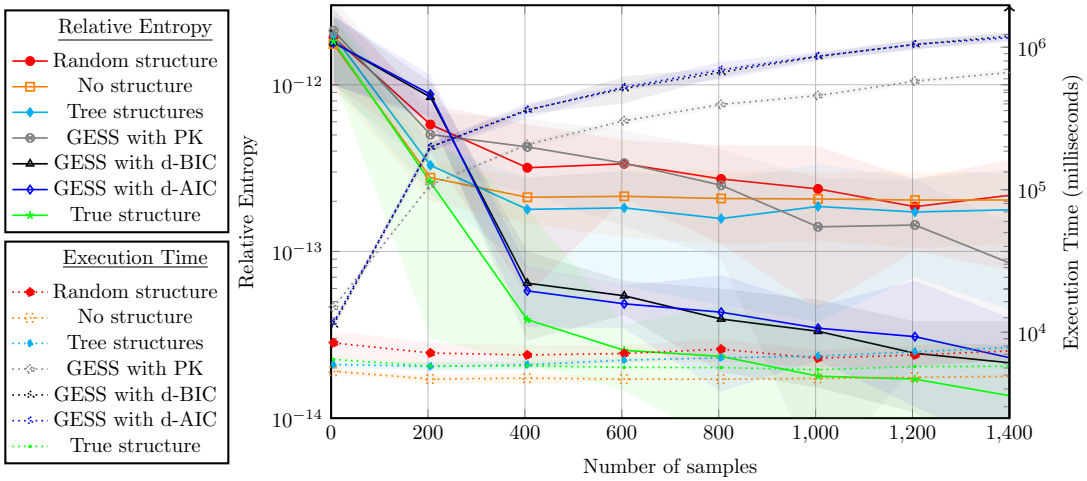


Figure 52: The performance of seven parameter and structure learning methods for hierarchical dynamic Bayesian networks.

The fourth learning method is 'GESS with PK', which used prior knowledge of the ground-truth structures with learned latent and observable parameters. This method used the dynamic likelihood score but restricted recovered structures with an in-degree greater than 3 and more edges than the ground-truth. Figure 53 shows an example of this structure learning scenario. The x-axis shows the structure search iterations (i), and the y-axis shows the dynamic score value of the selected DIN at each iteration. There are two variables in this learning scenario shown in the figure:

the value of the ground-truth structure score (orange line) and the total score of the current network at iteration i (just the dynamic likelihood relative to the data) is given by the green line. The figure shows how after only 6 iterations, the dynamic likelihood score has monotonically increased towards the ground-truth's dynamic likelihood score. From this result, we can observe how prior knowledge about a the ground-truth structure allows us to learn a suitable structure. However, in the figure we note that the learned structure does not have the same score as the ground-truth structure. This is because the EM algorithm is not able to recover the complete correct set of latent class labels for each data instance, which reduces the likelihood of learned network relative to the test data.

The fifth learning task in Figure 52 is the 'd-BIC with GESS', which uses the d-BIC score with GESS and learned both types of parameters. Figure 54 shows an example of this structure learning scenario. The x-axis is the structure search iterations (i), and the y-axis is the log-scale score value for the selected transition DIN. There are four parameters in this learning model: the value of the ground-truth total structure score (orange); the dynamic likelihood score of the recovered network at iteration i (blue); the penalty-score of the recovered network at iteration i (red); and the total sum of the dynamic likelihood and penalty scores for the recovered network at iteration i (green). We note that the total score of the recovered network monotonically increases before reaching a local optimum at iteration 10, it then performs a random restart and continues to monotonically increase. It appears that up to iteration 22 the structure score cannot improve, however, after one more restart it finds a better optimum at iteration 27. The recovered network appears to approach the ground-truth network structure score. From these results we see that at every transition DIN the d-BIC score increases, except when random restart are performed. Therefore, unlike in Figure 53 (ie. when using some prior knowledge), the d-BIC is able to inform the GESS algorithm about when a more complicated network structure is prefered with respect to the fit to data (as suggested by the properties of d-BIC in Section 5.2.2).

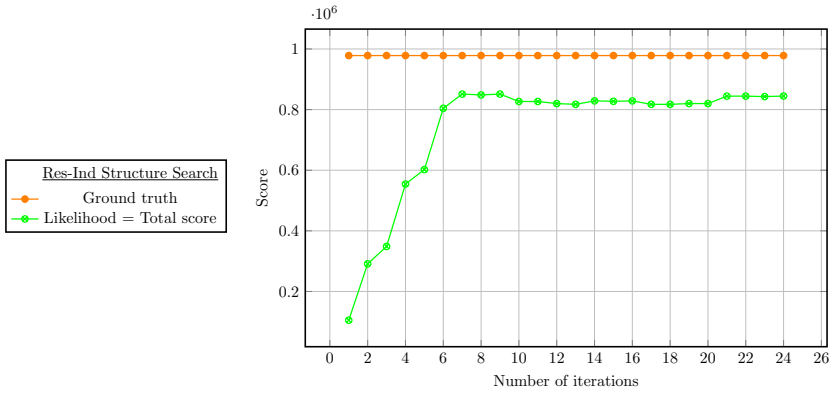


Figure 53: The value of structure scores for the GESS with PK learning method. The y-axis is the score value and the x-axis represents the increase in structure search iterations.

The sixth learning method in Figure 52 is the 'd-AIC with GESS' which used the d-AIC score with GESS and learned both types of parameters. Finally, the seventh learning method in Figure 52 is true structure with learned parameters, which used the ground-truth structure but relearned both types of parameters. The parameters of the seven learning methods are summarised in Table 5.

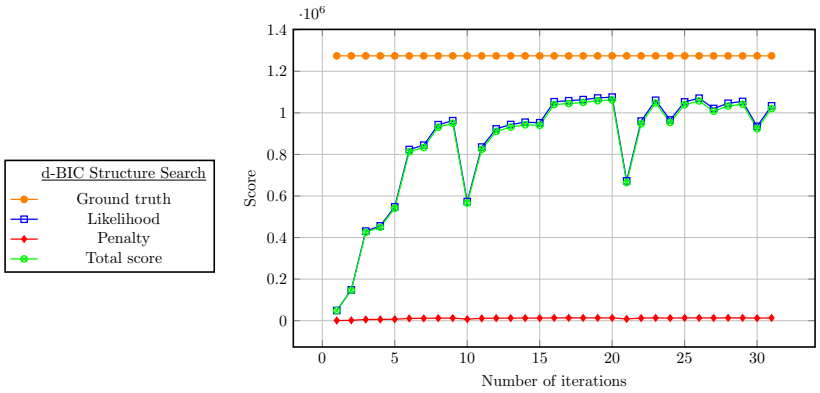


Figure 54: The value of structure scores for the GESS with d-BIC learning method. The y-axis is the score value and the x-axis represents the increase in structure search iterations.

	Selection	Random structure	No structure	Tree structure	GESS with PK	GESS with BIC	GESS with AIC	True structure
1	α	2	2	2	2	2	2	2
5	No. edges	-	-	-	15	-	-	15
6	Max in-degree	3	-	-	3	-	-	-
7	No. observable var	5	5	5	5	5	5	5
8	Dirichlet prior	5	5	5	5	5	5	5
9	Parameter threshold	-	-	-	-	5000	5000	-
14	EM iterations	20	20	20	20	20	20	20
15	EM accuracy $(\mu\%, \sigma)$				$(76\%, 10)$			
16	Likelihood score	-	-	Log-Like	Log-Like	Log-Like	Log-Like	-
17	Penalty score	-	-	-	-	BIC	AIC	-
18	Search iterations	-	-	-	50	50	50	-
19	No. random restarts	-	-	-	5	5	5	-
20	Tabu-list length	-	-	-	10	10	10	-
23	α	2	2	2	2	2	2	2

Table 5: A summary of the parameters used by the parameters and structure learning tasks for recovering the delayed influence between HDBNs.

In Figure 52 we note that using no structure (orange), tree structures (blue), random structure (red), and learning with knowledge of the maximum order in-degree from the ground-truth structure performs similarly with respect to their relative entropy to the ground-truth distribution. In particular, the p-value from a t-test for using GESS with PK and using a random structure is $0.796 > 0.05$ which fails to reject the null hypothesis. However, knowledge of the in-degree performs better on average then these methods for a large number of samples (> 1000). Our penalty-based scores perform better on average than all other learning methods, with the exception of learning using the true structure. In particular, the p-value of learning using GESS with d-BIC and using a random structure is statistically significantly $0.04 < 0.05$. We also note that the t-test for the d-BIC and d-AIC penalty based learning methods fail to reject the null hypothesis (i.e. that their samples are not significantly different).

The second y-axis in Figure 52 is the execution times for each learning method. In terms of execution time, all three penalty-based procedures ('d-BIC with GESS', 'd-AIC with GESS', and 'GESS with PK') yield the highest computational time, whereas,

learning a tree-structure, selecting a random structure, using no structure, or being given the true structure can be done in constant time. Learning with prior knowledge can be done faster than using d-AIC or d-BIC, which have roughly the same execution time.

Once again, our results suggest that using our penalty-based procedures with a large amount of samples perform better on average than the other tested methods. However, the main limitation of using these penalty-based procedure are their execution times. An overview of the complexity of these penalty-based procedures are discussed in Section 6.5.3. In the next section we will discuss the high-level interpretation of our results.

7.5 DISCUSSION OF RESULTS

In this section we provide a high-level interpretation of the results from the previous section. This discussion section is structured as follows. In Section 7.5.1 we discuss the ability of our learning procedures to recover the ground-truth structure; Section 7.5.2 discusses the execution time of our learning procedures; Section 7.5.3 discusses which learning procedures are appropriate given the availability of training data; Section 7.5.4 discusses learning influence models given domain knowledge; Section 7.5.5 discusses the advantages of learning influence models with penalty-based procedures; Section 7.5.6 discusses the consequences of learning with latent variables; and finally, Section 7.5.7 discusses the scalability of our learning procedures to more difficult influence learning methods.

7.5.1 *Ability to Recover the Ground-truth*

We see in all the provided results (Figure 47, 49, 51, and 52) that it is possible to learn an influence structure and parameter setting which appears to resemble the ground-truth DIN as we increase the number of samples. These results suggest that the structure learning problem is not intrinsically more difficult than the parameter estimation problem for influence networks of small sizes. Certainly, learning from synthetic data has a much stronger signal for the correlation between variables than in real data. However, the general performance of learning procedures, in Figure 47, 49, 51, and 52, are promising.

The three penalty-based learning procedures ('d-BIC with GESS', 'd-AIC with GESS', and 'GESS with PK') generally perform better to recover the ground-truth network in all the provided learning scenarios. In particular, Figure 49 shows how the three penalty-based learning procedures generally perform better on average than using a random and empty influence structure. In Figure 51, which is the delayed case, the three penalty-based learning procedures maintain the ability to recover the ground-truth distribution better than any of the other tested learning procedures. It is not clear in Figure 51 whether the GESS with the d-AIC or the GESS with the d-BIC score performs better. When using HBDNs in the delayed case, Figure 52 suggests that both procedures give roughly the same likelihood with respect to the ground-truth. This might be since we might lose information about the correlation between

models in settings when we have more latent layers in the description of the HDBN representing each stochastic process. As expected, knowing the true structure gives us the most information and thus performs better on average than all the other methods as the number of observations increase.

7.5.2 *Execution Time to Recover the Ground-truth*

All of the figures provided on the execution time of our structure learning methods (Figure 47, 49, 51, and 52) are consistent in approximating the rate at which the execution time grows with training samples. These results are also supported by the theoretical complexities derived in Section 6.5.3. The GESS with d-AIC and d-BIC scores grow exponentially given the number of training samples as predicted in the theoretical section. Although, caching sufficient statistics does reduce the computational burden, the proposed search procedure using these penalty-based scores requires traversing and evaluating the scores of a large amount of DINs which need to be relearned at every search step.

The learning procedures with approximately linear growth rates, evident in Figure 52, are selecting random structures, using no structure, learning tree-based structure, and given the true structure. Although the complexity of these methods are less of a computational burden than the penalty-based methods, they all perform worse on average when approximating the true distribution. The execution time taken to recover the influence distribution for GESS with the restriction on the in-degree grows slightly slower than the GESS with d-AIC and d-BIC. However, it performs worse on average in terms of relative entropy to the ground-truth. This is because learning the parameters for sparse representations (with the exception of the random structure which could be dense) can be done relatively quickly due to minimised fragmentation.

Constructing a tree structure costs more time than using no structure. Learning a true structure, no structure, or random structure are just parameter learning methods. However, random graphs are denser than tree structures which are denser than no influence structure. Given the number of dependences in the learning model, the parameter learning task complexity increases due to fragmentation. Thus, from the fastest to slowest learning procedures for these given structure learning methods we have no structure, tree structure, and lastly random structure (or the respective true structure).

7.5.3 *Availability of Data*

In non-dynamic models (Figure 47), the relative entropy of an I-equivalent structure to $\mathcal{I}(\mathcal{G}^*(\mathbb{I}^{\mathcal{G}}))$ can still recover the true distribution, $\mathrm{P}^*(\mathbb{I}^{\mathcal{G}})$. As we see, in Figure 47, the d-AIC and d-BIC scores tends to correctly recover the distribution between each naïve Bayes model (NBM) compared to random guesses and over mutually independent models (no structure), except for when we have very little training instances.

In Figure 52, learning a tree network takes less time than learning a random structure and it provides a better generalisation of the ground-truth. This is because a tree structure summarises the most important influence dependencies between any two temporal networks due to learning the maximum weighted spanning tree. From these results we see that when we have fewer samples (less than 250) we are better-off not using any structure, since fewer parameters allow us to generalise better since we have more data to learn each parameter than in cases where we have a large number of parameters with little data. However, when we do have a sufficient amount of data, then a random structure gives us more information than no structure, and thus the random structure performs better.

7.5.4 *Domain Knowledge*

In most practical applications we do not know much about our domain. However, in very few situations it might be the case where we obtain reliable information about some domain knowledge. For example perhaps if we know that only a maximum of 2 processes can influence a process based on isolating and accessible environments in the domain, then using a learning procedure that restricts the in-degree of influence scenarios might provide a more efficient and time-effective solution. Using a heuristic method with a restriction to the in-degree still might provide a better option given that this problem is intractable.

The results of our presented experiments suggest that providing some domain knowledge about the influence structure is useful since it can be learned quicker than our d-BIC and d-AIC with GESS procedures, and it performs better on average than a random and empty influence network.

7.5.5 *Penalty Scores*

The sensitivity of the d-BIC and d-AIC scores to judge when to restrict a structure guides the selection of independence assumptions, with roughly the same execution time, and performs better on average than all tested methods for a large number of samples in Figure 52.

On the one hand, the d-BIC score considers both the number of samples and the structural complexity, on the other hand, the d-AIC score only considers complexity (or model dimension). The d-BIC score does not consider complicated structures when given too little data, and thus the d-AIC score provides more edges which gains a larger likelihood to the data. This is evident at later iterations in Figure 52 when we see that the d-BIC score (at roughly 700 samples) performs better on average than the d-AIC score. Overall, both of these methods provide an excellent example of the theoretical properties and effectiveness of the discussed penalty-based structure scores.

Another effective property of using dynamic score-based penalties (e.g. the d-BIC and d-AIC) as opposed to the hard restriction (e.g. restricting the in-degree) is also seen in Figure 53, 58, and 54. Each of these graphs indicate scores at various search

iterations. The restriction of the in-degree causes the search space to be small and independent of the data, whereas, the score-based penalties allow the data to persuade and inform the learning procedure about structures which are worth considering (e.g. Figure 58 at 25 iterations, and Figure 54 at 15 iterations).

RESTRICTING PARAMETERS A necessary penalty restriction to reduce the computational burden is to restrict the total number of parameters used to express a DIN. Since the number of parameters when learning a DIN can grow exponentially as a function of the number of parents per variable, it is necessary to restrict the learning procedure to use the amount of memory available to the computer doing the computation. This can be done by simply limiting the number of parameters that can be used to express the DIN. Doing so results in a reliable termination of learning procedures by providing a bound to the total time to learn a model. This bound is evident in Figure 52 and Figure 51 (as well as in Figure 55(d) and Figure 55(e)), where the execution time begins to plateau as the graph sizes of both dense and sparse graphs increase.

7.5.6 *Learning Latent Parameters*

From the analysis done in Section 6.5.3, we see that using simple optimisation techniques makes our search for the true structure more manageable. However, introducing latent variables dramatically increases the search space and thus the complexity of the problem. This is because of the multiple optimum in the likelihood function caused by missing data (Section 2.3.1.3).

The orange line in Figure 53, 58, and 54, indicates the ground-truth's total score with respect to the penalty-based score used. Since the EM algorithm used in our experiments is never able to fully recover all of the latent class labels in the data, we could never achieve this score in our procedures. However, even with this limitation, we find that empirically as we increase the number of samples our recovered structures resemble the ground-truth distribution.

Another difficulty arises when we increase the number of latent layers in the HDBN models which describe each process. The more latent variables we have, the less likely we are to recover the ground-truth distribution due to the accumulative error of our EM algorithm as we recover the class labels of our latent variables.

7.5.7 *Generalisation of Learning Tasks*

We lastly discuss the scalability of our structure learning procedure to recover the ground-truth distribution as we increase the difficulty of the learning problem. We will demonstrate this by exploring the sensitivity of the initialization parameters. Figure 55 shows the performance of six learning methods as the environments of the learning problem changes. Figure 55(a) shows the relative entropy to the ground-truth distribution for several learning methods as the number of bins describing each

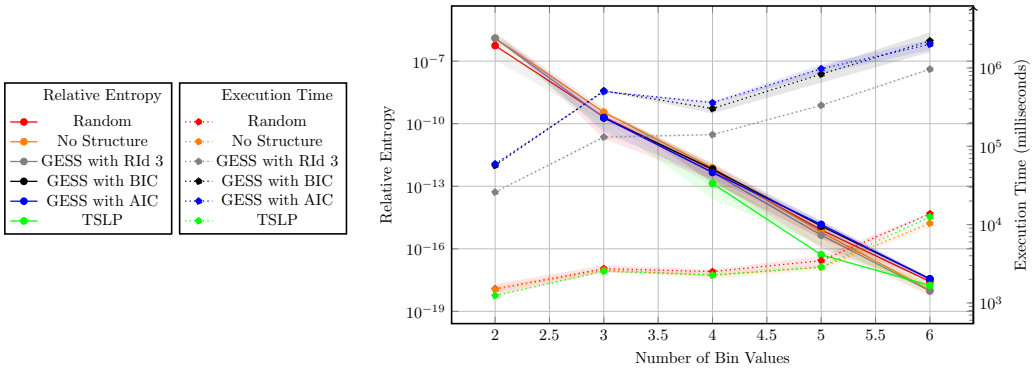
random variable increases. Figure 55(b) shows the relative entropy to the ground-truth distribution for several learning methods as the number of time-slices in the temporal model used to describe each trajectory increases. Figure 55(c) shows the relative entropy to the ground-truth distribution for several learning methods as the number of given observations describing the latent variables increases. Figure 55(d) shows the relative entropy to the ground-truth distribution for several learning methods as the size of sparse influence graphs increases. Figure 55(e) shows the relative entropy to the ground-truth distribution for several learning methods as the size of dense influence graphs increases.

In all of the figures in Figure 55 we see that as the difficulty of the learning methods increases, the relative entropy exponentially decreases as the execution time exponentially increases. We also noted that (for our experiments) the general time used to recover the model increases exponentially up until a point when it appears to increase constantly. Examples of this can be seen in Figure 55(d) after sparse sizes of 12 and Figure 55(e) after dense sizes of 10.

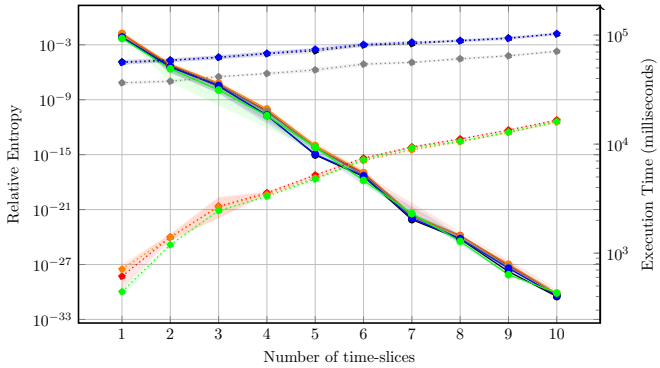
In Figure 55 we provide the performance of our method in several learning environments. In all of these learning environments we see that our learning methods scale significantly well with respect to the learning problem. This tells us two things:

- at every increment of the learning problem becoming more difficult on the x-axis, the relative entropy of the learning procedures decrease.
- as the joint assignment becomes larger with more terms, the value of the joint distribution decreases.

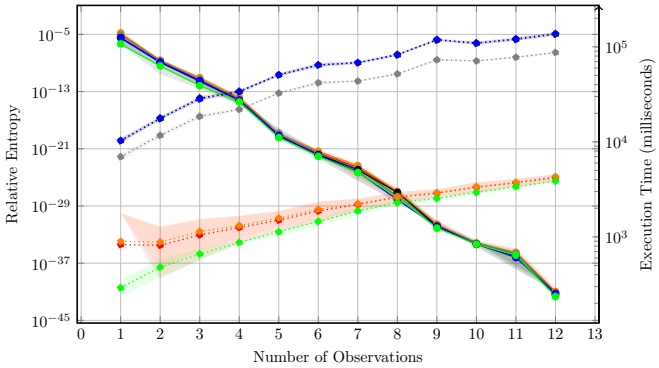
Finally, Figure 55 suggests that as the learning problem's difficulty increases, all learning performance increases exponentially (be it parameter or structure learning methods). In the next chapter we conclude this work by providing a summary of this work, our main contributions, and future directions.



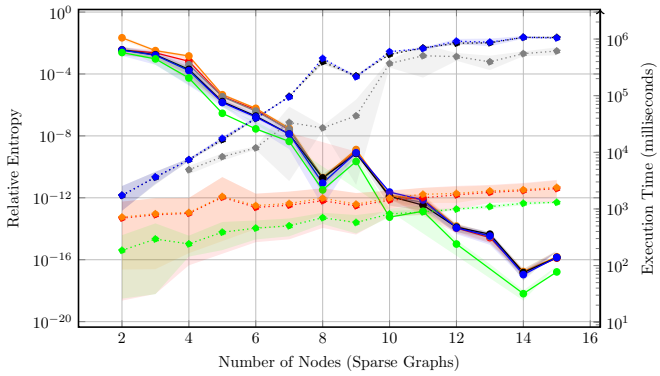
(a) The performance of parameter and structure learning tasks as the number of bin values increase.



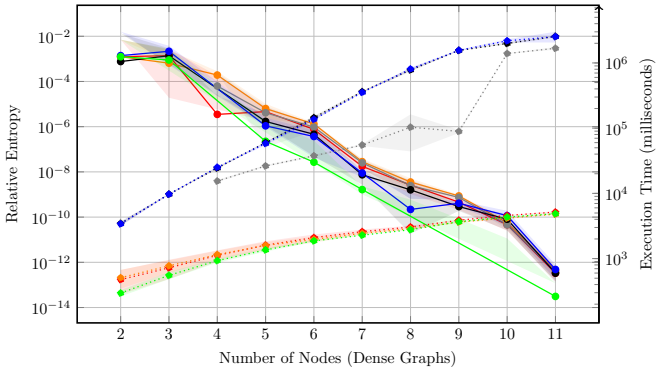
(b) The performance of parameter and structure learning tasks as the time-slices increases.



(c) The performance of parameter and structure learning tasks as the number of observations increase.



(d) The performance of parameter and structure learning tasks as the sparse graph sizes increase.



(e) The performance of parameter and structure learning tasks as the dense graphs sizes increase.

Figure 55: The performance of several learning procedures as the complexity of the learning problem increases. (a) shows the relative entropy to the ground-truth distribution for several learning methods as the number of bins describing each random variable increases. (b) shows the relative entropy to the ground-truth distribution for several learning methods as the number of timeslices in a temporal model used to describe each trajectory increases. (c) shows the relative entropy to the ground-truth distribution for several learning methods as the number of given observations describing the latent phenomenon increases. (d) shows the relative entropy to the ground-truth distribution for several learning methods as the size of sparse influence graphs increase. (e) shows the relative entropy to the ground-truth distribution for several learning methods as the size of dense influence graphs increase. The error bars show the standard deviation of the relative entropy over 10 trials and are given by the shaded regions.

8

CONCLUSION

In this thesis we provide the first score-based structure learning algorithm to reconstruct a *dynamic influence network* (DIN) between a set of partially observable stochastic processes. Alongside the mathematical development of models and inference algorithms, we empirically demonstrated the effectiveness of our approach.

Learning a DIN is useful in situations when we are given temporal features as data and asked to deduce the probability distribution which explains the interaction between partially observed processes. Constructing an influence structure from temporal incomplete data, that is induced by a set of temporal observations, appears in practical applications where we wish to perform density estimation or knowledge discovery.

This conclusion chapter is structured as follows: Section 8.1 provides an overview of the thesis content and shows how all the major sections fit together; Section 8.2 discusses influence model selection for knowledge discovery and density estimation; Section 8.3 provides a summary of the major contributions of this thesis; and finally, Section 8.4 provides relevant future directions which can be explored to improve the performance of the general algorithm to recover influence in the dynamic setting.

8.1 SUMMARY

Revising the high-level proposed architecture from Chapter 1 (the diagram is repeated), we can now give a more detailed perspective on the high-level learning procedures that were developed throughout this thesis.

In (i), we input the processes to learn influence between; In (ii), we learn each stochastic process independently as a temporal model using Bayesian estimation (BE) and expectation maximisation (EM) to maximise the likelihood function for latent variables (Section 6.3). We have explored two different representations of processes, the hidden Markov model (HMM)(Section 2.2.2.3) and the hierarchical dynamic Bayesian network (HDBN) (Section 5.2.1.3). We have seen the effects of using different latent layers in the representation of these models and how the abstraction of observations can be achieved (Section 7.5.6). Although the focus of this thesis was on general structure learning between temporal models, describing the representation for each stochastic processes before learning could produce better results since different processes may have distinct structural dependencies between variables which cannot be captured using a generic template network. The description of these models may play an important role in the discovery of the ground-truth distribution.

In (iii), we compute the structure score of an influence network using a scoring function. We have seen four examples of scoring functions for DINs (Section 5.2): the

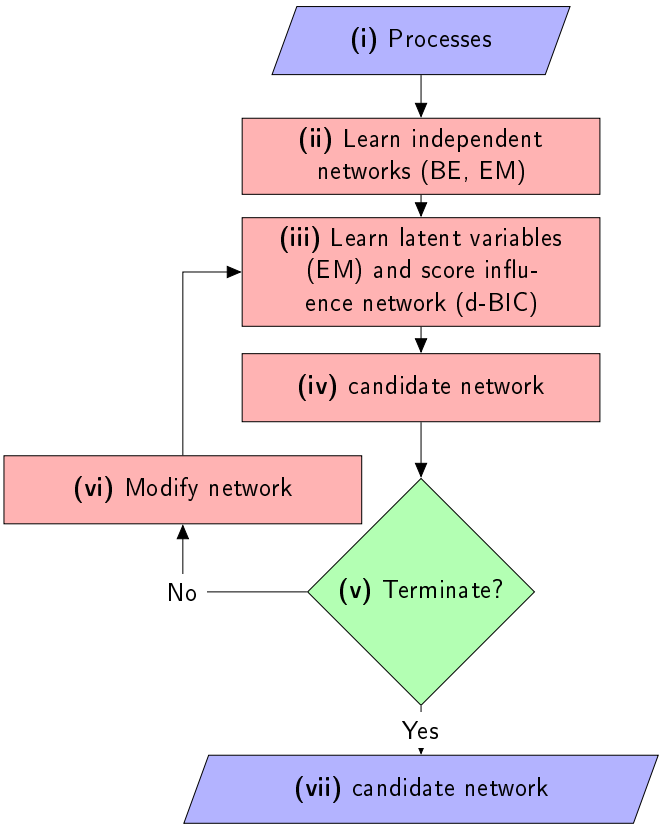


Figure 56: An overview of the proposed algorithm to recover influence between stochastic processes represented as temporal networks.

d-BIC, d-AIC, restricting the in-degree, and the standard dynamic likelihood score. All of these scoring functions have properties which are effective in various domains. For example, using the likelihood score alone can build a tree-based structure to summarise the most important dependency relations between temporal models. These scores have been compared empirically in Chapter 7 showcasing the precise value of their properties in various contexts.

We note the major difficulty of this problem lies in the representation of the latent components of the influence network. Therefore, we develop a learning procedure that learns a probability distribution that describes interactions between processes which manifest in the temporal observations that describe each process (Section 6.3.1). By doing so we take advantage of significant computational savings (Section 6.5.3). In (iii), we relearn the parameters for the model (with the new independence assertions) which gives us the candidate network (iv). These candidate networks are constructed using an influence assemble relation (Section 5.3) which connects families of dynamic Bayesian networks. We constructed various influence structures (both direct (Section 5.3.1) and delayed (Section 5.3.2)), and showed how the performance of using sparse representations allowed for a better generalisation to the ground-truth structure .

In (vi), we performed an operation, by changing the graph structure which encodes the distribution, to improve the network fit to data (Section 6.5.1). Each operation was a choice between an edge addition, removal, or deletion. All of these operations were with respect to the selected influence assemble used. By using various operations we

performed greedy hill-climbing algorithms in-order to optimise the dynamic scoring function with respect to the data (Section 6.5.2). We iteratively performed steps (ii), (iii), (v) until we could not improve the dynamic score for the DIN structure. The DIN structure with the best score was selected. We also provided ways to promote computational saving for structure search procedures (Section 6.5.3).

8.2 LEARNING A DIN FOR KNOWLEDGE DISCOVERY AND DENSITY ESTIMATION

The significance of learning a DIN structure depends on our learning objective. If one is attempting to discover exactly the ground-truth network structure, which involves stating precisely $\mathcal{I}(\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}))$, then we should concede that there exists many *perfect maps* for $P^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}})$ which can be recovered from $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}}$.

It is understood that recognising $\mathcal{I}(\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}))$ from $\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}})$'s set of structures which give the same fit to the data is *not identifiable* from $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}}$ since each I-equivalent structure produces the same likelihood for $\mathcal{D}_{\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}}$. This is evident in the analysis from Section 4.2.5, where we see that the same set of independence assumptions can give rise to different influence structures. Therefore, if our goal is knowledge discovery, we should instead try to recover $\mathcal{G}^*$'s I-equivalence class. This is difficult as data sampled from $P^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}})$ does not perfectly and uniquely reconstruct the independence assumptions of $\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}))$.

Obviously, selecting a random structure is unlikely to tell us much about the ground-truth structure. However, tree structure attempts to summarise the most important dependencies. General graph learning algorithms can help us develop an I-equivalence class which we can use, along with more meta-data, to select a network for knowledge discovery.

Alternatively, one could also attempt to learn a DIN for *density estimation*, i.e. to estimate a statistical model of the underlying distribution $P^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}})$. Such a model can be used to reason about new data instances.

On the one hand, if we capture more independence assertions than those specified in $\mathcal{I}(\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}))$, we could still capture $P^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}$ using some setting of our recovered DIN's parameters. However, our selection of more independence assumptions, rather than fewer in $\mathcal{I}(\mathcal{G}^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}))$, could result in *data fragmentation*. On the other hand, selecting too few edges will result in not capturing the true distribution $P^*(\langle \mathbb{I}_0, \mathbb{I}_\rightarrow \rangle^{\mathcal{G}}$, but will however provide a sparse structure that avoids data fragmentation. Generally, the latter case is preferred in density estimation for Bayesian networks since it provides better generalisation to new instances through a sparser representation [Koller and Friedman 2009]. In the next section we revise the major contributions of this research.

8.3 MAJOR CONTRIBUTIONS

We provided the following contributions.

1. The notion of influence between processes. This includes the formulations of direct and delayed influence (Chapter 4).

2. Several scoring function for dynamic influence networks by extending and adapting traditional scores for random variables along with their key properties (Chapter 5).

3. The notion of a structural assemble to relate temporal models for dynamic influence tasks (Chapter 5).

4. A learning procedure to recover the influence structure between temporal models with latent variables. We further extend the local search procedures to use assembles that link temporal models meaningfully while preserving decomposability and score-equivalence required for a manageable search (Chapter 6).

5. We provide empirical evidence for the effectiveness of our method with respect to a generative ground-truth distribution (Chapter 7).

8.4 FUTURE WORK

There are many future directions that can improve the performance of the high-level algorithm presented in Figure 56. We can attempt to improve the description of each stochastic process in step (ii). This can be done using HDBN or perhaps more sophisticated dynamic Bayesian networks with a causal interpretation over the characteristics of each process. An alternative option is to learn the structure of the dynamic Bayesian network which best describes the temporal observations.

We may use more sensitive dynamic scoring functions in (iii) which incorporate structural characteristics of the model or perhaps more emphasis on dynamic scoring functions as our experiments indicated incoherence between available memory and sensitivity of the score to training samples and number of independent parameters. We may also improve the performance of the EM algorithm (iii) used to recover the original cluster assignments. Perhaps the use of hard-EM or soft-EM could traverse the search space differently in-order to ensure a better recovery of the original class assignments.

Perhaps more computational power is necessary to allow for convergence by increasing the number of structure search iterations (v). Also, with more computational power we can explore influence structures between more temporal networks. Finally, one could improve the type of operators used by the learning procedure in (vi). Perhaps operators which take larger steps in the search space will allow for faster convergence by exploring search spaces faster.

Part I

APPENDIX

A

ALGORITHMS

Algorithm 7 Generate Independent Factors

1: **procedure** GENINDEP($\mathcal{D}_{\mathcal{L}}$, $\mathcal{D}_{\mathcal{O}}$, $\mathcal{O} = (O_1, \ldots, O_j)$, $\mathcal{L} = (L_1, \ldots, L_k)$, Var $= (O_1, \ldots, O_j, L_1, \ldots, L_k)$, „, NumBi)
2: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}]' = \{\}$
3: **for** $i = 1 \rightarrow Var.Length$ **do**
4: **if** $Var[i] \in \mathcal{O}$ **then**
5: $\mathcal{D}_x =$`Extract`$(\mathcal{O}.$`posOf`$(Var[i]), \mathcal{D}_{\mathcal{O}}, 0, \mathcal{O}.Length)$
6: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}]'.$`pos`$(i) = \mathbb{F}(\mathcal{D}_x, NumBi[i], Var[i],$`"OBSERVABLE"`$, \alpha[i])$
7: **else if** $Var[i] \in \mathcal{L}$ **then**
8: $\mathcal{D}_x =$`Extract`$(\mathcal{L}.$`posOf`$(Var[i]), \mathcal{D}_L, 0, \mathcal{D}_L.Length)$
9: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}]'.$`pos`$(i) = \mathbb{F}(\mathcal{D}_x, NumBi[i], Var[i],$`"LATENT"`$, \alpha[i])$
return $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}]'$

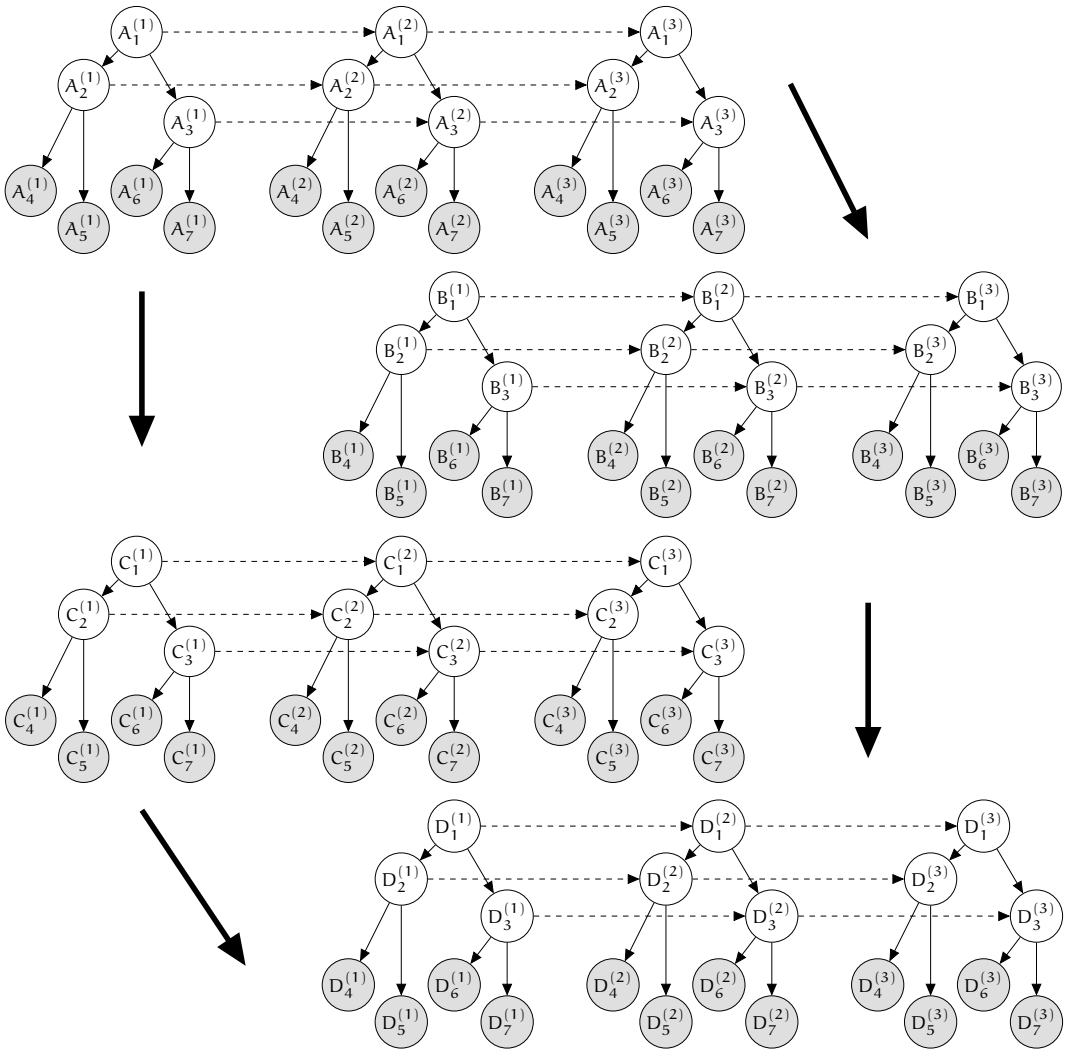


Figure 57: An illustration of a dynamic influence network between four hierarchical models with 3 timeslices.

Algorithm 8 Maximization

1: **procedure M-step**([$\mathcal{D}_{\mathcal{L}}$, $\mathcal{D}_{\mathcal{O}}$, $\mathcal{O} = (O_1, \ldots, O_j)$, $\mathcal{L} = (L_1, \ldots, L_k)$, Var, $\mathcal{I}_\ell(\mathcal{H})$, ¸, $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}]$)
2: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}]' = [\,]$
3: **for** $i = 0 \rightarrow (j + k)$ **do**
4: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}]'[i] = [\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}][i]$
5: **for** $\forall a \in \mathcal{I}_\ell(\mathcal{H})$ **do**
6: $x = getVar(a)$
7: $index = Var.posOf(x)$;
8: **if** $a has no dependencies$ **then**
9: $\mathcal{D}_a = null$;
10: **if** $x \in \mathcal{O}$ **then**
11: $\mathcal{D}_a =$Extract$(\mathcal{O}.posOf(x), \mathcal{D}_{\mathcal{O}}, 0, \mathcal{D}_{\mathcal{O}}.length)$;
12: **else if** $x \in \mathcal{L}$ **then**
13: $\mathcal{D}_a =$Extract$(\mathcal{L}.posOf(x), \mathcal{D}_{\mathcal{L}}, 0, \mathcal{D}_{\mathcal{L}}.length)$;
14: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}]'[index] = \mathbb{F}([\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}][index], \mathcal{D}_a, null, ¸[Var.posOf(x)])$;
15: **else**

```
            D_a = null
            if x ∈ O then
                D_a = Extract(O.posOf(x), D_O, 0, D_O.length);
            else if x ∈ L then
                D_a = Extract(L.posOf(x), D_L, 0, D_L.length);
            dN = [getDep(a)]
            m = dN.Length
            D_d = {D[1], ..., D[m]}
            for i = 1 → m do
                if dN[i] ∈ O then
                    D_d[i] = Extract(O.posOf(dN[i]), D_O, 0, D_O.length);
                else
                    D_d[i] = Extract(L.posOf(dN[i]), D_L, 0, D_L.length);
            [F_O1, ..., F_Lk]'[index] = IF([F_O1, ..., F_Lk][index], D_a, D_d, ¸[Var.posOf(x)])
        return [F_O1, ..., F_Lk]'
```

Algorithm 9 Expectation

```
1:  procedure E-step([F_O1, ..., F_Lk], D_O, O = (O1, ..., Oj), L = (L1, ..., Lk), I_ℓ(H),
    Var, NumBi)
2:      D'_L = [D^1_L, ..., D^k_L]
3:      for r = 0 → D'_L.Length do
4:          [F_L1, ..., F_Lk] = { }
5:          arraysize = 1
6:          List = [ ]
7:          for i = 1 → k do
8:              [F_L1, ..., F_Lk][i] = IF(getFactor(L[i], [F_O1, ..., F_Lk]))
9:              arraysize *= NumBi[Var.posOf(L[i])]
10:             List[i] = NumBi[Var.posOf(L[i])]
11:         A-MAP = [A_1, ..., A_arraysize]        ▷ Where each A_i is an array of size k
12:         NumRep = [ ]
13:         for i = 1 → k do
14:             prod = 1
15:             for j = i + 1 → k do
16:                 prod *= List[j]
17:             NumRep[i] = prod
18:         for j = 1 → k do
19:             t = NumRep[j]
20:             u = 0
21:             count = t
22:             ActualValue = 0
23:             while u < arraysize do
24:                 if count == 0 then
25:                     ActualValue ++
26:                     count = t
27:                     if ActualValue ⩾ List[j] then
28:                         ActualValue = 0
29:                 A_u[j] = ActualValue
```

30: count −−
31: u ++
32: $\mathcal{T} = [\mathcal{T}_1, \ldots, \mathcal{T}_k]$ ▷ Where each $\mathcal{T}_i$ is an array of size getMax(List)
33: **for** $i = 0 \rightarrow k$ **do**
34: $adder = (1/List[i])/2$
35: **for** $j = 0 \rightarrow getMax(List)$ **do**
36: $\mathcal{T}_i[j] = adder$
37: $adder \mathrel{+}= 1/List[i]$
38: $\xi = \{\xi[1], \ldots, \xi[j+k]\}$
39: **for** $i = 0 \rightarrow (j+k)$ **do**
40: **if** $Var[i] \in \mathcal{O}$ **then**
41: $\xi[i] = \mathcal{D}_O[r][\mathcal{O}[Var[i]]]$
42: $\mathcal{M}_\ell = -\infty$
43: $\mathcal{M}_\xi = [\mathcal{M}_\xi^1, \ldots, \mathcal{M}_\xi^{(j+k)}]$
44: $\mathcal{M}_\mathcal{C} = [\mathcal{M}_\mathcal{C}^1, \ldots, \mathcal{M}_\mathcal{C}^k]$
45: $\mathcal{C} = [\mathcal{C}_1, \ldots, \mathcal{C}_k]$
46: **for** $i = 1 \rightarrow arraysize$ **do**
47: **for** $j = 1 \rightarrow k$ **do**
48: $\xi[Var[\mathcal{L}[j]]] = \mathcal{T}_j[A\text{-}MAP[i][j]]$
49: $\mathcal{C}[j] = \mathcal{T}_j[A\text{-}MAP[i][j]]$
50: **if** $P(\xi|[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}], \mathcal{I}_\ell(\mathcal{H})) > \mathcal{M}_\ell$ **then**
51: $\mathcal{M}_\ell = P(\xi|[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_k}], \mathcal{I}_\ell(\mathcal{H}))$
52: $\mathcal{M}_\xi = \xi$
53: $\mathcal{M}_\mathcal{C} = \mathcal{C}$
54: $\mathcal{D}'_L[r] = \mathcal{M}_\mathcal{C}$
return $\mathcal{D}'_\mathcal{L}$

Line 2 of Algorithm 10 initialises a set of mutually independent factors for each variable. The full procedure of this initialisation of independent factors is given in Algorithm 7 in Appendix A. Line 3 of Algorithm 10 makes a copy of the independent factors. In this copy we later specify the conditional probability distribution for each factor according to the independence assumptions in $\mathcal{I}_\ell(\mathcal{G})$.

Line 4 to 28 states that for every independence assumption in $\mathcal{I}_\ell(\mathcal{H})$ do the following: (a) if there are no implied independence assertion then just copy the mutually independent factor (line 8); (b) otherwise, construct a new factor for the variable which specifies all conditional dependencies for the table CPD (lines 9 to 28). A complete detailed implementation of the **M-step**^{0} is given in Algorithm 10 in Appendix A.

Algorithm 10 Initial Maximization Algorithm

1: **procedure M-step**^{0}($\mathcal{D}_\mathcal{L}$, $\mathcal{D}_\mathcal{O}$, $\mathcal{O} = (O_1, \ldots, O_j)$, $\mathcal{L} = (L_1, \ldots, L_k)$, $Var = (O_1, \ldots, O_j, L_1, \ldots, L_k)$, $\mathcal{I}_\ell(\mathcal{H})$, ˛, NumBi)
2: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_j}]' =$ `GenIndep`$(Var, \mathcal{O}, \mathcal{L}, \mathcal{D}_\mathcal{O}, \mathcal{D}_\mathcal{L}, NumBi, ˛)$
3: $[\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_j}] = [\mathcal{F}_{O_1}, \ldots, \mathcal{F}_{L_j}]'$
4: **for** $\forall a \in \mathcal{I}_\ell(\mathcal{H})$ **do**
5: $x =$`getVar`(a)
6: index = Var.`posOf`(x)

```
        if a has no dependencies then
            [F_O1,...,F_Lj][index] = [F_O1,...,F_Lj]'[index]
        else
            dN = [getDep(a)]
            m = dN.Length
            dF = [F^a_1,...,F^a_m]
            for i = 1 → m do
                dIndex = Var.posOf(dN[i])
                dF[i] = [F_O1,...,F_Lj]'[dIndex]
            D_a = {}
            if x ∈ O then
                D_a =Extract(O[x], D_O, 0, D_O.length)
            else if x ∈ L then
                D_a =Extract(L.posOf(x), D_L, 0, D_L.length)
            D_d = {D_d[1],...,D_d[m]}
            for i = 1 → m do
                if O.contains(dN[i]) then
                    D_d[i] =Extract(O.posOf(dN[i]), D_O, 0, D_O.length)
                else
                    D_d[i] =Extract(L.posOf(dN[i]), D_L, 0, D_L.length)
            [F_O1,...,F_Lk][index] = ...
            IF(x, dN, D_a, D_d, [F_O1,...,F_Lk]'[index], dF, ,[Var.posOf(x)])
    return [F_O1,...,F_Lk]
```

Figure 58 an example of this structure learning scenario. The x-axis is the structure search iterations (i), and the y-axis is the dynamic score value. There are four parameters in this learning model similar to the d-AIC learning scenario in Figure 58: the value of the ground-truth total structure score (orange); the dynamic likelihood score of the recovered network at iteration i (blue); the dynamic penalty score of the recovered network at iteration i (red); and the total sum of the dynamic likelihood and penalty scores for the recovered network at iteration i (green).

We note that the total score of the recovered network monotonically increases before reaching a local optimum at iteration 9, it then performs a random restart and continues to monotonically increase. It appears that up to iteration 15 the structure score cannot improve, however, it finds a better optimum at iteration 20. The recovered network appears to approach the ground-truth network structure score.

A.1 EXAMPLE OF DATA

Here is an example of 3 instances of training data. I have separated each instance by a new line. The feature set is also given below.

Feature List: [Variable 0 LatentTIMESLICE0, Variable 0 Observable 0TIMESLICE0, Variable 0 Observable 1TIMESLICE0, Variable 1 LatentTIMESLICE0, Variable 1 Observable 0TIMESLICE0, Variable 1 Observable 1TIMESLICE0, Variable 2 LatentTIMESLICE0, Variable 2 Observable 0TIMESLICE0, Variable 2 Observable 1TIMESLICE0,

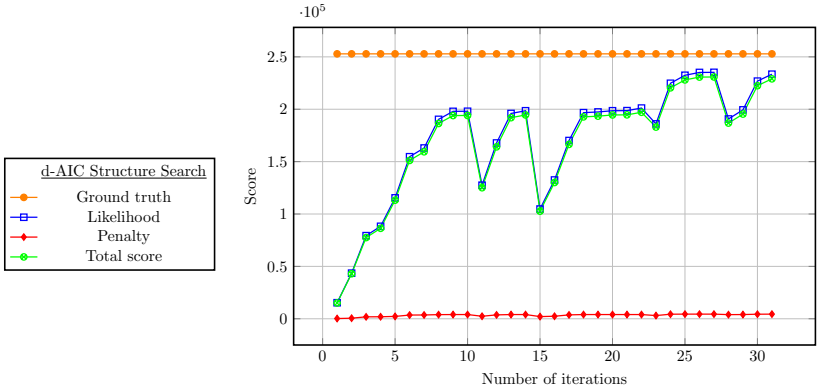


Figure 58: The value of structure scores for the GESS with AIC learning task. The y-axis is the score value and the x-axis represents the increase in structure search iterations

Variable 3 LatentTIMESLICE0, Variable 3 Observable 0TIMESLICE0, Variable 3 Observable 1TIMESLICE0, Variable 4 LatentTIMESLICE0, Variable 4 Observable 0TIMESLICE0, Variable 4 Observable 1TIMESLICE0, Variable 5 LatentTIMESLICE0, Variable 5 Observable 0TIMESLICE0, Variable 5 Observable 1TIMESLICE0, Variable 6 LatentTIMESLICE0, Variable 6 Observable 0TIMESLICE0, Variable 6 Observable 1TIMESLICE0, Variable 7 LatentTIMESLICE0, Variable 7 Observable 0TIMESLICE0, Variable 7 Observable 1TIMESLICE0, Variable 8 LatentTIMESLICE0, Variable 8 Observable 0TIMESLICE0, Variable 8 Observable 1TIMESLICE0, Variable 9 LatentTIMESLICE0, Variable 9 Observable 0TIMESLICE0, Variable 9 Observable 1TIMESLICE0, Variable 0 LatentTIMESLICE1, Variable 0 Observable 0TIMESLICE1, Variable 0 Observable 1TIMESLICE1, Variable 1 LatentTIMESLICE1, Variable 1 Observable 0TIMESLICE1, Variable 1 Observable 1TIMESLICE1, Variable 2 LatentTIMESLICE1, Variable 2 Observable 0TIMESLICE1, Variable 2 Observable 1TIMESLICE1, Variable 3 LatentTIMESLICE1, Variable 3 Observable 0TIMESLICE1, Variable 3 Observable 1TIMESLICE1, Variable 4 LatentTIMESLICE1, Variable 4 Observable 0TIMESLICE1, Variable 4 Observable 1TIMESLICE1, Variable 5 LatentTIMESLICE1, Variable 5 Observable 0TIMESLICE1, Variable 5 Observable 1TIMESLICE1, Variable 6 LatentTIMESLICE1, Variable 6 Observable 0TIMESLICE1, Variable 6 Observable 1TIMESLICE1, Variable 7 LatentTIMESLICE1, Variable 7 Observable 0TIMESLICE1, Variable 7 Observable 1TIMESLICE1, Variable 8 LatentTIMESLICE1, Variable 8 Observable 0TIMESLICE1, Variable 8 Observable 1TIMESLICE1, Variable 9 LatentTIMESLICE1, Variable 9 Observable 0TIMESLICE1, Variable 9 Observable 1TIMESLICE1, Variable 0 LatentTIMESLICE2, Variable 0 Observable 0TIMESLICE2, Variable 0 Observable 1TIMESLICE2, Variable 1 LatentTIMESLICE2, Variable 1 Observable 0TIMESLICE2, Variable 1 Observable 1TIMESLICE2, Variable 2 LatentTIMESLICE2, Variable 2 Observable 0TIMESLICE2, Variable 2 Observable 1TIMESLICE2, Variable 3 LatentTIMESLICE2, Variable 3 Observable 0TIMESLICE2, Variable 3 Observable 1TIMESLICE2, Variable 4 LatentTIMESLICE2, Variable 4 Observable 0TIMESLICE2, Variable 4 Observable 1TIMESLICE2, Variable 5 LatentTIMESLICE2, Variable 5 Observable 0TIMESLICE2, Variable 5 Observable 1TIMESLICE2, Variable 6 LatentTIMESLICE2, Variable 6 Observable 0TIMESLICE2, Variable 6 Observable 1TIMESLICE2, Variable 7 LatentTIMESLICE2, Variable 7 Observable 0TIMESLICE2, Variable 7 Observable 1TIMESLICE2, Variable 8 LatentTIMESLICE2, Variable 8 Observable 0TIMESLICE2, Variable 8 Observable 1TIMESLICE2, Variable 9 LatentTIMESLICE2, Variable 9 Observable 0TIMESLICE2, Variable 9 Observable 1TIMESLICE2]

```
Training Set:
0.0625 0.6875 0.4375 0.4375 0.5625 0.1875 0.8125 0.9375 0.5625 0.3125 0.8125
0.5625 0.1875 0.6875 0.9375 0.4375 0.6875 0.4375 0.3125 0.3125 0.4375 0.3125
0.6875 0.4375 0.5625 0.3125 0.9375 0.9375 0.4375 0.0625 0.8125 0.5625 0.0625
0.5625 0.3125 0.6875 0.6875 0.0625 0.4375 0.0625 0.0625 0.0625 0.8125 0.6875
0.4375 0.4375 0.4375 0.8125 0.0625 0.5625 0.9375 0.3125 0.1875 0.8125 0.4375
0.5625 0.6875 0.6875 0.9375 0.0625 0.3125 0.4375 0.3125 0.1875 0.0625 0.6875
0.0625 0.1875 0.9375 0.6875 0.4375 0.4375 0.6875 0.0625 0.0625 0.1875 0.8125
0.9375 0.5625 0.4375 0.9375 0.1875 0.3125 0.4375 0.3125 0.5625 0.8125 0.9375
0.9375 0.4375

0.8125 0.3125 0.5625 0.3125 0.8125 0.9375 0.1875 0.0625 0.3125 0.4375 0.3125
0.5625 0.5625 0.5625 0.5625 0.4375 0.6875 0.5625 0.8125 0.5625 0.4375 0.3125
0.5625 0.3125 0.9375 0.0625 0.1875 0.3125 0.4375 0.0625 0.6875 0.1875 0.3125
0.5625 0.0625 0.9375 0.8125 0.3125 0.9375 0.0625 0.6875 0.8125 0.9375 0.9375
0.9375 0.0625 0.5625 0.0625 0.6875 0.0625 0.0625 0.1875 0.5625 0.6875 0.5625
0.5625 0.8125 0.8125 0.1875 0.3125 0.4375 0.9375 0.1875 0.3125 0.3125 0.0625
0.4375 0.0625 0.1875 0.0625 0.3125 0.8125 0.4375 0.6875 0.0625 0.0625 0.3125
0.4375 0.8125 0.4375 0.4375 0.4375 0.9375 0.3125 0.9375 0.0625 0.1875 0.9375
0.6875 0.5625

0.9375 0.6875 0.8125 0.3125 0.3125 0.9375 0.1875 0.5625 0.6875 0.5625 0.4375
0.0625 0.6875 0.8125 0.5625 0.0625 0.5625 0.1875 0.0625 0.9375 0.0625 0.8125
0.0625 0.3125 0.8125 0.4375 0.3125 0.1875 0.6875 0.8125 0.5625 0.4375 0.5625
0.4375 0.5625 0.6875 0.3125 0.8125 0.4375 0.5625 0.0625 0.5625 0.6875 0.9375
0.8125 0.6875 0.5625 0.8125 0.1875 0.0625 0.8125 0.1875 0.9375 0.4375 0.8125
0.9375 0.1875 0.0625 0.8125 0.6875 0.1875 0.9375 0.1875 0.4375 0.1875 0.1875
0.6875 0.9375 0.5625 0.9375 0.9375 0.0625 0.8125 0.3125 0.4375 0.8125
```

BIBLIOGRAPHY

[Abbeel *et al.* 2006] Pieter Abbeel, Daphne Koller, and Andrew Y Ng. Learning factor graphs in polynomial time and sample complexity. *Journal of Machine Learning Research*, 7(Aug):1743–1788, 2006.

[Abramson 1994] Bruce Abramson. The design of belief network-based systems for price forecasting. *Computers & Electrical Engineering*, 20(2):163–180, 1994.

[Ajoodha and Rosman 2017] Ritesh Ajoodha and Benjamin Rosman. Tracking influence between naıve bayes models using score-based structure learning. In *IEEE proceedings, Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), 2017.*, Nov 2017.

[Ajoodha and Rosman 2018] Ritesh Ajoodha and Benjamin Rosman. Advancing intelligent systems by learning the influence structure between partially observed stochastic processes using iot sensor data. In *AAAI SmartIoT: AI Enhanced IoT Data Processing for Intelligent Applications, New Orleans Riverside, New Orleans, Lousiana, USA.*, 2018.

[Ajoodha *et al.* 2014] Ritesh Ajoodha, Richard Klein, and Marija Jakovljevic. Using statistical models and evolutionary algorithms in algorithmic music composition. In Khosrow-Pour Mehdi, editor, *The Encyclopedia of Information Science and Technology*. IGI Global, Hershey, Pennsylvania, United States, 3rd edition, 2014.

[Ajoodha *et al.* 2015] Ritesh Ajoodha, Richard Klein, and Benjamin Rosman. Single-labelled music genre classification using content-based features. In *IEEE proceedings, Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference (PRASA-RobMech), 2015*, pages 66–71, Nov 2015.

[Ajoodha 2014] Ritesh Ajoodha. *Automatic music genre classification*. Master's thesis, 2014.

[Akaike 1974] H Akaike. A new look at the statistical identification model. *IEEE Trans. Auto. Control*, 19:716–723, 1974.

[Andreassen *et al.* 1987] Steen Andreassen, Marianne Woldbye, Björn Falck, and Stig K Andersen. Munin: A causal probabilistic network for interpretation of electromyographic findings. In *Proceedings of the 10th international joint conference on Artificial intelligence-Volume 1*, pages 366–372. Morgan Kaufmann Publishers Inc., 1987.

[Bailey 1990] Norman TJ Bailey. *The elements of stochastic processes with applications to the natural sciences*, volume 25. John Wiley & Sons, 1990.

[Barron *et al.* 1998] Andrew Barron, Jorma Rissanen, and Bin Yu. The minimum description length principle in coding and modeling. *IEEE Transactions on Information Theory*, 44(6):2743–2760, 1998.

[Bates 1996] David S Bates. Jumps and stochastic volatility: Exchange rate processes implicit in deutsche mark options. *The Review of Financial Studies*, 9(1):69–107, 1996.

[Bauer *et al.* 1997] Eric Bauer, Daphne Koller, and Yoram Singer. Update rules for parameter estimation in bayesian networks. In *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*, pages 3–13. Morgan Kaufmann Publishers Inc., 1997.

[Beinlich *et al.* 1989] Ingo A Beinlich, Henri J Suermondt, R Martin Chavez, and Gregory F Cooper. *The ALARM monitoring system: A case study with two probabilistic inference techniques for belief networks*. Springer, 1989.

[Beretta *et al.* 2017] Stefano Beretta, Mauro Castelli, Ivo Goncalves, and Daniele Ramazzotti. A quantitative assessment of the effect of different algorithmic schemes to the task of learning the structure of bayesian networks. *arXiv preprint arXiv:1704.08676*, 2017.

[Bernardo and Smith 2001] José M Bernardo and Adrian FM Smith. *Bayesian theory*, 2001.

[Binder *et al.* 1997] John Binder, Daphne Koller, Stuart Russell, and Keiji Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29(2):213–244, 1997.

[Binford *et al.* 2013] Thomas O Binford, Tod S Levitt, and Wallace B Mann. Bayesian inference in model-based machine vision. *arXiv preprint arXiv:1304.2720*, 2013.

[Bishop 2006] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[Bishop 2007] C Bishop. Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*, 2007.

[Blei and Lafferty 2006] David M Blei and John D Lafferty. Dynamic topic models. In *Proceedings of the 23rd international conference on Machine learning*, pages 113–120. ACM, 2006.

[Blei *et al.* 2003] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[Blunsom 2004] Phil Blunsom. Hidden markov models. *Lecture notes, August*, 15:18–19, 2004.

[Bouchaala *et al.* 2010] Lobna Bouchaala, Afif Masmoudi, Faiez Gargouri, and Ahmed Rebai. Improving algorithms for structure learning in bayesian networks using a new implicit score. *Expert Systems with Applications*, 37(7):5470–5475, 2010.

[Bouckaert 1993] Remco Bouckaert. Probabilistic network construction using the minimum description length principle. *Symbolic and quantitative approaches to reasoning and uncertainty*, pages 41–48, 1993.

[Breese *et al.* 1992] JOHNS Breese, Eric Horvitz, MARKA Peot, Rodney Gay, and GEORGEH Quentin. Automated decision-analytic diagnosis of thermal performance in gas turbines. In *Proceedings of the International Gas Turbine and Aeroengine Congress and Exposition, Cologne, Germany, American Society of Mechanical Engineers*, 1992.

[Bunge 2017] Mario Bunge. *Causality and modern science*. Routledge, 2017.

[Buntine 1991] Wray Buntine. Theory refinement on bayesian networks. In *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*, pages 52–60. Morgan Kaufmann Publishers Inc., 1991.

[Buntine 1994] Wray L. Buntine. Operations for learning with graphical models. *JAIR*, 2:159–225, 1994.

[Buntine 1996] Wray Buntine. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on knowledge and data engineering*, 8(2):195–210, 1996.

[Buxton 1997] Hilary Buxton. Advanced visual surveillance using bayesian networks. 1997.

[Caffo *et al.* 2005] Brian S Caffo, Wolfgang Jank, and Galin L Jones. Ascent-based monte carlo expectation–maximization. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):235–251, 2005.

[Campos and Ji 2011] Cassio P de Campos and Qiang Ji. Efficient structure learning of bayesian networks using constraints. *Journal of Machine Learning Research*, 12(Mar):663–689, 2011.

[Carvalho *et al.* 2011] Alexandra M Carvalho, Teemu Roos, Arlindo L Oliveira, and Petri Myllymäki. Discriminative learning of bayesian networks via factorized conditional log-likelihood. *Journal of machine learning research*, 12(Jul):2181–2210, 2011.

[Casella and Berger 2002] George Casella and Roger L Berger. *Statistical inference*, volume 2. Duxbury Pacific Grove, CA, 2002.

[Charniak 1991] Eugene Charniak. Bayesian networks without tears. *AI magazine*, 12(4):50, 1991.

[Cheeseman *et al.* 1988] Peter Cheeseman, Matthew Self, James Kelly, Will Taylor, Don Freeman, and John C Stutz. Bayesian classification. In *AAAI*, volume 88, pages 607–611, 1988.

[Cheeseman *et al.* 1993] Peter Cheeseman, James Kelly, Matthew Self, John Stutz, Will Taylor, and Don Freeman. Autoclass: A bayesian classification system. In *Readings in knowledge acquisition and learning*, pages 431–441. Morgan Kaufmann Publishers Inc., 1993.

[Chen and Gopalakrishnan 1998] Scott Chen and Ponani Gopalakrishnan. Speaker, environment and channel change detection and clustering via the bayesian information criterion. In *Proc. darpa broadcast news transcription and understanding workshop*, volume 8, pages 127–132. Virginia, USA, 1998.

[Cheng *et al.* 1997] Jie Cheng, David A Bell, and Weiru Liu. Learning belief networks from data: An information theory based approach. In *Proceedings of the sixth international conference on Information and knowledge management*, pages 325–331. ACM, 1997.

[Chesley 1978] GR Chesley. Subjective probability elicitation techniques: A performance comparison. *Journal of Accounting Research*, pages 225–241, 1978.

[Chickering *et al.* 1994] David M Chickering, Dan Geiger, and David Heckerman. *Learning Bayesian networks is NP-hard*. Technical report, Technical Report MSR-TR-94-17, Microsoft Research, 1994.

[Chickering *et al.* 1995] Do Chickering, Dan Geiger, and David Heckerman. Learning bayesian networks: Search methods and experimental results. In *proceedings of fifth conference on artificial intelligence and statistics*, pages 112–128, 1995.

[Chickering *et al.* 2004] David Maxwell Chickering, David Heckerman, and Christopher Meek. Large-sample learning of bayesian networks is np-hard. *Journal of Machine Learning Research*, 5(Oct):1287–1330, 2004.

[Chickering 1995] David Maxwell Chickering. A transformational characterization of equivalent bayesian network structures. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 87–98. Morgan Kaufmann Publishers Inc., 1995.

[Chickering 1996] David Maxwell Chickering. Learning bayesian networks is np-complete. *Learning from data: Artificial intelligence and statistics V*, 112:121–130, 1996.

[Chickering 2002] David Maxwell Chickering. Learning equivalence classes of bayesian-network structures. *Journal of machine learning research*, 2(Feb):445–498, 2002.

[Chow and Liu 1968] C Chow and Cong Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

[Cohen 1992] Jacob Cohen. Statistical power analysis. *Current directions in psychological science*, 1(3):98–101, 1992.

[Colombo and Maathuis 2014] Diego Colombo and Marloes H Maathuis. Order-independent constraint-based causal structure learning. *The Journal of Machine Learning Research*, 15(1):3741–3782, 2014.

[Commenges and Gégout-Petit 2009] Daniel Commenges and Anne Gégout-Petit. A general dynamical statistical model with causal interpretation. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 71(3):719–736, 2009.

[Conati *et al.* 2002] Cristina Conati, Abigail Gertner, and Kurt Vanlehn. Using bayesian networks to manage uncertainty in student modeling. *User modeling and user-adapted interaction*, 12(4):371–417, 2002.

[Conati 2002] Cristina Conati. Probabilistic assessment of user's emotions in educational games. *Applied artificial intelligence*, 16(7-8):555–575, 2002.

[Cong *et al.* 2008] Rong-Gang Cong, Yi-Ming Wei, Jian-Lin Jiao, and Ying Fan. Relationships between oil price shocks and stock market: An empirical analysis from china. *Energy Policy*, 36(9):3544–3553, 2008.

[Cooper and Herskovits 1992] Gregory F Cooper and Edward Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine learning*, 9(4):309–347, 1992.

[Cooper 1990] Gregory F Cooper. The computational complexity of probabilistic inference using bayesian belief networks. *Artificial intelligence*, 42(2-3):393–405, 1990.

[Daneshkhah 2004] AR Daneshkhah. Psychological aspects influencing elicitation of subjective probability. *BEEPâs report*, 2004.

[Dasgupta 1999] Sanjoy Dasgupta. Learning polytrees. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 134–141. Morgan Kaufmann Publishers Inc., 1999.

[Dawid 1984] A Philip Dawid. Present position and potential developments: Some personal views: Statistical theory: The prequential approach. *Journal of the Royal Statistical Society. Series A (General)*, pages 278–292, 1984.

[De Dombal *et al.* 1972] FT De Dombal, DJ Leaper, John R Staniland, AP McCann, and Jane C Horrocks. Computer-aided diagnosis of acute abdominal pain. *Br Med J*, 2(5804):9–13, 1972.

[Dean and Kanazawa 1989] Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational intelligence*, 5(2):142–150, 1989.

[DeGroot and Schervish 2012] Morris H DeGroot and Mark J Schervish. *Probability and statistics*. Addison-Wesley,, 2012.

[D'Elia *et al.* 2003] Ciro D'Elia, Giovanni Poggi, and Giuseppe Scarpa. A tree-structured markov random field model for bayesian image segmentation. *IEEE Transactions on Image Processing*, 12(10):1259–1273, 2003.

[Dempster *et al.* 1977] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[Deng and Moore 1995] Kan Deng and Andrew W Moore. Multiresolution instance-based learning. In *IJCAI*, volume 95, pages 1233–1239, 1995.

[Dojer *et al.* 2006] Norbert Dojer, Anna Gambin, Andrzej Mizera, Bartek Wilczyński, and Jerzy Tiuryn. Applying dynamic bayesian networks to perturbed gene expression data. *BMC bioinformatics*, 7(1):249, 2006.

[Doob 1953] Joseph L Doob. *Stochastic processes*, volume 7. Wiley New York, 1953.

[Drton and Maathuis 2017] Mathias Drton and Marloes H Maathuis. Structure learning in graphical modeling. *Annual Review of Statistics and Its Application*, 4:365–393, 2017.

[Druzdzel 1993] Marek J Druzdzel. *Probabilistic reasoning in decision support systems: from computation to common sense*. PhD thesis, Carnegie Mellon University, 1993.

[Duda and Hart 1973] Richard O Duda and Peter E Hart. *Pattern recognition and scene analysis*, 1973.

[Duda *et al.* 1979] Richard Duda, John Gaschnig, and Peter Hart. Model design in the prospector consultant system for mineral exploration. *Expert systems in the microelectronic age*, 1234:153–167, 1979.

[Durbin *et al.* 1998] Richard Durbin, Sean R Eddy, Anders Krogh, and Graeme Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge university press, 1998.

[Eddy 1996] Sean R Eddy. Hidden markov models. *Current opinion in structural biology*, 6(3):361–365, 1996.

[Elidan *et al.* 2002] Gal Elidan, Matan Ninio, Nir Friedman, and Dale Shuurmans. Data perturbation for escaping local maxima in learning. In *AAAI/IAAI*, pages 132–139, 2002.

[Ellis and Wong 2008] Byron Ellis and Wing Hung Wong. Learning causal bayesian network structures from experimental data. *Journal of the American Statistical Association*, 103(482):778–789, 2008.

[Enders and Bandalos 2001] Craig K Enders and Deborah L Bandalos. The relative performance of full information maximum likelihood estimation for missing data in structural equation models. *Structural equation modeling*, 8(3):430–457, 2001.

[Fan *et al.* 2014] Xiannian Fan, Changhe Yuan, and Brandon M Malone. Tightening bounds for bayesian network structure learning. In *AAAI*, pages 2439–2445, 2014.

[Fayyad *et al.* 1996] Usama M Fayyad, Gregory Piatetsky-Shapiro, Padhraic Smyth, and Ramasamy Uthurusamy. *Advances in knowledge discovery and data mining*, volume 21. AAAI press Menlo Park, 1996.

[Fraley and Raftery 2002] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.

[Friedman and others 1997] Nir Friedman et al. Learning belief networks in the presence of missing values and hidden variables. In *ICML*, volume 97, pages 125–133, 1997.

[Friedman and Yakhini 1996] Nir Friedman and Zohar Yakhini. On the sample complexity of learning bayesian networks. In *Proceedings of the Twelfth international conference on Uncertainty in artificial intelligence*, pages 274–282. Morgan Kaufmann Publishers Inc., 1996.

[Friedman *et al.* 1998] Nir Friedman, Kevin Murphy, and Stuart Russell. Learning the structure of dynamic probabilistic networks. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 139–147. Morgan Kaufmann Publishers Inc., 1998.

[Friedman *et al.* 1999] Nir Friedman, Iftach Nachman, and Dana Peér. Learning bayesian network structure from massive datasets: the sparse candidate algorithm. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 206–215. Morgan Kaufmann Publishers Inc., 1999.

[Friedman *et al.* 2000] Nir Friedman, Michal Linial, Iftach Nachman, and Dana Pe'er. Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620, 2000.

[Friedman 1998] Nir Friedman. The bayesian structural em algorithm. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 129–138. Morgan Kaufmann Publishers Inc., 1998.

[Gänssler and Stute 1979] Peter Gänssler and Winfried Stute. Empirical processes: a survey of results for independent and identically distributed random variables. *The Annals of Probability*, pages 193–243, 1979.

[Garcia 2004] Luis David Garcia. Algebraic statistics in model selection. In *Proceedings of the 20th conference on Uncertainty in artificial intelligence*, pages 177–184. AUAI Press, 2004.

[Gardiner 1985] Crispin Gardiner. Stochastic methods. *Springer Series in Synergetics (Springer-Verlag, Berlin, 2009)*, 1985.

[Geiger and Pearl 2013] Dan Geiger and Judea Pearl. On the logic of causal models. *arXiv preprint arXiv:1304.2355*, 2013.

[Geiger *et al.* 1990] Dan Geiger, Thomas Verma, and Judea Pearl. Identifying independence in bayesian networks. *Networks*, 20(5):507–534, 1990.

[Geiger *et al.* 2001] Dan Geiger, David Heckerman, Henry King, and Christopher Meek. Stratified exponential families: graphical models and model selection. *Annals of statistics*, pages 505–529, 2001.

[Geiger *et al.* 2013] Dan Geiger, Tom S Verma, and Judea Pearl. d-separation: From theorems to algorithms. *arXiv preprint arXiv:1304.1505*, 2013.

[Gelman *et al.* 2014] Andrew Gelman, John B Carlin, Hal S Stern, and Donald B Rubin. *Bayesian data analysis*, volume 2. Chapman & Hall/CRC Boca Raton, FL, USA, 2014.

[Getoor *et al.* 2002] Lisa Getoor, Nir Friedman, Daphne Koller, and Benjamin Taskar. Learning probabilistic models of link structure. *Journal of Machine Learning Research*, 3(Dec):679–707, 2002.

[Ghahramani and Jordan 1994] Zoubin Ghahramani and Michael I Jordan. Supervised learning from incomplete data via an em approach. *Advances in neural information processing systems*, pages 120–120, 1994.

[Gilks *et al.* 1994] Wally R Gilks, Andrew Thomas, and David J Spiegelhalter. A language and program for complex bayesian modelling. *The Statistician*, pages 169–177, 1994.

[Gilks *et al.* 1995] Walter R Gilks, Sylvia Richardson, and David Spiegelhalter. *Markov chain Monte Carlo in practice*. CRC press, 1995.

[Glover and Laguna 2013] Fred Glover and Manuel Laguna. *Tabu Searchâ*. Springer, 2013.

[Gorry and Barnett 1968] G Anthony Gorry and G Octo Barnett. Experience with a model of sequential diagnosis. *Computers and Biomedical Research*, 1(5):490–507, 1968.

[Greenberg 1998] Harvey J Greenberg. Greedy algorithms for minimum spanning tree. *University of Colorado at Denver*, 1998.

[Greiner *et al.* 2005] Russell Greiner, Xiaoyuan Su, Bin Shen, and Wei Zhou. Structural extension to logistic regression: Discriminative parameter learning of belief net classifiers. *Machine Learning*, 59(3):297–322, 2005.

[Grinthal and Berkeley Heights 2015] Ted Grinthal and NJ Berkeley Heights. Correlation vs. causation. *AMERICAN SCIENTIST*, 103(2):84–84, 2015.

[Gu *et al.* 1994] Yiqun Gu, D Ramanee Peiris, John W Crawford, JW NcNicol, B Marshall, and RA Jefferies. An application of belief networks to future crop production. In *Artificial Intelligence for Applications, 1994., Proceedings of the Tenth Conference on*, pages 305–309. IEEE, 1994.

[Gyftodimos and Flach 2002] Elias Gyftodimos and Peter A Flach. Hierarchical bayesian networks: A probabilistic reasoning model for structured domains. In *Proceedings of the ICML-2002 Workshop on Development of Representations*, pages 23–30. The university of New South Wales, 2002.

[Hartemink *et al.* 2002] Alexander J Hartemink, David K Gifford, Tommi S Jaakkola, and Richard A Young. Bayesian methods for elucidating genetic regulatory networks. *IEEE Intelligent Systems*, 17(2):37–43, 2002.

[Hastie *et al.* 2001] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. The elements of statistical learning. 2001. *NY Springer*, 2001.

[Hatfield *et al.* 2006] Julie Hatfield, Gavin J Faunce, and RS Job. Avoiding confusion surrounding the phrase âcorrelation does not imply causationâ. *Teaching of Psychology*, 33(1):49–51, 2006.

[Heckerman *et al.* 1995a] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20(3):197–243, 1995.

[Heckerman *et al.* 1995b] David Heckerman, Abe Mamdani, and Michael P Wellman. Real-world applications of bayesian networks. *Communications of the ACM*, 38(3):24–26, 1995.

[Heckerman *et al.* 2016] David Heckerman, Eric Horvitz, and Bharat N Nathwani. Toward normative expert systems part i. *Methods of information in medicine*, 31, 2016.

[Heckerman 1996] David Heckerman. 1 1 bayesian networks for knowledge discovery. 1996.

[Heckerman 1998] David et. al. Heckerman. A tutorial on learning with bayesian networks. *Nato Asi Series D Behavioural And Social Sciences*, 89:301–354, 1998.

[Hilbe 2011] Joseph M Hilbe. Logistic regression. In *International Encyclopedia of Statistical Science*, pages 755–758. Springer, 2011.

[Hoadley 1971] Bruce Hoadley. Asymptotic properties of maximum likelihood estimators for the independent not identically distributed case. *The Annals of mathematical statistics*, pages 1977–1991, 1971.

[Höffgen 1993] Klaus-U Höffgen. Learning and robust learning of product distributions. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 77–83. ACM, 1993.

[Horas 2014] Horas. Beta distribution, based on the work of krishnavedala (own work) [public domain], via wikimedia commons. 2014. Accessed: 2018-01-01.

[Howard and Matheson 1984] RA Howard and JE Matheson. Influence diagrams, the principles and applications of decision analysis (vol. ii). *Menlo Park, CA: Strategic Decisions Group*, 1984.

[Howard 1970] Ronald A Howard. Decision analysis: Perspectives on inference, decision, and experimentation. *Proceedings of the IEEE*, 58(5):632–643, 1970.

[Huang *et al.* 2006] Chung-Lin Huang, Huang-Chia Shih, and Chung-Yuan Chao. Semantic analysis of soccer video using dynamic bayesian network. *IEEE Transactions on Multimedia*, 8(4):749–760, 2006.

[Huang *et al.* 2009] Feixue Huang, Pengfei Gao, and Yu Wang. Comparison of prim and kruskal on shanghai and shenzhen 300 index hierarchical structure tree. In *Web Information Systems and Mining, 2009. WISM 2009. International Conference on*, pages 237–241. IEEE, 2009.

[Indyk 2004] Piotr Indyk. Nearest neighbors in high-dimensional spaces. 2004.

[Jayakrishnan *et al.* 1994] R Jayakrishnan, Hani S Mahmassani, and Ta-Yin Hu. An evaluation tool for advanced traffic information and management systems in urban networks. *Transportation Research Part C: Emerging Technologies*, 2(3):129–147, 1994.

[Jensen 1996] Finn V Jensen. *An introduction to Bayesian networks*, volume 210. UCL press London, 1996.

[Johansen and Juselius 1990] Søren Johansen and Katarina Juselius. Maximum likelihood estimation and inference on cointegrationâwith applications to the demand for money. *Oxford Bulletin of Economics and statistics*, 52(2):169–210, 1990.

[John and Langley 1995] George H John and Pat Langley. Estimating continuous distributions in bayesian classifiers. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 338–345. Morgan Kaufmann Publishers Inc., 1995.

[Jordan 1998] Michael Irwin Jordan. *Learning in graphical models*, volume 89. Springer Science & Business Media, 1998.

[Joyce 2011] James M Joyce. Kullback-leibler divergence. In *International Encyclopedia of Statistical Science*, pages 720–722. Springer, 2011.

[Judea Pearl 1991] TS Vermal J Judea Pearl. Equivalence and synthesis of causal models. In *Proceedings of Sixth Conference on Uncertainty in Artijicial Intelligence*, pages 220–227, 1991.

[Kafai and Bhanu 2012] Mehran Kafai and Bir Bhanu. Dynamic bayesian networks for vehicle classification in video. *IEEE Transactions on Industrial Informatics*, 8(1):100–109, 2012.

[Kalman 1960] Rudolph Emil et. al. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[Kanji 2006] Gopal K Kanji. *100 statistical tests*. Sage, 2006.

[Karlin 2014] Samuel Karlin. *A first course in stochastic processes*. Academic press, 2014.

[Kass and Raftery 1995] Robert E Kass and Adrian E Raftery. Bayes factors. *Journal of the american statistical association*, 90(430):773–795, 1995.

[Kearns *et al.* 1998] Michael Kearns, Yishay Mansour, and Andrew Y Ng. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Learning in graphical models*, pages 495–520. Springer, 1998.

[Kemeny and Snell 1960] John G Kemeny and James Laurie Snell. *Finite markov chains*, volume 356. van Nostrand Princeton, NJ, 1960.

[Kim 2000] Hongseok Kim. Stochastic model based audio watermark and whitening filter for improved detection. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 4, pages 1971–1974. IEEE, 2000.

[Klein *et al.* 2011] Richard Klein, Angelo Kyrilov, and Mayya Tokman. Automated assessment of short free-text responses in computer science using latent semantic analysis. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, pages 158–162. ACM, 2011.

[Klugh 2013] Henry E Klugh. *Statistics: The essentials for research*. Psychology Press, 2013.

[Koivisto and Sood 2004] Mikko Koivisto and Kismat Sood. Exact bayesian structure discovery in bayesian networks. *Journal of Machine Learning Research*, 5(May):549–573, 2004.

[Kok and Domingos 2005] Stanley Kok and Pedro Domingos. Learning the structure of markov logic networks. In *Proceedings of the 22nd international conference on Machine learning*, pages 441–448. ACM, 2005.

[Koller and Friedman 2009] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques. (Chapter 16; 17; 18; and 19)*. MIT press, 2009.

[Komarek and Moore 2000] Paul Komarek and Andrew W Moore. A dynamic adaptation of ad-trees for efficient machine learning on large data sets. In *ICML*, pages 495–502, 2000.

[Koski and Noble 2011] Timo Koski and John Noble. *Bayesian networks: an introduction*, volume 924. John Wiley & Sons, 2011.

[Krishnapuram *et al.* 2005] Balaji Krishnapuram, Lawrence Carin, Mario AT Figueiredo, and Alexander J Hartemink. Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE transactions on pattern analysis and machine intelligence*, 27(6):957–968, 2005.

[Kschischang *et al.* 2001] Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[Kullback 1997] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[Lam and Bacchus 1993] Wai Lam and Fahiem Bacchus. Using causal information and local measures to learn bayesian networks. In *Proceedings of the Ninth international conference on Uncertainty in artificial intelligence*, pages 243–250. Morgan Kaufmann Publishers Inc., 1993.

[Lauritzen and Spiegelhalter 1988] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 157–224, 1988.

[Lauritzen 1995] Steffen L Lauritzen. The em algorithm for graphical association models with missing data. *Computational Statistics & Data Analysis*, 19(2):191–201, 1995.

[Lee *et al.* 2007] Su-In Lee, Varun Ganapathi, and Daphne Koller. Efficient structure learning of markov networks using l_1-regularization. In *Advances in neural Information processing systems*, pages 817–824, 2007.

[Lehmann and Romano 2006] Erich L Lehmann and Joseph P Romano. *Testing statistical hypotheses*. Springer Science & Business Media, 2006.

[Lehn 2017] Christian Lehn. Maximum likelihood estimation. 2017.

[Levitt *et al.* 2013] Tod S Levitt, John Mark Agosta, and Thomas O Binford. Model-based influence diagrams for machine vision. *arXiv preprint arXiv:1304.1517*, 2013.

[Lewis 1998] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer, 1998.

[Little and Rubin 2014] Roderick JA Little and Donald B Rubin. *Statistical analysis with missing data*. John Wiley & Sons, 2014.

[Little 1976] Roderick JA Little. Inference about means from incomplete multivariate data. *Biometrika*, 63(3):593–604, 1976.

[Liu *et al.* 1996] Chi-Shi Liu, Hsiao-Chuan Wang, and C Lee. Speaker verification using normalized log-likelihood score. *IEEE Transactions on speech and audio processing*, 4(1):56, 1996.

[Madsen *et al.* 2017] Anders L Madsen, Frank Jensen, Antonio Salmerón, Helge Langseth, and Thomas D Nielsen. A parallel algorithm for bayesian network structure learning from large data sets. *Knowledge-Based Systems*, 117:46–55, 2017.

[Mcauliffe and Blei 2008] Jon D Mcauliffe and David M Blei. Supervised topic models. In *Advances in neural information processing systems*, pages 121–128, 2008.

[McCallum and Nigam 1998] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Madison, WI, 1998.

[McLachlan and Krishnan 2007] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[Meek 1995] Christopher Meek. Causal inference and causal explanation with background knowledge. In *Proceedings of the Eleventh conference on Uncertainty in artificial intelligence*, pages 403–410. Morgan Kaufmann Publishers Inc., 1995.

[Meek 2001] Christopher Meek. Finding a path is harder than finding a tree. *J. Artif. Intell. Res. (JAIR)*, 15:383–389, 2001.

[Minka and Lafferty 2002] Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

[Minka 2001] Thomas P Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*, pages 362–369. Morgan Kaufmann Publishers Inc., 2001.

[Minka 2005] Tom Minka. *Divergence measures and message passing*. Technical report, Technical report, Microsoft Research, 2005.

[Mohammadi and Wit 2015] Abdolreza Mohammadi and Ernst C Wit. Bayesian structure learning in sparse gaussian graphical models. *Bayesian Analysis*, 10(1):109–138, 2015.

[Moore and Essa 2002] Darnell Moore and Irfan Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *AAAI/IAAI*, pages 770–776, 2002.

[Moore and Lee 1998] AW Moore and Mary Soon Lee. Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8(3):67–91, 1998.

[Moore and Wong 2003] Andrew Moore and Weng-Keen Wong. Optimal reinsertion: A new search operator for accelerated and more accurate bayesian network structure learning. In *ICML*, volume 3, pages 552–559, 2003.

[Moore 2000] Andrew W Moore. The anchors hierarchy: Using the triangle inequality to survive high dimensional data. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 397–405. Morgan Kaufmann Publishers Inc., 2000.

[Murphy and Russell 2002] Kevin Patrick Murphy and Stuart Russell. Dynamic bayesian networks: representation, inference and learning. 2002.

[Murphy 1998] Kevin Murphy. A brief introduction to graphical models and bayesian networks. 1998.

[Murphy 2012] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[Nadi *et al.* 1991] Fariborz Nadi, Alice M Agogino, and David A Hodges. Use of influence diagrams and neural networks in modeling semiconductor manufacturing processes. *IEEE Transactions on Semiconductor Manufacturing*, 4(1):52–58, 1991.

[Neal and Hinton 1998] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*, pages 355–368. Springer, 1998.

[Nielsen *et al.* 2002] Jens D Nielsen, Tomáš Kočka, and Jose M Peña. On local optima in learning bayesian networks. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 435–442. Morgan Kaufmann Publishers Inc., 2002.

[Oliver and Horvitz 2005] Nuria Oliver and Eric Horvitz. A comparison of hmms and dynamic bayesian networks for recognizing office activities. *User Modeling 2005*, pages 149–149, 2005.

[Opgen-Rhein and Strimmer 2007] Rainer Opgen-Rhein and Korbinian Strimmer. From correlation to causation networks: a simple approximate learning algorithm and its application to high-dimensional plant gene expression data. *BMC systems biology*, 1(1):37, 2007.

[Ortiz and Kaelbling 1999] Luis E Ortiz and Leslie Pack Kaelbling. Accelerating em: An empirical study. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 512–521. Morgan Kaufmann Publishers Inc., 1999.

[Papageorgiou 1990] Markos Papageorgiou. Dynamic modeling, assignment, and route guidance in traffic networks. *Transportation Research Part B: Methodological*, 24(6):471–495, 1990.

[Pavlovic *et al.* 1999] Vladimir Pavlovic, James M Rehg, Tat-Jen Cham, and Kevin P Murphy. A dynamic bayesian network approach to figure tracking using learned dynamic models. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 1, pages 94–101. IEEE, 1999.

[Pearl and Verma 1995] Judea Pearl and Thomas S Verma. A theory of inferred causation. *Studies in Logic and the Foundations of Mathematics*, 134:789–811, 1995.

[Pearl 1988] Judea Pearl. Probabilistic reasoning in intelligent systems. palo alto. *Morgan Kaufmann. PEAT, J., VAN DEN BERG, R., & GREEN, W.(1994). Changing prevalence of asthma in australian children. British Medical Journal*, 308:1591–1596, 1988.

[Pearl 2011] Judea Pearl. Bayesian networks. *Department of Statistics, UCLA*, 2011.

[Pearl 2014] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 2014.

[Peelen *et al.* 2010] Linda Peelen, Nicolette F de Keizer, Evert de Jonge, Robert-Jan Bosman, Ameen Abu-Hanna, and Niels Peek. Using hierarchical dynamic bayesian networks to investigate dynamics of organ failure in patients in the intensive care unit. *Journal of biomedical informatics*, 43(2):273–286, 2010.

[Pepe and Mori 1993] Margaret Sullivan Pepe and Motomi Mori. Kaplanâmeier, marginal or conditional probability curves in summarizing competing risks failure time data? *Statistics in medicine*, 12(8):737–751, 1993.

[Perriere and Thioulouse 2003] Guy Perriere and Jean Thioulouse. Use of correspondence discriminant analysis to predict the subcellular location of bacterial proteins. *Computer methods and programs in biomedicine*, 70(2):99–105, 2003.

[Pólya 1937] George Pólya. Kombinatorische anzahlbestimmungen für gruppen, graphen und chemische verbindungen. *Acta mathematica*, 68(1):145–254, 1937.

[Press 1989] S James Press. *Bayesian statistics: principles, models, and applications*, volume 210. John Wiley & Sons Inc, 1989.

[Rabiner and Juang 1986] Lawrence Rabiner and B Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986.

[Rabiner 1989] Lawrence R Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[Renooij and van der Gaag 2002] Silja Renooij and Linda C van der Gaag. From qualitative to quantitative probabilistic networks. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 422–429. Morgan Kaufmann Publishers Inc., 2002.

[Rish 2001] Irina Rish. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence*, volume 3, pages 41–46. IBM, 2001.

[Rissanen 1987] Jorma Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 223–239, 1987.

[Rubin 1976] Donald B Rubin. Inference and missing data. *Biometrika*, pages 581–592, 1976.

[Rusakov and Geiger 2005] Dmitry Rusakov and Dan Geiger. Asymptotic model selection for naive bayesian networks. *Journal of Machine Learning Research*, 6(Jan):1–35, 2005.

[Sachs *et al.* 2005] Karen Sachs, Omar Perez, Dana Pe'er, Douglas A Lauffenburger, and Garry P Nolan. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721):523–529, 2005.

[Sakamoto and Ghanem 2002] Shigehiro Sakamoto and Roger Ghanem. Polynomial chaos decomposition for the simulation of non-gaussian nonstationary stochastic processes. *Journal of engineering mechanics*, 128(2):190–201, 2002.

[Salmon 1984] Wesley Salmon. Scientific explanation and the causal structure of the world. 1984.

[Schachter and Heckerman 1987] Ross D Schachter and David Heckerman. Thinking backward for knowledge acquisition. *AI magazine*, 8(3):55, 1987.

[Schervish 2012] Mark J Schervish. *Theory of statistics*. Springer Science & Business Media, 2012.

[Scholz 1985] FW Scholz. Maximum likelihood estimation. *Encyclopedia of statistical sciences*, 1985.

[Schum 1994] David A Schum. *The evidential foundations of probabilistic reasoning*. Northwestern University Press, 1994.

[Schwarz and others 1978] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[Schweppe 1973] Fred C Schweppe. *Uncertain dynamic systems*. Prentice Hall, 1973.

[Settimi and Smith 1998] Raffaella Settimi and Jim Q Smith. On the geometry of bayesian graphical models with hidden variables. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pages 472–479. Morgan Kaufmann Publishers Inc., 1998.

[Shafer 1976] Glenn et.al. Shafer. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, 1976.

[Shinde and Prasad 2017] Tejaswinee A Shinde and Jayashree R Prasad. Iot based animal health monitoring with naive bayes classification. *IJETT*, 1(2), 2017.

[Silander and Myllymaki 2012] Tomi Silander and Petri Myllymaki. A simple approach for finding the globally optimal bayesian network structure. *arXiv preprint arXiv:1206.6875*, 2012.

[Singh and Moore 2005] Ajit P Singh and Andrew W Moore. Finding optimal bayesian networks by dynamic programming. 2005.

[Smith 1989] JQ Smith. Influence diagrams for statistical modelling. *The Annals of Statistics*, pages 654–672, 1989.

[Smyth *et al.* 1997] Padhraic Smyth, David Heckerman, and Michael I Jordan. Probabilistic independence networks for hidden markov probability models. *Neural computation*, 9(2):227–269, 1997.

[Spetzler and Stael von Holstein 1975] Carl S Spetzler and Carl-Axel S Stael von Holstein. Exceptional paperâprobability encoding in decision analysis. *Management science*, 22(3):340–358, 1975.

[Spiegelhalter *et al.* 1993] David J Spiegelhalter, A Philip Dawid, Steffen L Lauritzen, and Robert G Cowell. Bayesian analysis in expert systems. *Statistical science*, pages 219–247, 1993.

[Spirtes *et al.* 2000] Peter Spirtes, Clark N Glymour, and Richard Scheines. *Causation, prediction, and search*. MIT press, 2000.

[Sun *et al.* 2006] Shiliang Sun, Changshui Zhang, and Guoqiang Yu. A bayesian network approach to traffic flow forecasting. *IEEE Transactions on intelligent transportation systems*, 7(1):124–132, 2006.

[Suzuki 1993] Joe Suzuki. A construction of bayesian networks from databases based on an mdl principle. In *Proceedings of the Ninth international conference on Uncertainty in artificial intelligence*, pages 266–273. Morgan Kaufmann Publishers Inc., 1993.

[Suzuki 2017] Joe Suzuki. An efficient bayesian network structure learning strategy. *New Generation Computing*, 35(1):105–124, 2017.

[Tamura *et al.* 1991] Y Tamura, T Sato, M Ooe, and M Ishiguro. A procedure for tidal analysis with a bayesian information criterion. *Geophysical Journal International*, 104(3):507–516, 1991.

[Tanner and Wong 1987] Martin A Tanner and Wing Hung Wong. The calculation of posterior distributions by data augmentation. *Journal of the American statistical Association*, 82(398):528–540, 1987.

[Tanner 1991] Martin A Tanner. *Tools for statistical inference*, volume 3. Springer, 1991.

[Tenenbaum *et al.* 2011] Joshua B Tenenbaum, Charles Kemp, Thomas L Griffiths, and Noah D Goodman. How to grow a mind: Statistics, structure, and abstraction. *science*, 331(6022):1279–1285, 2011.

[Teyssier and Koller 2012] Marc Teyssier and Daphne Koller. Ordering-based search: A simple and effective algorithm for learning bayesian networks. *arXiv preprint arXiv:1207.1429*, 2012.

[Thiesson 1995] Bo Thiesson. *Accelerated quantification of Bayesian networks with incomplete data*. University of Aalborg, Institute for Electronic Systems, Department of Mathematics and Computer Science, 1995.

[Tompkins and Lawley 2009] Penny Tompkins and James Lawley. Opinions on global optimum. 2009. http://www.writeopinions.com/global-optimum. Accessed: 2010-09-30.

[Tompkins and Lawley 2013] Penny Tompkins and James Lawley. Applying cross-domain thinking. 2013. http://www.cleanlanguage.co.uk/articles/articles/337/1/Applying-Cross-Domain-Thinking/Page1.html. Accessed: 2010-09-30.

[Tsamardinos *et al.* 2006] Ioannis Tsamardinos, Laura E Brown, and Constantin F Aliferis. The max-min hill-climbing bayesian network structure learning algorithm. *Machine learning*, 65(1):31–78, 2006.

[Tversky and Kahneman 1975] Amos Tversky and Daniel Kahneman. Judgment under uncertainty: Heuristics and biases. In *Utility, probability, and human decision making*, pages 141–162. Springer, 1975.

[Van Kampen 1992] Nicolaas Godfried Van Kampen. *Stochastic processes in physics and chemistry*, volume 1. Elsevier, 1992.

[Varadarajan *et al.* 2010] Jagan Varadarajan, Rémi Emonet, and Jean-Marc Odobez. A sparsity constraint for topic models-application to temporal activity mining. 2010.

[Verma and Pearl 1992] Thomas Verma and Judea Pearl. An algorithm for deciding if a set of observed independencies has a causal explanation. In *Proceedings of the Eighth international conference on uncertainty in artificial intelligence*, pages 323–330. Morgan Kaufmann Publishers Inc., 1992.

[Verma and Pearl 2013] Tom S Verma and Judea Pearl. Causal networks: Semantics and expressiveness. *arXiv preprint arXiv:1304.2379*, 2013.

[Wainwright *et al.* 2008] Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

[Wang *et al.* 2008] Peng Wang, Peter B Luh, Shi-Chung Chang, and Jin Sun. Modeling and optimization of crowd guidance for building emergency evacuation. In *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pages 328–334. IEEE, 2008.

[Warner *et al.* 1961] Homer R Warner, Alan F Toronto, L George Veasey, and Robert Stephenson. A mathematical approach to medical diagnosis: application to congenital heart disease. *Jama*, 177(3):177–183, 1961.

[Weber and Jouffe 2003] Philippe Weber and Lionel Jouffe. Reliability modelling with dynamic bayesian networks. *IFAC Proceedings Volumes*, 36(5):57–62, 2003.

[Weber *et al.* 2012] Philippe Weber, Gabriela Medina-Oliva, Christophe Simon, and Benoît Iung. Overview on bayesian networks applications for dependability, risk analysis and maintenance areas. *Engineering Applications of Artificial Intelligence*, 25(4):671–682, 2012.

[Wellman 1990] Michael P Wellman. Fundamental concepts of qualitative probabilistic networks. *Artificial intelligence*, 44(3):257–303, 1990.

[West 1996] Mike West. *Bayesian forecasting*. Wiley Online Library, 1996.

[Wolbrecht *et al.* 2000] Eric Wolbrecht, Bruce D'ambrosio, Robert Paasch, and Doug Kirby. Monitoring and diagnosis of a multistage manufacturing process using bayesian networks. *Ai Edam*, 14(1):53–67, 2000.

[Wright 1921] Sewall Wright. Correlation and causation. *Journal of agricultural research*, 20(7):557–585, 1921.

[Wright 1934] Sewall Wright. The method of path coefficients. *The annals of mathematical statistics*, 5(3):161–215, 1934.

[Zhao *et al.* 2011] Wayne Xin Zhao, Jing Jiang, Jianshu Weng, Jing He, Ee-Peng Lim, Hongfei Yan, and Xiaoming Li. Comparing twitter and traditional media using topic models. In *European Conference on Information Retrieval*, pages 338–349. Springer, 2011.

[Zou and Feng 2009] Cunlu Zou and Jianfeng Feng. Granger causality vs. dynamic bayesian network inference: a comparative study. *BMC bioinformatics*, 10(1):122, 2009.

[Zweig and Russell 1998] Geoffrey Zweig and Stuart Russell. Speech recognition with dynamic bayesian networks. 1998.

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The original code was reworked by the author of this thesis. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

http://code.google.com/p/classicthesis/

Happy users of `classicthesis` usually send a real postcard to André Miede, a collection of postcards received so far is featured here:

http://postcards.miede.de/