New Techniques for Learning Parameters in Bayesian Networks

Yun Zhou

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

of

Queen Mary University of London

Department of Computer Science

Queen Mary
University of London

August 8, 2015

I, Yun Zhou, confirm that the work presented in this thesis is my own. Where information has been derived from other sources, I confirm that this has been indicated in the work.

Abstract

One of the hardest challenges in building a realistic Bayesian network (BN) model is to construct the node probability tables (NPTs). Even with a fixed predefined model structure and very large amounts of relevant data, machine learning methods do not consistently achieve great accuracy compared to the ground truth when learning the NPT entries (parameters). Hence, it is widely believed that incorporating expert judgment or related domain knowledge can improve the parameter learning accuracy. This is especially true in the sparse data situation. Expert judgments come in many forms. In this thesis we focus on expert judgment that specifies inequality or equality relationships among variables. Related domain knowledge is data that comes from a different but related problem.

By exploiting expert judgment and related knowledge, this thesis makes novel contributions to improve the BN parameter learning performance, including:

- The multinomial parameter learning model with interior constraints (MPL-C) and exterior constraints (MPL-EC). This model itself is an auxiliary BN, which encodes the multinomial parameter learning process and constraints elicited from the expert judgments.

- The BN parameter transfer learning (BNPTL) algorithm. Given some potentially related (source) BNs, this algorithm automatically explores the most relevant source BN and BN fragments, and fuses the selected source and target parameters in a robust way.

- A generic BN parameter learning framework. This framework uses both expert judgments and transferred knowledge to improve the learning accuracy. This framework transfers the mined data statistics from the source network as the parameter priors of the target network.

Experiments based on the BNs from a well-known repository as well as two real-world case studies using different data sample sizes demonstrate that the proposed new approaches can achieve much greater learning accuracy compared to other state-of-the-art methods with relatively sparse data.

Acknowledgements

I would like to express my appreciation to my supervisor, Professor Norman Fenton, for leading me to this fancy research topic and for giving me the freedom to explore different exciting learning techniques in Bayesian networks. His observations and comments helped me to move forward with investigation in depth.

Many thanks to the other members of my thesis committee: Martin Neil, William Marsh and Timothy Hospedales for their useful comments and suggestions during the development of this research. I would like to especially thank Dr. Hospedales, for constantly giving me inspirations and guidances.

I would like to thank Peng Lin, Yongxin Yang and Barbaros Yet for sharing their expertise with me in the earlier stages of my thesis.

I would like to thank all my friends.

Special thanks to China Scholarship Council for providing part of the funding for my research.

Last, but not least, I would like to dedicate this thesis to my parents, Zhaohui Zhou and Ping Zhang, my fiancée Xiaojing Chen, for continuously encouraging and supporting my study.

Glossary of Abbreviations

AAAI The Conference of Association for the Advancement of Artificial Intelligence.

AIC Akaike Information Criterion.

AIJ Artificial Intelligence Journal.

AUC Area Under the Curve.

BDe Likelihood-equivalence Bayesian Dirichlet score.

BDeu Bayesian Dirichlet equivalent uniform score.

BIC Bayesian Information Criterion.

BNs Bayesian Networks.

BNT Bayes Net Toolbox for Matlab.

CO Constrained Convex Optimization.

DAG Directed Acyclic Graph.

DDJT Dynamic Discretization Junction Tree Algorithm.

EM Expectation Maximization.

IJAR International Journal of Approximate Reasoning.

IJCAI International Joint Conference on Artificial Intelligence.

JMLR Journal of Machine Learning Research.

JSD Jensen-Shannon Divergence.

K-L Kullback–Leibler Divergence.

MAP Maximum a Posteriori Estimation.

MDL Minimum Description Length.

MLE Maximum Likelihood Estimation.

MPL Multinomial Parameter Learning Model.

MPL-C Multinomial Parameter Learning Model with Constraints.

MPL-EC Multinomial Parameter Learning Model with Constraints.

MPL-TC Multinomial Parameter Learning Model with Transferred prior and Constraints.

NPT Node Probability Table.

UAI The Conference on Uncertainty in Artificial Intelligence.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Bayesian networks (BNs) (Pearl, 1988; Fenton and Neil, 2012) have become increasingly popular in the AI field during the last two decades because of their ability to model probabilistic causal relationships among variables describing many real-world problems; these include: medical diagnosis (Velikova et al., 2014), stock market prediction (Al Nasseri et al., 2014), fraud detection (Sá et al., 2014), neuron classification (López-Cruz et al., 2014), anomaly detection in vessel tracks (Mascaro et al., 2014) etc. BNs constitute a widely accepted formalism for representing knowledge with uncertainty and efficient reasoning. Moreover, unlike other machine learning techniques such as Neural Networks or Support Vector Machines, BNs are relatively easy to interpret by a non-expert.

A BN consists of a directed acyclic graph (DAG) that represents the dependencies among related nodes (variables), together with a set of local probability distributions attached to each node (called a node probability table – NPT – in this thesis) that quantify the strengths of these dependencies.

Constructing a BN from data is widely accepted as a major challenge in real-world applications. For many critical problems, there is little or no direct historical data to draw upon. The challenge is especially acute when the problem involves novel or rare systems and events (Fenton and Neil, 2012) (e.g., think of novel project planning (Khodakarami et al., 2007), predicting events like accidents (Ancel et al., 2014), terrorist attacks (Ezell et al., 2010), digital forensic investigation (Overill et al., 2012), and cataclysmic weather events (Ban et al., 2014)).

There are two typical categories of challenges in learning BNs: one is structure learning, where the BN structure is unknown; and the other is parameter learning given

a fixed graphical structure of the BN. The problem of BN learning has been studied in depth over the last two decades, and a considerable number of learning algorithms have been developed. Ideally, with sufficient data, these learning algorithms like the score-based algorithms, constraint-based algorithms or hybrids of them can learn a good BN that fits the data accurately (according to certain criteria). During the learning, two tasks are performed:

- *Learning the graphical structure*. This involves determining the (in)dependencies between nodes, and how well a candidate DAG fits the data.
- *Table learning (also called parameter learning)*. This involves estimating the NPT entries for each node (i.e., the prior and conditional probabilities).

In this thesis, we are only interested in algorithms for learning the parameters of BNs. Many previously developed parameter learning algorithms fail in real-world BN learning problems where only scarce data are provided. The challenge is to develop more accurate algorithms where only scarce data are provided, by exploiting expert knowledge and knowledge from related domains. Related domains are also referred to as source domains in transfer learning (Torrey and Shavlik, 2009), and used to improve learning in the original problem domain (target domain). Expert judgments are widely available in some real-world applications. For example, a doctor may say: "all the other risk factors can be ignored (have little additional influence) when deciding on a diagnosis of lung cancer given that the patient is a smoker with a long smoking history". Knowledge coming from medical books may state: smoking increases the risk of lung cancer. Related knowledge is, for example, a medical diagnosis model trained in a big inner-city hospital in England that might be used or help train a similar model in a small country hospital in Ireland.

1.1 Research Hypothesis and Objectives

The major research hypothesis in this thesis is that, by incorporating small amounts of expert judgments and/or related data it is possible to learn more accurate BN models than is possible with current state-of-the-art methods. In other words, this thesis proposes new BN learning algorithms which overcome previous limitations. The research hypothesis involves the following research objectives:

I Review and analyse previously proposed algorithms for BN learning in general and parameter learning with both data and knowledge in particular.

II Incorporate natural qualitative expert judgments or domain knowledge such that when combined with data, more accurate BN models can be built.

III Investigate the way to reduce the burden of eliciting expert judgments, and propose new or updated models that are more appropriate for parameter learning with these judgments.

IV Propose new algorithms to find the most relevant source BN or BN fragments to transfer, and to fuse source and target knowledge in a robust way.

V Propose extended algorithms for generic parameter learning with both expert judgments and transferred knowledge, which leverages the benefits of both constraint-based and transfer-based parameter learning algorithms.

1.2 Research Methodology

In order to address the research hypothesis and objectives stated in Section 1.1, we adopt a number of different research methods.

Starting with objective I, we perform two literature reviews and literature analyses. The first is focused on algorithms for BN learning in general, while the second is focused on algorithms for parameter learning with scarce data and additional domain knowledge.

For objectives II, III and IV, we investigate the existence of expert judgments and related knowledge in real-world problems (namely a case study from software development and a medical case study on trauma care). Then, we develop the new parameter learning models and algorithms to integrate expert judgments or related knowledge. The models and algorithms implemented in Java^{TM} and MATLAB are largely based on functions and subroutines from the AgenaRisk[1] Java application program interface (API) by Agena and the MATLAB toolbox BNT[2] by Kevin Murphy.

For objective V, no relevant work has exploited this before. In order to develop a reasonable framework to address this objective, we analyse many papers about other

[1] http://www.agenarisk.com/
[2] https://code.google.com/p/bnt/

types of fusion algorithms and integration methods (from premier conferences and journals in this field, i.e. UAI, AAAI, IJCAI, AIJ, JMLR, IJAR, etc.). Hence our generic parameter learning approach is based on existing conventions where appropriate.

In order to evaluate the proposed algorithms according to the identified performance measures, we perform a series of experiments using the implementations of the corresponding algorithms and publicly available BNs[3]. However, public availability of real-world BN data is very limited due to confidentiality and proprietary rights. An alternative to real-world data is to simulate data, which has the advantages that the reproducibility of experiments is enhanced, under the assumption that the data can more easily be made publicly available (downloaded) or accurately re-created given the parameters and details of the simulation process.

The main drawback of using simulated data is that validity or generalisability may be questioned. This is especially relevant in the transfer learning setting, as the simulated similarities between target and source domains may not reflect the actual similarities encountered in a real BN application. In this thesis, we evaluate the performance of the proposed algorithms using both real and simulated judgments and related problem domains. Moreover, we reproduce a number of previously published algorithms on these datasets.

1.3 Scientific Contributions and Work Already Published

In real-world BN learning, there is typically limited data. Therefore, most of the network structures are handcrafted by domain experts, and their parameters are estimated with both data and domain knowledge to avoid empty or unreliable estimations for some subsets of parameters. In this thesis, the major concern is developing new algorithms for BN parameter learning[4] with limited data. The contributions of this thesis are as follows:

- We present a multinomial parameter learning method, which can easily incorporate both expert judgments and data in the parameter learning process in a

[3] `http://www.bnlearn.com/bnrepository/`

[4] Ideally, with sufficient data, classical learning algorithms like maximum likelihood estimation (MLE) can produce accurate estimation.

much richer and less constrained way than the current state-of-the-art modelling and tools. This method uses an auxiliary BN model to learn the parameters of a given BN. The auxiliary BN contains continuous variables and the parameter estimation amounts to updating these variables using an iterative discretization technique.

- The expert judgments are provided in the form of constraints on parameters divided into two categories: *interior* and *exterior* constraints that constrain parameters under the same or different parent state configurations respectively.

  For interior constraints, we discuss the linear inequality constraints and approximate equality constraints. We propose a guideline to help identify those parameters where expert judgment should be most focused. Also, we discuss some common verbal expressions of judgments and their associated approximate equality constraints.

  For exterior constraints, we explore the way of automatically generating such constraints from monotonic causalities in a BN. We also discuss the overall margins in these constraints. Moreover, we present an extended multinomial parameter learning method to deal with the parameter learning with these exterior constraints.

- We present a novel transfer learning algorithm. This algorithm involves splitting the target and source BNs into fragments and then reasoning explicitly about both network-level and fragment-level relatedness. We achieve this via an Expectation Maximization (EM) style algorithm that alternates between (i) performing a Bayesian model comparison to infer per fragment relatedness and (ii) updating a source network relatedness prior. Finally, the actual transfer is performed per-fragment using Bayesian model averaging to robustly fuse the source and target fragments.

- We present a novel generic framework that improves the BN parameter learning accuracy with both qualitative constraints and transferred knowledge. The new parameter learning paradigm converts the transferred source parameters as the prior distributions which are encoded in the auxiliary multinomial parameter

learning model of the target domain.

- To validate our approach, we perform experiments on both synthetic and real-world BNs. We compare our models with standard well-known baseline models (such as *Asia*, *Alarm*, *Hailfinder*, etc.) using the standard K-L divergence measurement. Additionally, the method is evaluated in a real-world software defects prediction BN model and a real-world Trauma Care medical case study. Experimental results demonstrate the superiority of our method at various data sparsity and source relevance levels compared to conventional learning algorithms and other state-of-the-art parameter transfer methods.

Much of the work in this thesis is based on articles that have already been published and papers currently under revision (with Yun Zhou being the main author in each case). The following list provides a summary of the peer-reviewed publications and how they contribute to the thesis (see Table 1.1).

Table 1.1: List of publications with references to the research objectives and the thesis chapters.

Publication	Referred objective	Referred chapter
(1)	I, II	Chapter 1, 2
(2)	I, II	Chapter 2, 3
(3)	I, III	Chapter 4
(4)	I, IV	Chapter 5
(5)	II, III	Chapter 4
(6)	V	Chapter 6

(1) **Yun Zhou**, Norman Fenton, Martin Neil, and Cheng Zhu. "Incorporating Expert Judgment into Bayesian Network Machine Learning." In *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*, AAAI Press, 2013: 3249-3250.

This paper was the first to present the idea of using an auxiliary model to deal with parameter learning with constraints. Some preliminary experiments are carried out to show the plausibility of incorporating expert provided constraints in parameter learning. This paper contributes to objectives I and II.

(2) **Yun Zhou**, Norman Fenton, and Martin Neil. "Bayesian Network Approach to

Multinomial Parameter Learning using Data and Expert Judgments." *International Journal of Approximate Reasoning*, 55.5, 2014: 1252-1268.

This paper presents the detailed description of a multinomial parameter learning method, which can easily incorporate both expert judgments and data during the parameter learning process. This method uses an auxiliary BN model to learn the parameters of a given BN. The auxiliary BN is referred to as MPL-C model, which contains continuous variables and the parameter estimation amounts to updating these variables using an iterative discretization technique. The expert judgments are provided in the form of constraints on parameters divided into two categories: linear inequality constraints and approximate equality constraints. A number of well-known sample BNs are evaluated in the experiments, which show the benefits of this approach in real-world applications. Similar to Publication (1), the main contribution of this paper is the development, implementation and evaluation of the MPL-C model.

(3) **Yun Zhou**, Norman Fenton, and Martin Neil. "An Extended MPL-C Model for Bayesian Network Parameter Learning with Exterior Constraints." *Probabilistic Graphical Models*, Springer International Publishing, 2014: 581-596.

In this paper, we extend our initial work on parameter learning with expert judgments (Publication (2)) and introduce the exterior constraints generated from monotonic causalities. Analogously to the MPL-C model, the model in this paper also is a hybrid Bayesian network, and the learning is achieved by the inference in the auxiliary model. We outline the difference between interior constraints and exterior constraints. The results from some preliminary empirical investigations on standard small BNs and an empirical software defects prediction BN illustrate the benefits of using such exterior constraints in parameter learning. This paper contributes to the objective III.

(4) **Yun Zhou**, Timothy Hospedales, and Norman Fenton. "When and Where to Transfer for Bayes Net Parameter Learning." *Machine Learning Journal*, under revision, 2015.

This paper is concerned with the problem of BN parameter transfer learning.

The main contributions of the article are two-fold. The first is the proposal and analysis of a Bayesian model comparison fitness function, which can be used to measure the relatedness of different BN domains. The second contribution of this paper concerns the robust fusion of the target and source BN fragments. The results from both synthetic and empirical investigations on a wide collection of BNs illustrate the potential benefits of using our parameter transfer learning algorithm. This paper contributes to objectives I and IV.

(5) **Yun Zhou**, and Norman Fenton. "An Empirical Study of Bayesian Network Parameter Learning with Exterior Constraints." *International Journal of Approximate Reasoning*, under revision, 2015.

One of the main objectives of this paper is to empirically investigate whether the monotonic causalities exist in real-world BNs. To this end, we investigate 12 BNs from a publicly available repository. Results from empirical investigations confirm the hypothesis that the monotonic causalities are fully or partially present in these BNs. Moreover, experiment results show that incorporating exterior constraints generated from monotonic causalities can improve the parameter learning performance in scarce data situation. Hence, it is concluded that incorporating such constraints is non-trivial in real-world BN applications. The paper mainly contributes to objectives II and III.

(6) **Yun Zhou**, Norman Fenton, Timothy Hospedales, and Martin Neil. "Probabilistic Graphical Models Parameter Learning with Transferred Prior and Constraints" *31st Conference on Uncertainty in Artificial Intelligence*, AUAI, 2015: 972-981. (Oral Presentation.)

This is the first paper to present a generic framework that combines both approaches (introducing expert judgments and transferring knowledge from related domains) to improve BN parameter learning. In this approach, we generalise the state-of-the-art MPL-C model (Publication (2)) for learning with expert constraints to also exploit knowledge from related source domains via a bootstrap approach. The new model called MPL-TC (Multinomial Parameter Learning model with Transferred prior and Constraints) synergistically exploits both forms of external knowledge to improve learning performance in the target BN. This paper

mainly contributes to research objective V.

1.4 Thesis Outline

The thesis is structured as follows:

Chapter 2 describes the background and start-of-the-art of BN learning and parameter learning with data and expert judgments/source domain knowledge. It also discusses the current research gap and identifies the key properties in this research topic.

Chapter 3 describes the auxiliary model of parameter learning with interior constraints. For the constraints, this chapter shows their categories, and how to elicit them from expert judgments. It also shows how to compute the estimators based on the dynamic discretization junction tree algorithm. In particular, we show two examples to estimate their parameters given data and related expert judgments.

Chapter 4 presents the auxiliary model of parameter learning with exterior constraints. This chapter also presents the method of generating such exterior constraints from monotonic causalities. Moreover, this chapter investigates the existence of the monotonic causalities in real-world BNs based on a publicly available repository.

Chapter 5 presents the parameter transfer learning algorithm which includes the definition of target and source BNs and fragments, and the fitness/fusion functions. Synthetic experiments are performed to validate the potential benefits of using this algorithm.

Chapter 6 presents a unified model for parameter learning with both constraints and transferred information. Both synthetic and empirical experiments are performed to show the benefits of this unified model.

Chapter 7 presents the final conclusions and contributions of this thesis and how they contribute to the research hypothesis and objectives. This chapter also highlights future ideas for extending and improving the research on BN parameter learning with scarce data.

Chapter 2

Related Work

In this chapter, we provide an overview of the concepts which are fundamental to this thesis. We explain the theory and definitions behind BNs, and give a brief introduction to BN learning from both a frequentist and a Bayesian point of view. We present several models for learning with scarce data and domain knowledge and report the standard learning performance measurement methods. Finally, we identify the research gap in the state-of-the-art that we seek to address in the rest of the thesis.

2.1 Bayesian Networks – The Basics

2.1.1 Informal Definition and Overview of BNs

Figure 2.1 shows the DAG of a BN (Pearl, 2011) that can be used for probabilistic analysis. Factors like "Rain" (whether or not it rains) and "Sprinkler" (whether or not the sprinkler is on) can determine the presence of 'Wet' (the wet pavement), and the consequence of it "Slippery" (whether or not the pavement is slippery). For instance, if the sprinkler is on, then the pavement is probably wet; if someone slips on the pavement, that also provides evidence that it is wet. On the other hand, if we see that the pavement is wet, that makes it more likely that the sprinkler is on or that it is raining; but if we then observe that the sprinkler is on, that reduces the likelihood that it is raining (this is called "explaining away"). Once this BN has been built with fully specified NPTs, it is an efficient tool for performing inferences. For example, we can compute the probability that the pavement will be slippery given that it has rained, and this is called "top down" reasoning (prediction). If we had evidence of an effect (pavement wet), we can also infer the most likely cause, which is regarded as "bottom up" reasoning

(diagnosis).

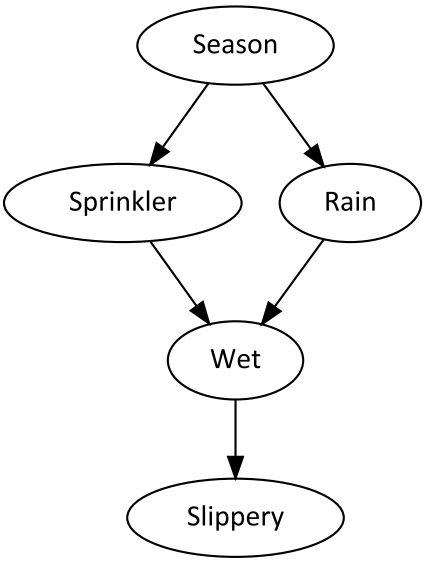


Figure 2.1: The directed acyclic graph of the wet pavement BN. The variables are: "Season", "Rain", "Sprinkler", "Wet" and "Slippery".

Table 2.1 shows the NPT of the "Wet" node in the BN example. The NPT has 8 probability values since the variable has 2 parents, and the node and both its parents have 2 states each. Given the historical data of "Wet" node under all state combinations (there are 4 in total) of its parent nodes, parameter learning is to fill every entry in this NPT.

Table 2.1: NPT of the "Wet" node in the example Bayesian network.

	Sprinkler	*true*		*false*	
	Rain	*true*	*false*	*true*	*false*
Wet	*true*	0.99	0.90	0.80	0.00
	false	0.01	0.10	0.20	1.00

2.1.2 Formal Definition of A BN

Let θ denote a set of numerical parameters of the categorical random variables in some set V, and let G represent a Directed Acyclic Graph (DAG), whose nodes $X_1, X_2, X_3, ..., X_n$ correspond to the random variables in V, and whose arcs represent the direct dependencies between these variables. Here $\theta = \{\theta_{ijk}\}$, and $\theta_{ijk} = p(X_i = k|\pi_i = j)$ represents a parameter for which X_i takes its k-th value and its parent set π_i takes its j-th value. As there is a one-to-one correspondence between nodes and variables, the terms 'node' and 'variable' are used interchangeably in this thesis. The Node

Probability Table (NPT) associated with every variable contains the (conditional) probability of each value of the variable given each instantiation of its parents in G, which is also referred to as an NPT parameter θ_{ijk}. An NPT column[1] $p(X_i|\pi_i = j)$ denotes the discrete probability distribution of X_i given the j-th state configuration of its parents ($\pi_i = j$).

Property 2.1.1. *Local Markov Condition: a variable X_i is conditionally independent of its non-descendants given its parent set π_i.*

We call (V, G, θ) a Bayesian network (BN) if it satisfies the *Local Markov Condition*. Taking advantage of this property, one can obtain a factor representation of the joint probability distribution over all the random variables. That means a BN encodes a simplified joint probability distribution over V given by:

$$p(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} p(X_i|\pi_i) \tag{2.1}$$

2.1.3 BN Inference Algorithms

Given a BN, inference is the process of computing the updated distribution of variables of interest, given that other variables are set to certain values.

The computational complexity of performing exact inference in a BN is known to be NP-hard (Cooper, 1990). However, in the late 1980s a major breakthrough was achieved with the development of exact inference algorithms that computed efficiently for a large class of real-world BNs. The most commonly used of these exact inference algorithms for discrete BNs is the junction tree algorithm (Lauritzen and Spiegelhalter, 1988), which propagates the evidence along the moralized and triangulated graph.

Graph moralization includes connecting nodes that have a common child and making all edges undirected. The triangulation step identifies subsets of nodes of the moral graph called "supernodes". After removing the supernode that is a subset of another supernode, we can insert edges between supernodes to create the tree structure called junction tree. Passing the messages between supernodes in the junction tree involves doing exact marginalization over the non-observed non-query variables in supernodes, which is inefficient for junction trees of large treewidth (the size of largest supernode

[1] Note in some other works, e.g., Netica BN software, each NPT row represents a discrete probability distribution given a parent configuration.

minus one). Therefore, many algorithms have been developed for building optimal junction trees. For a good survey of the literature, see the paper by Daly et al. (2011).

One of the fundamental weaknesses of popular BN inference algorithms is that they require all the variables to be discrete. But many real-world BNs involve continuous nodes as well as discrete nodes. Such BNs are referred to as *hybrid* BNs(Murphy, 1998), and they require special ways to handle inference.

In hybrid BNs, exact inference can only be performed when the network treewidth is small and the continuous nodes are assumed to be conditional Gaussian distributions (Lauritzen and Jensen, 2001) — a highly unrealistic assumptions in almost all real-world situations, which typically involve non-standard statistical distributions. Therefore, approximate inference is in general needed for hybrid BNs, and the most commonly used approach is to use static discretization whereby each continuous variable is converted to a discrete variable using pre-defined discretization intervals (Langseth et al., 2009). This is the only way to handle arbitrary continuous variables in popular BN software tools such as Hugin[2], and Netica[3]. However, there are severe problems with such an approach. Most notably, reasonable accuracy can only be achieved when the discretization is 'finest' in the highest density probability regions; but such regions change when inference is performed with different observations. Since the discretization has to be fixed in advance, reasonable accuracy can only be achieved by defining multiple intervals across the whole range, which results in a heavy cost of computational complexity. Fortunately, a dynamic discretization junction tree algorithm (DDJT) has been developed (Neil et al., 2007) and implemented in AgenaRisk[4] meaning that there is no need for any pre-defined discretization of continuous nodes in hybrid BNs. This dynamic discretization process (which we will describe in detail in Chapter 3) uses the relative entropy error to iteratively adjust the discretization in response to new evidence, and so achieves more accuracy in the zones of high posterior density. This algorithm provides a good balance between achieving high accuracy in the approximations and maintaining a reasonable computation cost to get the results. Because this thesis introduces auxiliary BNs (to support parameter learning) that are hybrid BNs, we use the DDJT algorithm for inference in order to achieve highly accu-

[2] http://www.hugin.com/
[3] https://www.norsys.com/
[4] http://www.agenarisk.com/

rate results.

2.2 Bayesian Network Learning

There are two typical categories of problems in BN learning: one is structure learning to identify the (in)dependencies between BN nodes; and the other is parameter learning to show the frequency of correlated events induced by the BN structure. Parameter learning is to fully specify the discrete probability distribution for each NPT column. Here we assume the NPTs can be parameterised, so the "parameters" of an NPT is a set of discrete probability NPT entries. Because the space of plausible DAGs in a BN is super-exponential in the number of variables, purely data-based structure learning is a challenging task.

2.2.1 Structure Learning

Over the years, there has been a great deal of work on different algorithms for improving BN structure learning accuracy and efficiency. Algorithms for estimating the DAG for a single BN generally fall into two broad classes[5]:

- ***Score-based algorithms.*** These search over the entire structure space to find an optimal one that best describes the observed data. The searching problem has been proven NP-hard by Chickering (1996).

- ***Constraint-based algorithms.*** These qualify the dependence and independence relationships between the variables, and reconstruct the structure that represents these relationships as far as possible.

Because the local score calculation is related to parameter learning, this thesis mainly discusses the class of score-based algorithms. Most of these algorithms (local search methods) take the following steps:

1. Randomly generate initial structures.

2. Defines metric which can be used to measure the quality of candidate network structures that reflect the extent to how the structure would fit the dataset. Most common metrics are: AIC metric (Akaike, 1998), MDL metric (Bouckaert,

[5]For detailed discussions we direct the reader to the books (Koller and Friedman, 2009; Barber, 2012).

1993), BIC metric (Schwarz et al., 1978), K2 metric (Cooper and Herskovits, 1992), BDe metric (Heckerman et al., 1995). A good scoring criterion should reward both a better match of the data to the structure, and a simpler structure.

3. Score candidate structures.

4. Identify the structure with maximal score. Search algorithms like K2 (Cooper and Herskovits, 1992), Hill climbing (Buntine, 1996), Repeated hill climbing, Max-Min hill climbing (Tsamardinos et al., 2006), Simulated annealing (Heckerman et al., 1995), Tabu search, Genetic search (Larrañaga et al., 1996), A* search (Yuan et al., 2011) and BFBnB search (Malone et al., 2011) are applied.

A detailed discussion and recent findings (which encode BN structure learning problems as an integer program (IP)) on structure learning can be found in (Cussens, 2011; Nie et al., 2014; Bartlett and Cussens, 2015).

Next, we present the basic formulation of score-based BN learning. We use the following notation:

- r_i represents the cardinality of random variable X_i in a BN.
- $|\pi_i|$ represents the number of state configurations of π_i. Thus, $|\pi_i| = \prod_{X_j \in \pi_i} r_j$ with $|\pi_i| = 1$ implying $\pi_i = \emptyset$.
- $D = \{d_1, d_2, ..., d_{|D|}\}$ is a dataset with $|D|$ data instances that are independently, and randomly sampled from some underlying distribution.
- d_l is a complete case of D, which is a vector of values of each variable.
- N_{ijk} ($1 \leq i \leq n$, $1 \leq j \leq |\pi_i|$ and $1 \leq k \leq r_i$) is the number of data instances in sample D for which X_i takes its k-th value and its parent set π_i takes its j-th value, which satisfies the equation:

$$N_{ijk} = \sum_{l=1}^{|D|} I_{(X_i=k \text{ and } \pi_i=j \text{ in } d_l)} \tag{2.2}$$

where $I_{(\cdot)}$ is the indicator function whose value equal to 1 if the condition in $(\cdot)$ is satisfied, otherwise its value equal to 0.

- $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$, which is the total number of instances in dataset D for which π_i takes its j-th value.

The log-likelihood score of a BN structure G is related to the compression that can be achieved over the dataset D with an optimal DAG induced by G:

$$\ell(G,D) = \sum_{i=1}^{n} \sum_{j=1}^{|\pi_i|} \sum_{k=1}^{r_i} N_{ijk} \log \frac{N_{ijk}}{N_{ij}} \tag{2.3}$$

This log-likelihood score has the decomposability property - the score of a particular structure can be obtained by the score for each node given its parents:

Property 2.2.1. *Decomposability of log-likelihood: the log-likelihood function can be written as the sum of n components, one for each variable in the BN. The component corresponding to variable X_i can be decomposed further into a sum of $|\pi_i|$ subcomponents, one for each instantiation of the parents of X_i. If there are no constraints between NPT columns describing different subcomponents, then this decomposability will allow us to efficiently perform log-likelihood calculation by solving a collection of smaller optimization problems, one for each subcomponent.*

Conventional score-based algorithms for structure learning make use of certain heuristics to find the optimal DAG that best describes the observed data D over the entire space. We define

$$\hat{G} = \arg\max_{G \in \Omega} \ell(G,D) \tag{2.4}$$

$\ell(G,D)$ is the log-likelihood score, which is the logarithm of the likelihood function of the data that measuring the fitness of a DAG G to the data D.

Ω is a set of adjacent matrices $\Omega = \{A \in \mathbb{Z}^{n \times n}\}$ to encode set of the candidate DAGs, e.g. an arc $A_{ij} = 1$ in A represents a dependence relationship between X_i and X_j ($X_i \rightarrow X_j$).

Unfortunately, maximizing the log-likelihood will most often result in a complete graph , which states that every pair of nodes is conditionally dependent. The major scope of structure learning studies focus on how to avoid this problem from the maximum likelihood estimation and infer, instead, a sparse graph structure. Specifically, an additional term is introduced, namely the total number of parameter values encoded in a DAG,

$P_G = \sum_{i=1}^{n}(r_i - 1)|\pi_i|$.

Thus, the resulting adjacent matrix has some zero entries owing to the effect of this regularization term. Now, this estimation problem is defined as:

$$\max_{G\in\Omega} \ell(G,D) - \eta \tag{2.5}$$

where η is the penalty term, and

- $\eta = P_G$ (in the AIC metric),
- $\eta = \frac{P_G}{2}\log|D|$ (in the BIC/MDL metric).

Some newer scoring criteria for BNs include MML score (minimum message length metric (Wallace et al., 1996)), mutual information tests metric (Campos, 2007), SparsityBoost score (Brenner and Sontag, 2013), Min-BDeu and Max-BDeu metric (Scanagatta et al., 2014), etc.

Assume prior probability of the DAG G is uniform over DAGs, and the associated NPT column's prior $p(X_i|\pi_i = j)$ is a Dirichlet distribution with parameters $\alpha = \{\alpha_{ijk}|1 \le i \le n, 1 \le j \le |\pi_i|, 1 \le k \le r_i\}$. Thus, we can have the most commonly used Bayesian Dirichlet score metric (BD):

$$S_{BD}(G,D) = \sum_{i=1}^{n}\sum_{j=1}^{|\pi_i|}(\log\frac{\Gamma(\alpha_{ij})}{\Gamma(\alpha_{ij}+N_{ij})} + \sum_{k=1}^{r_i}\log\frac{\Gamma(\alpha_{ijk}+N_{ijk})}{\Gamma(\alpha_{ijk})}) \tag{2.6}$$

where $\Gamma(\cdot)$ is the *Gamma* function, and $\Gamma(n) = (n-1)!$ if n is an integer.

The hyperparameter α_{ijk} satisfies the restriction that $\alpha_{ij} = \sum_{k=1}^{r_i}\alpha_{ijk}$.

The K2 score assumes the $\alpha_{ijk} = 1$, thus $\alpha_{ij} = r_i$.

The BDe score introduced in (Heckerman et al., 1995) is a likelihood-equivalent specialization of the BD score, which assumes

$\alpha_{ijk} = \alpha^* p(X_i|\pi_i = j)$

where α^* is the Equivalent Sample Size (ESS), and $p(X_i|\pi_i = j)$ conveys the prior beliefs about the occurrences of configuration $\{X_i = k, \pi_i = j\}$. If the distribution $p(X_i|\pi_i = j)$ is uniform, then we can get the BDeu score,

$\alpha_{ijk} = \frac{\alpha^*}{|\pi_i|r_i}$ $(\forall i,j,k)$.

The series of Bayesian Dirichlet scores aims at maximizing the posterior probability of the candidate structure G given D. However, the discovered optimal DAG

is highly sensitive to the chosen α^* (Steck and Jaakkola, 2002; Silander et al., 2012). Thus, recent research also focuses on developing new extensions of BDeu and choosing the right score in different BN learning scenarios (Scanagatta et al., 2014).

Recently, domain knowledge or multiple datasets have been introduced to help BN structure learning. For example, de Campos et al. (2009a) and de Campos and Ji (2011) apply domain knowledge such as different structure constraints to restrict the searching space of score-based algorithms. It is also shown that BDeu score has good properties to prune the search space (de Campos and Ji, 2010).

Cano et al. (2011) and Masegosa and Moral (2013) proposed an interactive structure learning approach that iteratively queries the domain expert about the reliability of learnt edges. This paradigm uses expert knowledge to boost the reliability of the learnt structure, especially in the limited data situation.

In some BN structure learning studies (Niculescu-mizil and Caruana, 2007; Oyen and Lane, 2012; Oates et al., 2014), multiple datasets are exploited to learn robust BN structures. For example, in group lung cancer studies, they learn functional BNs based on the search-and-score approach for several age groups and treat the data from each group as a task. They assume the tasks that are close in age should have similar network structures. The shared information between tasks helps the BN structure learning in situations with scarce data. These recent findings show both domain knowledge and additional datasets can improve BN structure learning.

2.2.2 Parameter Learning

2.2.2.1 Generative Learning

Given a fixed BN structure G, we are more interested in the generative parameter learning (Su et al., 2008), which is efficient and maximizes the data log-likelihood. The frequency estimation approach is a widely used generative learning technique, which determines parameters by computing their associated frequencies in the dataset. This approach is also referred to as maximum likelihood estimation (MLE). Based on the above definitions, the parameters in the BN can be represented as:

$$\theta = \{\theta_{ijk} | i = 1, ..., n;\ j = 1, ..., |\pi_i|;\ k = 1, ..., r_i\}$$

The MLE method tries to estimate the set of parameters θ that best describe the dataset. In general, this translates into finding the set of parameters that maximize the

data log-likelihood:

$\ell(\theta, D) = \log p(D|\theta)$.

Given the equation (2.3) and decomposability property of log-likelihood (Property 2.2.1), the estimation of θ can be represented as:

$$\arg\max_{\theta} \ell(\theta, D) = \arg\max_{\theta_{ijk}} \sum_{i=1}^{n} \sum_{j=1}^{|\pi_i|} \sum_{k=1}^{r_i} N_{ijk} \log \theta_{ijk} \tag{2.7}$$

We can see the estimation process is divided into a set of local calculations. Specifically, the local calculation (MLE) for θ_{ijk} is:

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ij}} \tag{2.8}$$

Previously, we have discussed the Dirichlet prior of an NPT column θ_{ij}. Intuitively, one can think of a Dirichlet distribution as an expert's guess of an NPT column. Parameters α_{ijk} can be thought of as how many times the expert believes he/she will observe $X_i = k$ in a sample of α_{ij} examples drawn independently at random from θ_{ij}. The Dirichlet distribution can be applied to represent the prior distribution for NPT column θ_{ij} in the BN, which has the following equation:

$$p(\theta_{ij}) = \frac{1}{Z_{ij}} \prod_{k=1}^{r_i} \theta_{ijk}^{(\alpha_{ijk}+1)-1} \ (\sum_{k} \theta_{ijk} = 1, \ \theta_{ijk} \geq 0, \ \forall k) \tag{2.9}$$

where Z_{ij} is a normalization constant to enforce that $\int_{-\infty}^{\infty} \prod_{k=1}^{r_i} \theta_{ijk}^{(\alpha_{ijk}+1)-1} d\theta_{ijk} = 1$.

Property 2.2.2. *Dirichlet multinomial conjugacy: assume $p(\theta)$ is a Dirichlet prior distribution, and D a dataset of fully observable cases drawn independently at random from the probability distribution represented by a BN, the posterior distribution $p(\theta|D) \propto p(D|\theta)p(\theta)$ is also a Dirichlet distribution.*

Based on the above discussion and Property 2.2.2, we can introduce another classical parameter learning approach called Maximum a Posteriori (MAP) estimation:

$$p(\theta|D) \propto p(D|\theta)p(\theta) \propto \prod_{i=1}^{n} \prod_{j=1}^{|\pi_i|} \prod_{k=1}^{r_i} \theta_{ijk}^{N_{ijk}+\alpha_{ijk}} \tag{2.10}$$

As a result, the MAP for θ_{ijk} is:

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + \alpha_{ijk}}{N_{ij} + \alpha_{ij}} \tag{2.11}$$

Intuitively, one can think of the hyperparameter α_{ijk} in the Dirichlet prior as an expert's guess of the virtual data counts of the parameter θ_{ijk}. When there is no related expert judgment, people usually use:

- K2 prior $\alpha_{ijk} = 1$,
- BDeu prior $\alpha_{ijk} = \frac{1}{|\pi_i|r_i}$ (Bayesian Dirichlet likelihood equivalent uniform prior (Heckerman et al., 1995)).

2.2.2.2 Learning with Missing Data

When D is incomplete, the Expectation Maximization (EM) algorithm is typically employed. For each data vector $d_l = \{z_{l1}, \ldots, z_{lo}, ?_{l1}, \ldots, ?_{lh} | z_{lj} \in Z_l, ?_{lj} \in Y_l\}$, where Z_l and Y_l represent the set of observed and unobserved variables in the l-th data vector, and $o+h=n$ (the total number of variables in a BN), EM starts with some initial parameters $\theta^{(0)}$, and includes two main steps performed alternatively until convergence:

- *E-step*: calculate the expected value of the log-likelihood function, with respect to the conditional distribution of Y_l given Z_l under the current estimate of parameters $\theta^{(t)}$:

  $$\ell(\theta|\theta^{(t)}) = \sum_l \sum_{y_{l1},\ldots,y_{lh} \in \{Y_l = \#\}} p(y_{l1}, \ldots, y_{lh} | Z_l, \theta^{(t)}) \log p(Z_l, y_{l1}, \ldots, y_{lh} | \theta^{(t)})$$

  where $y_{l1}, \ldots, y_{lh}$ represents an instantiation of unobserved variables Y_l in the d_l. And the set of possible instantiations of Y_l is denoted by $\{Y_l = \#\}$.

- *M-step*: compute the next estimate of parameters $\theta^{(t+1)}$ by maximizing the expected log-likelihood function in the first step:

  $$\theta^{(t+1)} = \arg\max_\theta \ell(\theta|\theta^{(t)})$$

More discussions and extensions of this algorithm can be found in text books (Darwiche, 2009; Koller and Friedman, 2009; Murphy, 2012; Barber, 2012). Recently, Broeck et al. (2014) proposed an inference-free, closed-form method for consistently

learning BN parameters from MCAR (missing completely at random) and MAR (missing at random) datasets. This method is based on the idea of a 'missingness graph' which is an augmented BN to represent the causal relationships for missingness (Mohan et al., 2013), and has significant computational advantages over EM.

2.2.2.3 Learning as Inference in Auxiliary Graphical Models

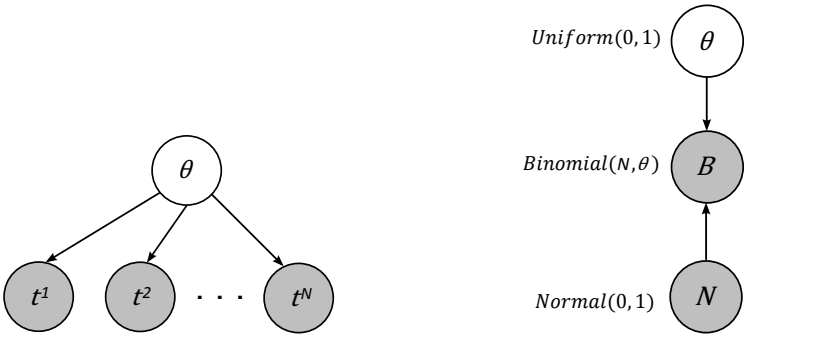
Given data, parameter learning is to estimate their distributions. This process can be represented as auxiliary BN models, whose nodes encode the parameter distributions and data observations. Let us start with a simple coin tossing problem:

$t^i = true$ if on i-th toss the coin comes up heads, $t^i = false$ if it is tails.

The parameter learning target is to estimate the probability θ that the coin will be a head. If the coin is fair, the probability prior satisfies:

$\theta' = p(t^i = true|\theta) = p(t^i = false|\theta) = 0.5$

Let us assume N independent tosses of the coin are made, and each toss is *identically and independently distributed* (i.i.d. assumption). Thus, we can have the auxiliary BN model (as shown in Figure 2.2 (a)) for this.



(a) The original representation **(b)** The compact representation

Figure 2.2: Graphical model representations for coin tossing problem.

Therefore, the parameter learning process is to infer the posterior distribution of probability θ:

$p(\theta|t^1,t^2,...,t^N) \propto p(\theta)\prod_{i=1}^{N} p(t^i|\theta) = p(\theta)\prod_{i=1}^{N} \theta^{\sum I_{(t^i=true)}}(1-\theta^{\sum I_{(t^i=false)}})$

here $\sum I_{(t^i=true)}$ represents the total number of occurrences of heads.

As we can see, the number of required nodes increases with the number of tosses, leading to an unrealistically large BN model. Fortunately, using the AgenaRisk toolset we can model this problem in a very compact way using the *Binomial* distribution (each

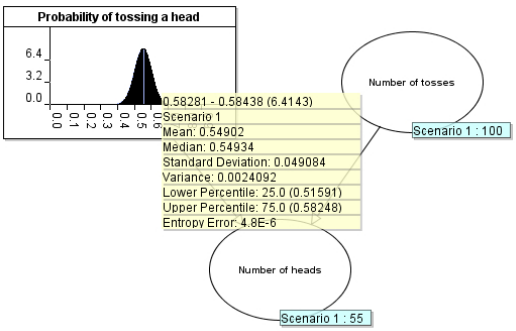
toss of the coin is treated as a *Bernoulli* trial). The compact representation is shown in Figure 2.2 (b).

The node *B* encodes the *Binomial* distribution with two hyperparameters:

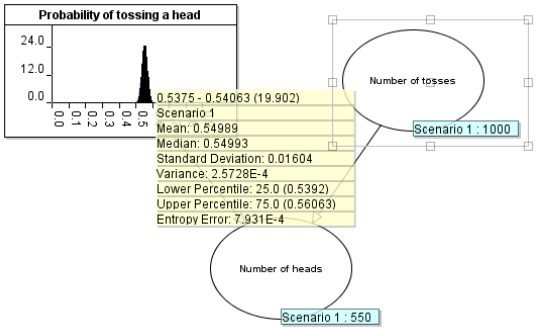
(i) θ – the success probability (the probability of the coin will be a head) in each trial/toss, and

(ii) N – the total number of trials/tosses, which is represented by a *Normal* distribution[6] and has an infinite range.

Therefore, after setting the total number of tosses and occurrences of heads in nodes N and B in the compact model, the posterior probability θ can be inferred.



(a) Updated distribution of parameter θ after observing 55 heads in 100 tosses



(b) Updated distribution of parameter θ after observing 550 heads in 1000 tosses

Figure 2.3: Updating the learnt value of parameter θ for different number of tosses and observations in AgenaRisk.

[6]This can be replaced with *Poisson* distribution to only allow positive integers. Because this root node is always observed with a valid number of trials during the inference, using *Normal* or *Poisson* distribution will produce the same results.

As discussed above, conventional BN inference algorithms cannot be applied accurately here due to the fact that the θ node is continuous and the other two nodes are potentially infinite integer nodes. Thus, the dynamic discretization junction tree algorithm is employed to update such BNs in the thesis. For example, Figure 2.3 shows the results in AgenaRisk of the updated distribution of the probability of tossing a head parameter in two different scenarios (the first when 55 heads are observed after 100 tosses and the second after 550 heads are observed after 1000 tosses). In each case we assume the prior distribution is *Uniform*(0,1).

2.3 Parameter Learning with Scarce Data and Domain Knowledge

In real-world problems that we wish to model as a BN, there is typically limited or no relevant data to learn either the structure or the parameters (Fenton and Neil, 2012). Therefore, the structure is usually hand-crafted by domain experts, and the parameters are estimated with both scarce data and domain knowledge to avoid empty or unreliable estimations for some subsets of parameters.

2.3.1 Active Parameter Learning

When training data is scarce, a straightforward BN parameter learning goal is to achieve greater accuracy with fewer training data instances. This goal can be realized by the active learning algorithm (Settles, 2010), which is also known as *optimal experimental design* in the statistics literature.

The first BN active learning algorithm was presented in previous work (Tong and Koller, 2001a,b). This algorithm assumes that some variables in V are controllable. This means the algorithm can select a subset of variables $Q \subset V$ and their particular instantiations q. We call $Q = q$ a query, and its result is a randomly sampled data instance d_{l+1} conditioned on $Q = q$. The sampled data instance will be integrated into the former data D to update the parameters in the BN.

The key step in the active learning algorithm is the mechanism to update the parameters and select the proper query based on the prior distribution $p(\theta)$. The *update rule* is very simple – the randomly sampled instances are only used to update variables, which are not in Q or their children in Q. The *select rule* is to find the query that would

most improve the correctness of the current estimations (Tong and Koller, 2001a). The measurement is carried out by Kullback-Leibler divergence (K-L divergence) (Kullback and Leibler, 1951) that will be discussed in depth in the next section.

As discussed above, active learning needs a good prior distribution of parameters to guide the query selection and affect the final learning results. However, such a prior is rarely available in many real-world applications, especially in scarce data situation. Thus, we next discuss the alternatives for parameter learning in this situation.

2.3.2 ICI Based Parameter Learning

In real-world applications of BN parameter learning, if it is difficult to collect more related data or elicited numerical assessment about the estimated parameters, an alternative way to improve the overall learning accuracy is to reduce the number of estimated parameters by introducing domain assumptions about the NPT.

The *Independence of Causal Influences* (ICI) models (see (Diez and Druzdzel, 2006) for a comprehensive review), provide a solution by assuming that parent variables contribute independently of each other to the child node. The benefit of this assumption is such that the number of required parameters is linear, rather than exponential, in the number of parent variables.

Pearl (1988) built the first ICI model, referred to as Noisy-OR model, which is a probabilistic extension of the logical OR relation. The variables in this model are all binary, e.g., the variable X_i can be either *true* or *false*. Based on the ICI assumption, each true parent event can independently produce the child effect, i.e.,

$$p(X_1 = true | X_2 = false, ..., X_i = true, ..., X_n = false) = z_i$$

If none of the parent variables $X_2, ..., X_n$ are *true*, then neither is the child variable X_1, i.e., $p(X_1 = false | X_2 = false, ..., X_n = false) = 1$. Given the parameter values $z_2, ..., z_n$, we can derive all the other parameters in the original NPT of X_1 (Noisy-OR's amechanistic property):

$$p(X_1 = true | X_2, ..., X_n) = 1 - \prod_{X_i = true} (1 - z_i) \tag{2.12}$$

To allow the unmodelled parents of X_1, Henrion (1988) extended the Noisy-OR model by introducing a leak probability z_L. Thus even if $X_2, ..., X_n$ are all *false*, the X_1

still has z_L probability in *true* state. The equation (2.12) can be rewritten as:

$$p(X_1 = true|X_2,...,X_n) = 1-(1-z_L)\prod_{X_i=true}(1-z_i) \tag{2.13}$$

To generalize the binary restriction of variable states, Henrion (1988) and Diez (1993) developed the Noisy-MAX model. The usage of the Noisy-OR and Noisy-MAX in practice was discussed in the work (Zagorecki and Druzdzel, 2013). Moreover, recent work (Woudenberg and van der Gaag, 2015) also shows that inappropriate use of these models can harm the BN's performance.

In recent years, renowned extensions of the ICI model include the Ranked Node (Fenton et al., 2007) and NIN-AND tree (Xiang and Jia, 2007; Xiang and Truong, 2014) models. Although these models could greatly reduce the number of estimated parameters, when the training data is extremely scarce, we still need numerical assessment of parameters in these models. Thus we are still using quantitative domain knowledge.

Previous work has shown that elicited qualitative domain knowledge is more reliable than quantitative knowledge (Feelders and van der Gaag, 2006). Next, we will discuss the parameter learning with such qualitative knowledge.

2.3.3 Constrained Parameter Learning

A widely used type of qualitative knowledge is the expert provided parameter constraint (Druzdzel and van der Gaag, 1995). A simple example of such a statement could be: "the probability of people getting cancer is smaller than 0.01". Such easily elicited constraints can greatly improve the learning accuracy, especially in scarce data situation. Because of such benefits, there is an increasing research interest in incorporating constraints into parameter learning.

The constrained convex optimization (CO) formulation (Niculescu et al., 2006; de Campos and Ji, 2008; de Campos et al., 2008, 2009a; Liao and Ji, 2009) is the most popular way to estimate the constrained parameters. In this setting, the algorithm seeks the global optimal estimation (maximal data log-likelihood) with respect to the parameter constraints. The parameters also can be estimated by the Monte Carlo method (Chang et al., 2008), where only the samples that are consistent with the constraints are kept. Recently, our own work on auxiliary BN models (Zhou et al., 2013, 2014a,b) has been developed for solving this problem. In this approach, the target parameters, data

observations and elicited constraints are all modelled as nodes in the auxiliary BNs. Thus, the parameters are estimated via the inference in the auxiliary BN. The thesis is mainly based on this method whose detail will be given in Chapter 3.

To further reduce the burden of expert elicitation, researchers (Wellman, 1990) find the monotonic influence property in some BNs, which can be used to generate exterior constraints. For example, a positive monotonic influence between two variables could be: "people who *smoke* have a higher chance of *lung cancer*". If a BN is fully constrained by such monotonic influences, this BN is referred to as a qualitative probabilistic network (QPN). If a BN is partially constrained by such monotonic influences (which means there also exists unconstrained parameters), this BN is referred to as a semi-qualitative probabilistic networks (SQPN). The learning and inference in these models has been studied in previous works (Renooij and van der Gaag, 2002; Renooij et al., 2002; de Campos and Cozman, 2005).

Of course, constraints can be generated from these monotonic influences, and also can be applied to improve the learning accuracy (Altendorf et al., 2005; Feelders and van der Gaag, 2006; van der Gaag et al., 2006). Thus, experts are only required to identify which edge in the BN has such a property. More recent discussions of such exterior constraints can be found in (van der Gaag et al., 2009; Liao and Ji, 2009; Yang and Natarajan, 2013; Zhou et al., 2014b). An empirical analysis of parameter learning with these constraints can be found in our own work (Zhou and Fenton, 2015) which is the subject of Chapter 4.

Next we outline the CO formulation for parameter learning. Based on the previous definition, a convex parameter constraint can be defined as $f(\theta_{ijk}) \leq \mu_{ijk}$, where $f : \Omega_{\theta_{ijk}} \rightarrow R$ is a convex function over θ_{ijk}. Regarding parameter constraints, the scores are computed using a constrained optimization problem, i.e. maximize the score function subject to simplex equality constraints and all parameter constraints defined by the user.

$$\begin{aligned} &\arg\max_{\theta}\ \ell(\theta, D) \\ &s.t.\ \forall_{i,j,k}\, g(\theta_{ijk}) = 0 \\ &\quad \forall_{i,j,k}\, f(\theta_{ijk}) \leq \mu_{ijk} \end{aligned} \tag{2.14}$$

where $g(\theta_{ijk}) = -1 + \sum_{k=1}^{r_i} \theta_{ijk}$ imposes that distributions defined for each variable

given a parent configuration sum to one over all variable states.

To allow constraint violations, a penalty term is introduced in the objective function. Following equation (2.14), suppose $f(\theta_{ijk}) = \theta_{ijk}$, then the penalty term is defined as

$penalty(\theta_{ijk}) = \left[\mu_{ijk} - \theta_{ijk}\right]^-$, where $[x]^- = \max(0, -x)$.

The strength of the penalty term is controlled by its weight. Moreover, to model uncertainty of qualitative constraints, Liao and Ji (2009) introduced an extra confidence level λ_{ijk} term. Therefore, the equation (2.14) can be rewritten as follows:

$$\begin{gathered} \arg\max_{\theta} \left(\ell(\theta, D) - \frac{w}{2} \sum_{ijk} \lambda_{ijk} penalty(\theta_{ijk})^2\right) \\ s.t.\ \forall_{i,j,k}\, g(\theta_{ijk}) = 0 \end{gathered} \tag{2.15}$$

where w is the penalty weight, which is chosen empirically. Obviously, the overall penalty varies with the confidence level for each constraint λ_{ijk}.

As discussed in previous work (Liao and Ji, 2009), $\Delta_{\theta_{ijk}} \ell(\theta, D)$ can be derived as: $\Delta_{\theta_{ijk}} \ell(\theta, D) = \sum_l \frac{p(X_i=k, \pi_i=j|d_l, \theta)}{\theta_{ijk}}$.

Because θ_{ijk} satisfies the constraint $\sum_{k=1}^{r_i} \theta_{ijk} = 1$, which can be eliminated by reparameterizing θ_{ijk}, so that the introduced new parameters β_{ijk} automatically respect the constraint on θ_{ijk} no matter what their values are. The new parameter satisfies the equation:

$$\theta_{ijk} = \frac{e^{\beta_{ijk}}}{\sum_{k=1}^{r_i} e^{\beta_{ijk}}} \tag{2.16}$$

Therefore, based on the chain rule,

$$\begin{aligned} \Delta_{\beta_{ijk}} \ell(\theta, D) &= \frac{\partial \ell(\theta, D)}{\partial \theta_{ijk}} \frac{\partial \theta_{ijk}}{\partial \beta_{ijk}} = \Delta_{\theta_{ijk}} \ell(\theta, D)(\theta_{ijk} - (\theta_{ijk})^2) \\ &= \sum_l p(X_i = k, \pi_i = j|d_l, \theta)(1 - \theta_{ijk}) \end{aligned} \tag{2.17}$$

Similarly, for $penalty(\theta_{ijk})$, the derivative is as follows:

$$\Delta_{\beta_{ijk}} penalty(\theta_{ijk}) = \begin{cases} (\theta_{ijk})^2 - \theta_{ijk} & if\ \theta_{ijk} \leq \mu_{ijk} \\ 0 & otherwise \end{cases} \tag{2.18}$$

Therefore, a gradient-based update approach can be applied to force the parameter estimation move towards the direction of reducing constraint violations. The detail of

INPUT : Data D
OUTPUT: Set of parameters θ

Randomly generate a set of parameters θ^t, and map θ^t to β^t (equation (2.16))
repeat
  $\Delta_{\theta^t} = 0$;
  for *each node* $i = 1$ **to** n **do**
    for *each parent configuration* $j = 1$ **to** $|\pi_i|$ **do**
      for *each state* $k = 1$ **to** r_i **do**
        for *data instance* $d_l = 1$ **to** D **do**
          $\Delta_{\theta^{t+1}_{ijk}} = \Delta_{\theta^t_{ijk}} + p(X_i = k, \pi_i = j | d_l, \theta^t)$;
        end
        if $\theta_{ijk} \leq \mu_{ijk}$ **then**
          $\Delta_{\beta^{t+1}_{ijk}} = \Delta_{\theta^{t+1}_{ijk}}(1 - \theta^t_{ijk}) + (\theta^t_{ijk})^2 - \theta^t_{ijk}$;
        else
          $\Delta_{\beta^{t+1}_{ijk}} = \Delta_{\theta^{t+1}_{ijk}}(1 - \theta^t_{ijk})$;
        end
        $\beta^{t+1}_{ijk} = \beta^t_{ijk} + \Delta_{\beta^{t+1}_{ijk}}$;
      end
    end
  end
until $\Delta_\beta \leq 1 \times 10^{-4}$;
for *each node* $i = 1$ **to** n **do**
  for *each parent configuration* $j = 1$ **to** $|\pi_i|$ **do**
    for *each state* $k = 1$ **to** r_i **do**
      $\theta^{t+1}_{ijk} = \frac{e^{\beta^{t+1}_{ijk}}}{\sum_{k=1}^{r_i} e^{\beta^{t+1}_{ijk}}}$;
    end
  end
end
return $\theta = \left\{ \theta^{t+1}_{ijk} \right\}$

Algorithm 2.1: Constrained parameter learning algorithm (CO)

the constrained parameter learning algorithm is shown in Algorithm 2.1. To ensure the solution is the global maximum, the objective function must be convex, which limits the usage of constraints. Meanwhile, because the starting points are randomly generated in gradient descent, this may cause unacceptably poor parameter estimation results when learning with zero or limited data counts N_{ijk}.

2.3.4 Multi–task or Parameter Transfer Learning

Previously, we have discussed BN parameter learning within a single problem domain. However, in many real-world applications, there may exist more than one problem domain. For example, suppose a specific medical diagnosis BN has been developed using minimal data from, say, one small country. Then it may be possible to exploit data from a similar medical diagnosis BN from a different country. Thus, multi-task or transfer learning algorithms are needed to exploit related domain knowledge to help learn the target domain with scarce data. The key difference between multi-task and transfer is the final learnt domains. In the multi-task setting, the learning accuracies are expected to be jointly improved by exploiting the shared information across all domains. In the transfer learning setting, only the learning accuracy of the target domain (domain of interest) is expected to be improved by exploiting other correlated domains (source domains). In this thesis, we are more interested in the transfer learning setting of BN parameters.

Transfer learning in general is now a well-studied area, with a good survey provided by (Torrey and Shavlik, 2009; Pan and Yang, 2010). Extensive work has been done on transfer and domain adaptation for machine learning models, including unsupervised transfer and analysis of relatedness (Duan et al., 2009; Seah et al., 2013b,a; Eaton et al., 2008). However, these studies have generally not addressed one or more of the important conditions that arise in the BN context addressed here, notably: transfer with heterogeneous state space, piece-wise transfer from multiple sources (a different subset of variables/dimensions in each source may be relevant), and scarce *unlabeled* target data (thus precluding conventional strategies that assume ample unlabeled target data, such as MMD (Huang et al., 2007; Seah et al., 2013b)).

In comparison to the single domain BN learning, few works have been designed for the BN transfer learning. Niculescu-mizil and Caruana (2007) developed the first multi-task framework of BN structure transfer. However, it assumes that all sources are equally related and simply learns the parameters for each task independently. The transfer framework of (Luis et al., 2010) proposes a K-L divergence method to measure the relatedness of tasks, and employs the heuristic weighted sum model for fusing target and selected source parameters. The weight w represents how much the target parameters will be used in the fusion. This weight is only proportional to the number

of target training samples and the number of all state configurations $|\pi_i|$ for the parents of X_i:

$$w = \begin{cases} 1 - \frac{\log \frac{|D|}{2|\pi_i|}}{\frac{|D|}{2|\pi_i|}} & if \ \frac{|D|}{2|\pi_i|} \geq 3 \\ 1 - \frac{\frac{|D|}{2|\pi_i|} \log 3}{3} & if \ \frac{|D|}{2|\pi_i|} < 3 \end{cases} \tag{2.19}$$

As we can see, this method does not take the sources into consideration and does not systematically address when, from where, and how much to transfer (indeed we will show in Chapter 5 this method significantly underperforms ours). Finally, the study (Oyen and Lane, 2012) considers multi-task structure learning, again with independently learned parameters. They investigate network/task-level relatedness, showing transfer performs poorly without knowledge of relatedness. However, they address this by manually specified relatedness. Finally, a recent study (Oates et al., 2014) improves this by automatically inferring the network/task-level relatedness. However, they do not consider information sharing of parameters. A related area to BN transfer is transfer in Markov Logic Networks (MLNs) (Mihalkova et al., 2007; Davis and Domingos, 2009; Mihalkova and Mooney, 2009).

This thesis proposes the first BN parameter transfer learning (BNPTL) algorithm to reason about both network and fragment relatedness. In contrast to these prior studies, the proposed approach has the following benefits: it can employ multiple fragments from different source networks to help the transfer; it automatically quantifies source relevance and is robust to some or all irrelevant sources (rather than assuming a single relevant source). The detail of this method will be discussed in Chapter 5.

2.4 Estimating Learning Performance

Previous sections discussed how to perform parameter estimation in BNs. However, we provided no way of assessing the quality of the learnt parameters. This section describes ways to estimate the performance of a BN.

Let $\hat{\theta}_{ij}$ and θ_{ij} represent the learnt parameter distribution and true parameter distribution respectively. The standard way to measure the distance between two distributions over the same values is to compute their K-L divergence:

$$KL(\theta_{ij}, \hat{\theta}_{ij}) = \sum_{k=1}^{r_i} \theta_{ijk} \log \frac{\theta_{ijk}}{\hat{\theta}_{ij}} = H(\theta_{ij}, \hat{\theta}_{ij}) - H(\theta_{ij}) \tag{2.20}$$

here $H(\theta_{ij})$ stands for the entropy of a probability distribution θ_{ij}, and $H(\theta_{ij}, \hat{\theta}_{ij})$ stands for the cross-entropy between two probability distributions θ_{ij} and $\hat{\theta}_{ij}$ over the same values. The smaller the K-L divergence is, the closer the estimated parameters $\hat{\theta}_{ij}$ to the true parameters θ_{ij}.

Because the standard K-L divergence is not symmetric, i.e.,

$$KL(\theta_{ij}, \hat{\theta}_{ij}) \neq KL(\hat{\theta}_{ij}, \theta_{ij}).$$

we will use the symmetric extension of K-L divergence in the thesis, which is also referred to as Jensen-Shannon divergence (JSD):

$$KL^s(\theta_{ij}, \hat{\theta}_{ij}) = \frac{KL(\theta_{ij}, \hat{\theta}_{ij}) + KL(\hat{\theta}_{ij}, \theta_{ij})}{2} \tag{2.21}$$

Using K-L divergence to compute the performance of our learnt distribution assumes we can access the true distribution of the data. In the real-world applications, such underlying distribution is not provided. In this case, we convert it into a forecast/classification problem (set evidence for random variables X_j $(X_j \in V, j \neq i)$ based on the existing learnt model to predict the posterior of interested variable X_i in different real-world applications), and chose the state/value with the maximal predicted probability. If the predicted value is equal to the original value in dataset, then this prediction is correct.

In the machine learning community, the most widely used classification performance measurement is the Area Under the Receiver Operating Characteristic Curve (ROC-AUC) (Bradley, 1997). Considering the simplest binary classification of X_i, if we are provided with the number of truly positive and negative values of X_i in the N testing data instances – C_p and C_n. After the BN inference, we have the number of predicted positive and negative values of X_i – R_p and R_n. Thus

$N = C_p + C_n$ and $N = R_p + R_n$.

Let T_p and T_n represent the number of true positives and true negatives respectively. Then we can have the following meaningful measurement about the classifier's performance:

$$\begin{aligned} Accuracy &= \frac{T_p+T_n}{C_p+C_n} \\ Sensitivity &= \frac{T_p}{C_p} \\ Specificity &= \frac{T_n}{C_n} \\ Positive\ predictive\ value &= \frac{T_p}{R_p} \\ Negative\ predictive\ value &= \frac{T_n}{R_n} \end{aligned} \tag{2.22}$$

All of these measurements are valid only for one particular operating point, which may introduce ambiguity. Thus, there is a need for a measurement that is invariant to the decision criterion selected, which is abbreviated AUC. Hand and Till (2001) presented a simple approach to calculating the AUC of a given classifier.

$$AUC = \frac{2S_p - C_p(C_p+1)}{2C_pC_n} \tag{2.23}$$

Here $S_p = \sum ra_i$, where ra_i is the rank of i-th positive values in the ranked list. The ranked list is ordered by the posterior values of X_i, where the smallest posterior value corresponds to the lowest rank.

2.5 The Research Gap

In this section we focus on the limitations of previous algorithms for BN parameter learning with scarce data and identify a set of key properties that need to be addressed by new or updated algorithms for BN parameter estimation. These properties motivate the proposal of MPL-C and MPL-EC in Chapter 3 and 4, BNPTL in Chapter 5 and MPL-TC in Chapter 6, respectively.

2.5.1 Parameter Learning with Constraints

The constrained convex optimization (CO) parameter learning method is attractive because of its efficiency; it applies fast iterative algorithms and provides globally optimal solutions for several important types of parameter constraints. However, when the constraints are nonlinear and non-convex, the best estimation of constrained parameter might not be found in polynomial time.

In addition, this method cannot handle the zero data situation and its parameter estimation performance is highly dependent on the choice of penalty function and weights of soft constraints, which need to be fine tuned. For example, in estimating the

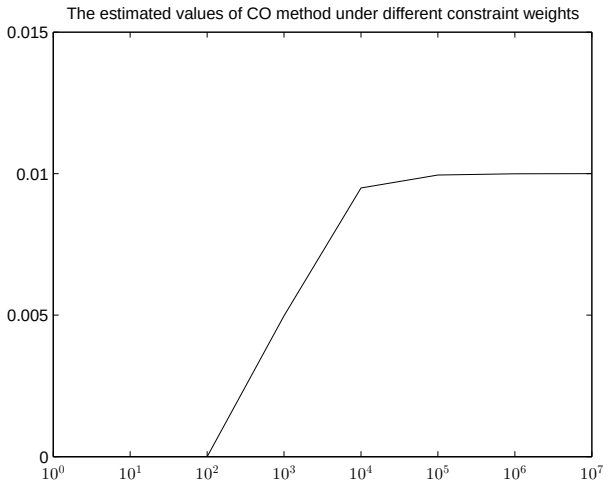


Figure 2.4: The estimated values of CO method under different constraint weights range from 10^0 to 10^7.

probability of smoker getting cancer, suppose there is a constraint

"$p(cancer = true|smoker = true) > 0.01$"

and 0 out of 10 data records is observed with cancer symptom. Intuitively, the MLE result for this parameter is 0 with equation (2.8), which conflicts with the domain constraints.

Given the equation (2.15), the CO will produce different estimation results under different constraint weight settings. This problem is illustrated by Figure 2.4.

As we can see from Figure 2.4, the CO can return a non-zero result for the parameter of interest, which improves the final learning results. However, this result is sensitive to the choice of the constraint weight. Thus, extra effort is needed to find the appropriate weight.

2.5.2 Parameter Transfer Learning

Previous transfer learning algorithms use the K-L divergence to measure the relatedness between different learning tasks. They transfer the closest whole source BN to the target. This transfer is based on the assumption that the node order is the same in both target and source BNs, which means the correspondence is known before the transfer (Figure 2.5).

The transfer with known node correspondence is a strong assumption, which could greatly reduce the computation cost. However, this setting limits its usage in real-world applications, where the nodes in the source BN may not follow the target node order

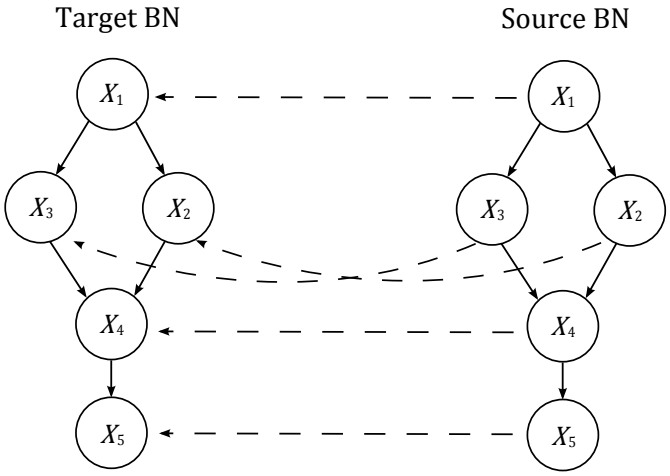


Figure 2.5: The conventional BN transfer paradigm with known node correspondence. The dash arrows represent the transfer from the node in source BN to its corresponding node in target BN.

(e.g. the nodes X_2 and X_3 in Figure 2.5 source BN can swap their positions). Moreover, this known node correspondence assumption does not allow the piece-wise transfer, which supports the transferred source nodes coming from different source BNs.

After finding the right transfer correspondence, another challenge in transfer learning is to combine the target and selected source parameters. Luis et al. (2010) discuss several cases of transforming the source substructure to consist with a target substructure, i.e., add/delete parents in the source substructure. However, after that, they employ a heuristic weighted sum model for aggregating target and selected source parameters, the weight is only proportional to the number of training samples and the number of parameters in an NPT (as shown in equation (2.19)).

We continue with the example in Section 2.5.1 ("smoker"→"cancer"), where all the nodes are binary, and none of the 10 smokers is observed with cancer symptom. Therefore, the $|D| = 10$, $|\pi_i| = 2$, and MLE of target parameter is:

$\theta_i^{target} = p(cancer = true|smoker = true) = 0$

If the transferred source parameter is θ_i^{source}, the final parameter $\hat{\theta}_i$ is estimated as below:

$$\hat{\theta}_i = w\theta_i^{target} + (1-w)\theta_i^{source}$$

According to the definition of equation (2.19), the weight w is calculated without any knowledge from the sources: $w = 1 - \frac{\frac{|D|}{2|\pi_i|}\log 3}{3} = 0.602$.

2.5.3 Identification of Key Properties

The limitations discussed above can be translated into a set of requirements that should be fulfilled by an algorithm for BN parameter learning in a real-world scarce data situation. Specifically, we identify the following key properties:

- **Multiple data sparsity:** the algorithm should be robust to different amount of data, especially the scarce data situation.
- **Parameter prior:** the algorithm should use the parameter prior (either informative or non-informative) in the parameter estimation with constraints.
- **Multiple types of constraints:** the algorithm should support a wide range of parameter constraints, i.e., convex or non-convex constraints, and constraints within or across NPT columns.
- **Robust parameter transfer:** the algorithm should support the piece-wise transfer, and robustly find the right source parameter without known node correspondence, and fuse it to the target.
- **Generalization:** the algorithm should support the parameter learning with both target parameter constraints and source data samples.

Considering previous work, we argue that there is no single algorithm for BN parameter learning that fulfils all of the above requirements. Hence, there is a clear research gap, which we aim to bridge in the following chapters.

Chapter 3

Parameter Learning with Constraints

In this chapter, we present a multinomial parameter learning method, which can easily incorporate both expert judgments and data during the parameter learning process. This method uses an auxiliary BN model to learn the parameters of a given BN. The auxiliary BN contains continuous variables that model data statistics and probability distributions of parameters. The auxiliary BN is updated by an iterative discretization technique. The expert judgments are provided in the form of constraints on parameters divided into two categories: linear inequality constraints and approximate equality constraints. The method is evaluated with experiments based on a number of well-known sample BN models as well as a real-world software defects prediction BN model.

3.1 Multinomial Parameter Learning

In this thesis, we mainly focus on learning NPTs for nodes with a finite set of discrete states. For a node with r_i states and no parents, its NPT is a single column whose r_i cells correspond to the unconditioned probabilities of r_i states. Hence, each NPT entry can be viewed as a parameter representing a probability value of a discrete distribution. For a node with parents, the NPT will have $|\pi_i|$ columns corresponding to each of the π_i instantiations of the parent node states. Hence, such an NPT will have $|\pi_i|$ different r_i-value parameter probability distributions to define or learn. Each such r_i-value parameter probability distribution can be learnt via the auxiliary Multinomial Parameter Learning model (MPL) (Ogunsanya, 2012). This method models the *Multinomial* distribution, which is a generalization of the *Binomial* distribution, and gives the probability of each combination of outcomes in a sequence of independent trials of an r_i-outcome process.

3.1.1 Model Construction

We use the estimation of probability distribution $p(X_i|\pi_i = j)$ (i.e., the j-th column of the NPT associated with the variable X_i) to illustrate the construction of the MPL model. Suppose there are r_i states, (i.e., r_i cell entries), and the goal is to learn the r_i probability parameters $\theta_{ij1},...,\theta_{ijr_i}$ corresponding to these states. Assume we have N_{ijk} data observations of the k-th state ($1 \leq k \leq r_i$) and the total number of observations is $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. Then we can create a multinomial parameter learning BN model for each NPT column (shown schematically in Figure 3.1) to estimate parameters $\theta_{ij1},...,\theta_{ijr_i}$.

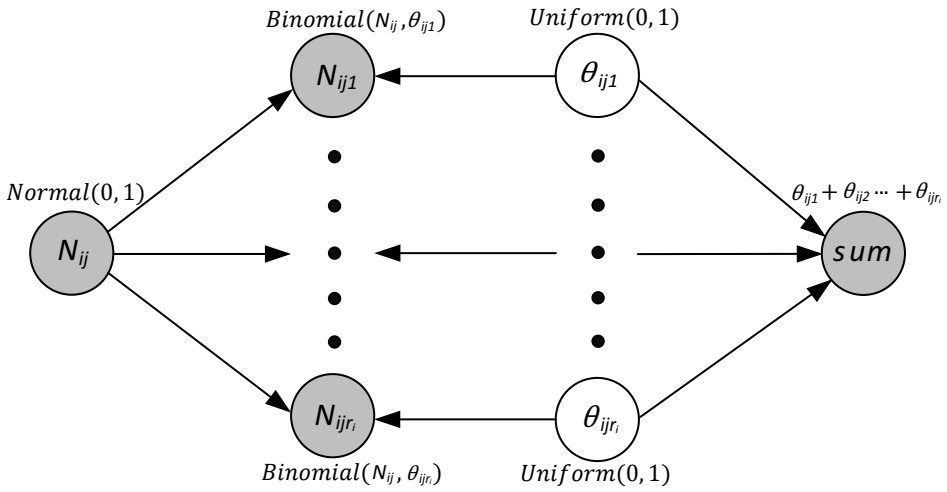


Figure 3.1: Graphical model representation for the MPL model. The associated prior probability distributions for each node are shown on their sides: $p(N_{ij}) = Normal(0,1)$, $p(N_{ijk}) = Binomial(N,\theta_{ijk})$, $p(\theta_{ijk}) = Uniform(0,1)$, and $p(sum) = \sum_{k=1}^{r_i}\theta_{ijk}$. The grey nodes are observed during the inference.

Specifically, we start by creating an integer node named N_{ijk} (corresponding to N_{ijk} as defined above) for each k that is *Binomial* distributed. This node has two parents N_{ij} and θ_{ijk} to model the total number of trials and success probability in the *Binomial* distribution. The N_{ij} has a *Normal* distribution, which provides an infinite range for the total number of trials. The prior distribution of each θ_{ijk} is uniform between 0 and 1. Finally, there is an integer node *sum*, which is a shared child of θ_{ijk} ($k = 1,...,r_i$). This node models the normalization constraint for the success probabilities, i.e., that they should sum to 1.

3.2 Incorporating Expert Judgments

With the exception of NPTs that involve logical certainty[1] (i.e., where cell entries must be either 0 or 1 as would be the case if the node represented "A OR B" for parent nodes A, B), it is known that experts find it extremely difficult to provide accurate and consistent exact values for NPTs (Fenton et al., 2007). They are more reliable when providing qualitative judgments like constraints (Feelders and van der Gaag, 2006).

3.2.1 Commonly Used Constraints

Previous work (Niculescu et al., 2006) has discussed a wide variety of parameter constraints elicited from expert judgments. However, in real-world applications, most expert judgments can be described with two important types of parameter constraints: linear inequality constraints and approximate equality constraints. Since in this chapter we focus purely on constraints that can be made about the parameters $\theta_{ij1},...,\theta_{ijr_i}$ of a single NPT column, these constraints can be defined formally as follows:

Definition 3.2.1. A ***linear inequality constraint*** is an expression about parameters constrained by a linear function, which has the following format:

$$\beta_0 + \sum_{k=1}^{r_i} \beta_k \theta_{ijk} \leq 0 \tag{3.1}$$

where the coefficients $\beta_0, \beta_k\,(1 \leq k \leq r_i)$ are real numbers.

Linear constraints are simple and can easily be formalized from expert judgments. For example, in the computer vision domain, recognizing facial expressions from a library of several thousand facial images, can be modelled by a BN where each node encodes a facial expression Action Unit (AU). Some intuitive judgments are easy to elicit, i.e., the AU "Mouth stretch" is not usual, therefore $p(Mouth\_stretch = true) \leq 0.5$ (de Campos et al., 2009b). In real-world applications, such constraints are common, especially between two parameters, or between a parameter and a constant. Most of them have one of the following even simpler formats:

$$\theta_{ijk} \geq \theta_{ijk'}$$

[1] Where an NPT value genuinely is 0 or 1 because of logical certainty, no amount of data will 'learn' these values. If the expert can identify such entries then it is assumed they are 'facts' and are not incorporated in the learning process.

$$\theta_{ijk} \geq \beta\,\theta_{ijk'}$$

$$\theta_{ijk} \geq \beta_0$$

where $\beta \in \boldsymbol{R}$, $k \neq k'$, and $\beta_0 \in [0,1]$.

Definition 3.2.2. An ***approximate equality constraint*** (represented by the symbol $\approx$) is an assertion that one parameter value is similar to another value.

In real-world applications, such constraints correspond to one of the following formats:

$$\theta_{ijk} \approx \theta_{ijk'}$$

$$\theta_{ijk} \approx \beta\,\theta_{ijk'}$$

$$\theta_{ijk} \approx \beta_0$$

where $\beta \in \boldsymbol{R}$, $k \neq k'$, and $\beta_0 \in [0,1]$.

In each case, $\theta_{ijk} \approx \theta_{ijk'}$ assumes there is some ε $(0 < \varepsilon < 1)$ for which

$$|\theta_{ijk} - \theta_{ijk'}| \leq \varepsilon \tag{3.2}$$

Note that (with the exception of 0 and 1 probabilities in the case of logical certainty as discussed above) the approximation (as opposed to equality) is generally needed not just because it matches the way experts think, but it also avoids the problem whereby exact continuous values have zero probability. Therefore, instead of specifying that θ_{ijk} is exactly equal to $\theta_{ijk'}$, the expert selects an appropriate (small) positive value ε such that $\theta_{ijk} \approx \theta_{ijk'}$ as is captured in the equation (3.2).

3.2.2 Constraints Elicitation

Eliciting constraints from experts is clearly infeasible if it has to be done for every single parameter. Therefore, people usually investigate constraints for a subset of BN parameters. Research in finding the best subset of parameters falls into the scope of active learning (Tong and Koller, 2001a), where the best eliciting scheme is investigated via maximal information gain. In this thesis, we do not focus on improving such elicitation processes. However, as a rule of thumb, when experts are able to provide

Table 3.1: The types of BN nodes for domain experts to focus their attention when providing the judgments.

Type	Feature
Logical connective or conditionally deterministic nodes	Nodes that represent logical expressions (like OR, AND, and NOR) or conditionally deterministic functions (arithmetical) of the parents.
'Confidence' nodes	Nodes for which experts are confident to provide constraints. For example, nodes for which they know certain conditional probability values are very low or very high.
'Known empirical' nodes	Nodes for which extensive data is available. Usually root nodes.

Table 3.2: Details of 9 commonly elicited verbal expressions and their associated approximate equality constraints (Mosteller et al., 1990).

Index	Verbal Expression	Approximate Equality Constraints
1	Very high probability	$\theta_{ijk} \approx 0.93\,(\varepsilon = 0.03)$
2	High probability	$\theta_{ijk} \approx 0.82\,(\varepsilon = 0.05)$
3	Moderate probability	$\theta_{ijk} \approx 0.52\,(\varepsilon = 0.09)$
4	Very likely	$\theta_{ijk} \approx 0.90\,(\varepsilon = 0.05)$
5	Likely	$\theta_{ijk} \approx 0.71\,(\varepsilon = 0.08)$
6	Unlikely	$\theta_{ijk} \approx 0.17\,(\varepsilon = 0.07)$
7	Very unlikely	$\theta_{ijk} \approx 0.03\,(\varepsilon = 0.04)$
8	Rarely	$\theta_{ijk} \approx 0.07\,(\varepsilon = 0.03)$
9	Very rarely	$\theta_{ijk} \approx 0.03\,(\varepsilon = 0.02)$

judgments about node parameters they should focus on and distinguish between the three types of nodes as shown in Table 3.1.

After finding the right parameter, experts are required to provide parameter constraints, e.g. "the probability of people getting cancer is smaller than 0.01". However, sometimes experts may provide verbal descriptions about such constraints, e.g. "the probability of people getting cancer is *very rare*". In this situation, the challenge is to understand the quantitative meanings of such verbal descriptions, and eliciting associate numerical parameter constraints from them. This challenge has been well addressed in previous works called 'quantifying probabilistic expressions' (Druzdzel, 1989; Mosteller et al., 1990), where empirical studies (survey research conducted via mail questionnaire) had been done to show the variability of numerical estimations for different verbal expressions. For example, according to previous results in (Mosteller

et al., 1990), the verbal expression 'very rarely' corresponds to a probability between 0.01 and 0.05. Such uncertainty can be modelled by approximate equality constraints discussed in last section, e.g. "the probability of people getting cancer is approximately equal to 0.03 with $\varepsilon = 0.02$". We list the commonly elicited verbal expressions and their associated approximate equality constraints in Table 3.2.

3.3 Multinomial Parameter Learning Model with Constraints

Given that an expert has identified a number of constraints as defined above within a NPT column, then these constraints can be integrated as additional observed constraint nodes within the MPL model to generate a new model called Multinomial Parameter Learning model with Constraints (MPL-C) as shown in Figure 3.2.

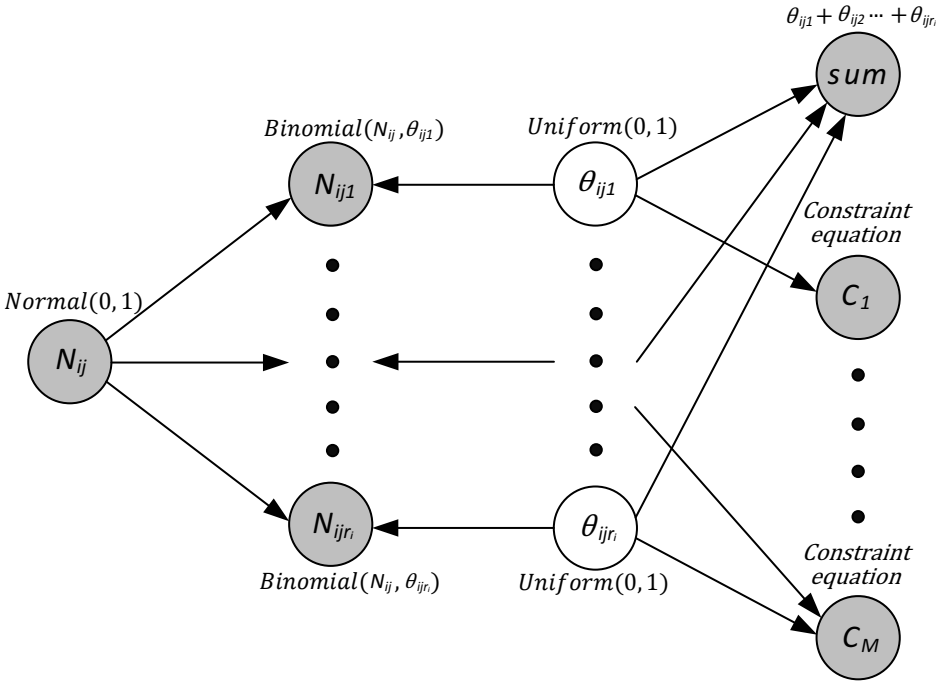


Figure 3.2: Graphical model representation for the MPL-C model with M constraints. For the node with constraints, their equations follow the representations in equation (3.1) and (3.2). The grey nodes are observed during the inference.

Here, each constraint node C_m is a deterministic binary (*true/false*) node with expressions that specify[2] the constraint relationships between its parents:

$$if(\beta_0 + \sum_{k=1}^{r_i} \beta_k \theta_{ijk} \leq 0, true, false)$$

[2]Using the syntax of *Conditional* expressions in AgenaRisk.

$$if(abs(\theta_{ijk} - \theta_{ijk'}) \leq \varepsilon, true, false)$$

When the constraint is between a single parameter and a constant (i.e., $\beta_0 + \beta_k \theta_{ijk} \leq 0$), the constraint node will only have a single parent. The following is a very simple example to demonstrate how to create the MPL-C model.

Example 3.3.1. *Suppose we have a single node F having just two states a and b (So $r_i = 2$). There are two probability parameters θ_1 and θ_2 corresponding to $p(F = a)$ and $p(F = b)$ respectively. The expert judgment consists of a single inequality constraint C_1 namely $\theta_1 \geq \theta_2$. The MPL-C model for this simple example is shown in Figure 3.3.*

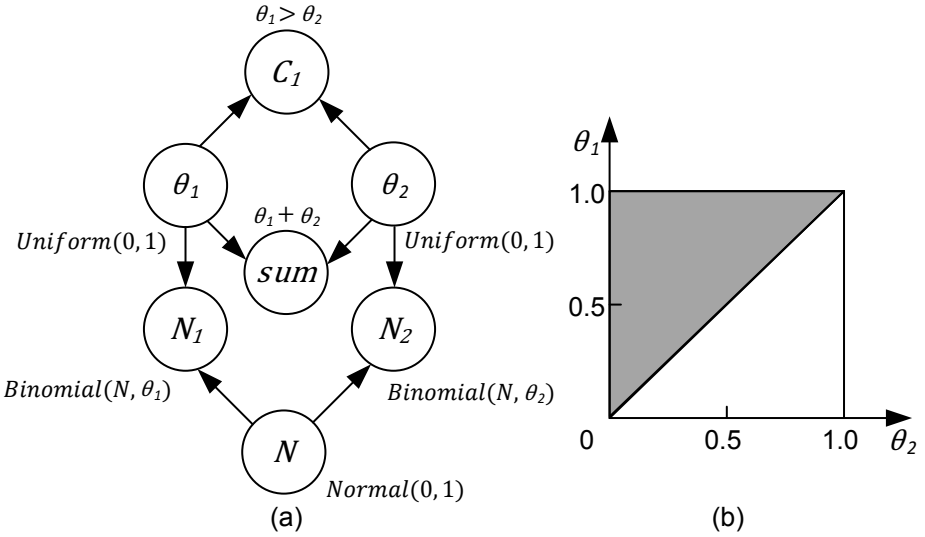


Figure 3.3: An illustrated MPL-C model for a 2-parameter learning problem: (a) The graphical representation of the MPL-C model. (b) The grey area represents the constraining parameter space.

After observing the data statistics $(N_{ij}, N_{ij1}, ..., N_{ijr_i})$ and available constraints (the constraint nodes $C_1, ..., C_M$ are all observed with *true* values), we can update these auxiliary models to get the parameter posteriors:

$$p(\hat{\theta}_{ij1}, ..., \hat{\theta}_{ijr_t} | N_{ij}, N_{ij1}, ..., N_{ijr_i}, C_1, ..., C_M, sum)$$

Although the MPL-C model is a conceptually simple BN, it has certain features that make accurate and efficient computation very challenging. First note that it contains both discrete and continuous nodes, i.e., it is a *hybrid* BN (Murphy, 1998). There are two well-known problems when dealing with continuous nodes in BNs. On the one hand if the underlying variable is *non-Normally* distributed (and here we certainly cannot assume *Normality*) then there is no known way of performing exact probabilistic

inference. As explained in Chapter 2, we need to use the DDJT inference algorithm to handle this kind of model. In the next section we explain the details of the inference algorithm.

3.4 Inference with Constraints

Inference in DDJT[3] refers to the process of computing the discretized posterior marginals of each of the unknown nodes (these are the nodes without evidence). Each of these nodes is continuous. For any such node suppose the range is Ω, and the probability density function (PDF) is f. The idea of discretization is to approximate f by, first, partitioning Ω into a set of intervals $\psi = \{\omega_i\}$ and, second, defining a locally constant function $f^{\frown}$ on the partitioning intervals.

The dynamic discretization approach involves searching Ω for the most accurate specification of the high-density regions, given the model and the evidence, calculating a sequence of discretization intervals in Ω iteratively. At each stage in the iterative process a candidate discretization, ψ, is tested to determine whether the resulting discretized probability density $f^{\frown}$ has converged to the true probability density f within an acceptable degree of precision.

At convergence or when the bound of maximal number of iterations is reached (the default is 25), f is then approximated by $f^{\frown}$. Here we consider two stopping convergence criteria: 1) stable entropy error and 2) low entropy error (Line 8, Algorithm 3.1). Finally, the mean value of $f^{\frown}$ will be assigned as the parameter estimation (i.e., the corresponding NPT cell value). Full details can be found in Algorithm 3.1.

Now we describe how we have adapted the Algorithm 3.1 to perform inference in the simple MPL-C model introduced in Example 3.3.1 to learn the parameters θ_1 and θ_2.

First consider the model in Example 3.3.1 but without the constraint. So there are just 2 parameters with no data observations or constraints. The auxiliary BN will be a 5-node MPL model without constraints (i.e., in this case we are using just MPL and not MPL-C). Because the priors of the nodes θ_1 and θ_2 are *uniform*, the perfect parameter estimation should be (0.5, 0.5); this estimation also can be achieved by MAP. The

[3]The time complexity of inference in MPL-C is determined by the iterations of dynamic discretization I and the largest clique size S of the junction tree, which is $O(I \cdot e^{(S-1)})$.

INPUT : A set of random variables V
Observed variables E
Query variables Q
OUTPUT: Posteriors of parameters $\hat{p}(Q|E)$

Initialize the discretization θ to get $\psi_m^{(0)}$ for each continuous variable $X_m \in V$. Build a junction tree structure for the MPL-C model to determine the cliques, Φ, and sepsets.
for $l = 1$ **to** *max_num_iterations* **do**
  Compute the NPTs for parameters, $p^{(l)}(X_m|pa(X_m))$, on $\psi_m^{(l-1)}$ for all nodes $X_m \in Q$ (query variables) that have new discretization or that are children of parent nodes that have a new discretization.
  Initialise the junction tree by multiplying the NPTs for all nodes into the relevant members of Φ. Enter evidence, $X_n = e_n$, $X_n \in E$ (observed variables), into the junction tree. Perform global propagation on the junction tree.
  for *all nodes* $X_m \in Q$ **do**
    Marginalize/normalise to get the discretized posterior marginals $p^{(l)}(X_m|X_n = e_n)$.
    Compute the approximate relative entropy error $S_{X_m}^{(l)} = \sum_{\omega_{mn}} Err_{mn}$, for $p^{(l)}(X_m|X_n = e_n)$ over all intervals ω_{mn} in $\psi_m^{(l-1)}$.
    if $(1-\varepsilon \leq S_{X_m}^{(l-c)}/S_{X_m}^{(l-c+1)} \leq 1+\varepsilon \; for \; c = 1,2,3)$ *or* $(S_{X_m}^{(l)} < \sigma)$ *where the* $\varepsilon, \sigma = 1 \times 10^{-4}$.
    then
      Stop discretization for node X_m, $\hat{p}(X_m|E) = p^{(l)}(X_m|X_n = e_n)$.
    else
      Create a new discretization $\psi_m^{(l)}$ for node X_m. Split it into two halves the interval ω_{mn} in $\psi_m^{(l-1)}$ with the highest entropy error Err_{mn}; Merge those consecutive intervals in $\psi_m^{(l-1)}$ with the lowest entropy error or that have zero mass and zero entropy error.
    end
  end
end
return $\hat{p}(Q|E) = \prod_{X_m \in Q} \hat{p}(X_m|E)$

Algorithm 3.1: Dynamic discretization junction tree algorithm (DDJT)

results of inference for node θ_1 via our method are listed in Table 3.3 (the value of θ_2 is, of course just $(1-\theta_1)$).

Using the default maximal number of iterations (25), the result of our algorithm has 0.0008 deviation from the 'correct' result. At 150 iterations there is no significant difference between these two results (at 4 decimal places).

After creating the MPL-C model, we perform moralization, triangulation and in-

Table 3.3: The evaluation of inference accuracy with different maximal number of iterations.

Maximal number of iterations	25	50	100	150
Probability values	0.5008	0.5003	0.5001	0.5000

tersection checking steps of the DDJT algorithm (3.4). Then, we set evidence into certain nodes as follows:

$C_m = true$ $(m = 1, ..., M)$,

N_{ijk} = actual number of observations of the k-th state in the data set,

$sum = 1$.

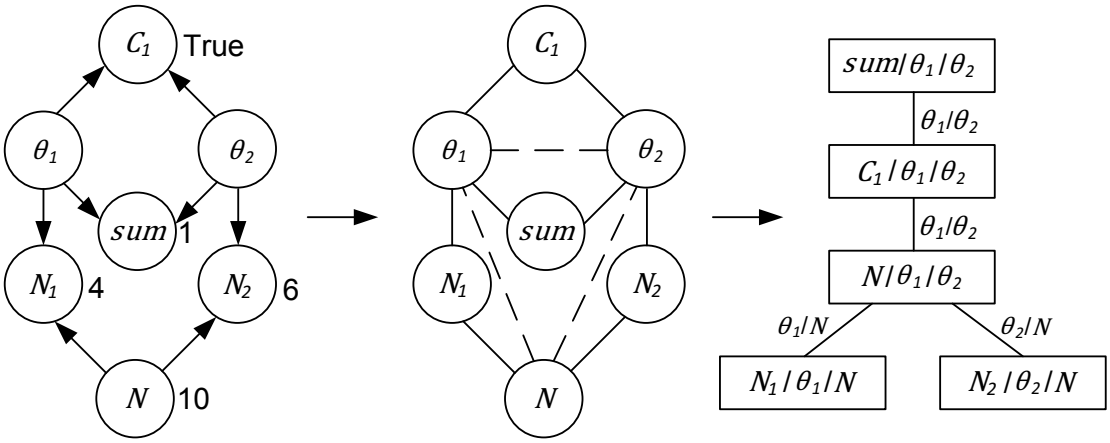


Figure 3.4: The moralization, triangulation, and intersection checking steps of the DDJT algorithm in a 2-parameter MPL-C model. The constraint is considered as evidence in this MPL-C model, as is the mutual consistency assumption (the sum of θ_1 and θ_2 is equal to one). Meanwhile, the observations of trials in the *Binomial* distributions are also regarded as evidence.

Next, we use another example (Example 3.4.1) to illustrate the evidence propagation in DDJT and present the final learning results of Example 3.3.1.

Example 3.4.1. *In this example, we use the same assumptions as in Example 3.3.1 that there is a single inequality constraint* C_1*:* $\theta_1 \geq \theta_2$ *and 10 data observations: 4 for state a and 6 for state b. Note that the relative empirical-frequencies of the parameters (MLE results) with the sample do not satisfy the constraint, so there is clear added value in being able to combine the expert judgment and data. Suppose that the "ground truth" of the discrete probability distribution is given as (0.58, 0.42), i.e.,* $\theta_1 = 0.58$ *and* $\theta_2 = 0.42$*. Figure 3.4 shows how the inference procedure works. The figure also*

shows the evidence assumptions in this case. Therefore, the problem is treated as using the junction tree algorithm to calculate the posterior probabilities of θ_1 and θ_2 given the evidence. In the DDJT algorithm, the posteriors of query nodes can be calculated given the initialized discretizations and evidence. After that, this algorithm continues to split those intervals with highest entropy error in each node until the model converges to an acceptable level of accuracy or reach the maximal number of iterations. The shadow area in Figure 3.3 represents the posterior ranges of θ_1 and θ_2. After inference, the mean values of these posteriors are finally used to fill the NPT column of target parameters. For this example, the final probabilities learned by MPL-C are (0.55, 0.45). So, in contrast to MLE, despite the observations which suggest θ_2 is greater than θ_1, the expert constraint has ensured that the final learned parameters are still quite close to the ground truth.

3.5 Experiments

The goal of the experiments is to evaluate the benefits of our method that uses expert judgment in parameter learning. We test the method against the pure machine learning techniques as well as against the competing method that incorporates expert judgment (i.e., the constraint optimization method CO) in 6 standard models from the BN repository[4]. Details and descriptions of these BNs can be found in Table 3.4. Section 3.5.1 focuses on parameter learning in the Asia BN with both real and synthetic expert judgment. Section 3.5.2 investigates the algorithm performance on the other 5 BNs with synthetic expert judgment.

In all cases, we assume that the structure of the model is known and that the 'true' NPTs that we are trying to learn are those that are provided as standard with the models. In each experiment we are given a number of sample observations which are randomly generated based on the true NPTs. The experiments consider a range of sample sizes. In all cases the resulting learnt NPTs are evaluated against the true NPTs by the average K-L divergence (equation (2.21)) between learnt and 'true' NPT columns[5]. The smaller the average K-L divergence is, the closer the estimated NPT is

[4] `http://www.bnlearn.com/bnrepository/`

[5] Usually the K-L divergence between the joint distributions should be used, however a good reconstructed model should close to the ground truth piece-by-piece. Thus the K-L divergence is locally measured for each NPT column, and we use the average K-L divergence in this thesis.

Table 3.4: Descriptions of Asia, Weather, Cancer, Insurance, Alarm and Hailfinder BNs

Name	Nodes	Arcs	Paras[†]	M-ind[‡]	Descriptions
Asia	8	8	18	2	Used for a patient entering a chest clinic to diagnose his/her most likely condition given the symptoms and risk factors (Lauritzen and Spiegelhalter, 1988).
Weather	4	4	9	2	Models the risk factors like rain and sprinkler, which can be affected by the weather condition and can all detemince the presence of wet grass (Russell and Norvig, 2009).
Cancer	5	5	10	2	Models the interaction between risk factors and symptoms for the purpose of diagnosing the most likely condition for a patient getting lung cancer (Korb and Nicholson, 2010).
Insurance	27	52	984	3	Used for estimating the expected claim costs for a car insurance policyholder (Binder et al., 1997).
Alarm	37	46	509	4	Is an acronym for "A Logical Alarm Reduction Mechanism". This network is a medical diagnostic application used to explore probabilistic reasoning techniques in belief networks, and is used for patient monitoring (Beinlich et al., 1989).
Hailfinder	56	66	2656	4	Is used for predicting risk of hails in northern Colorado (Abramson et al., 1996).

[†] Total number of parameters in each BN.
[‡] The maximum edge in-degree, the maximum number of node parents in each BN.

to the true NPT. If frequency estimated values are zero, they are replaced by a tiny real value (1×10^{-7}) to guarantee they can be computed by the K-L measure.

3.5.1 Asia BN Experiments

The Asia BN, which models the interaction between risk factors, diseases and symptoms for the purpose of diagnosing the most likely condition for a patient entering a chest clinic, is shown in Figure 3.5. All 8 nodes are binary so each NPT column has just 2 NPT entries to learn; since the values of 2 entries sum to 1, each column has only one independent parameter. Hence there are 18 independent parameters to learn in the model (1 each for nodes *VA* and *S*, 2 each for nodes *TB*, *LC*, *B*, *PX*, and 4 each for nodes *LCTB* and *D*). Here the data samples are generated from the 'true' NPTs specified in previous work (Lauritzen and Spiegelhalter, 1988). For example, the 'true' probability of "*VA*=*true*" is 0.01, so its data will be randomly sampled based on this probability.

Eliciting real expert judgments As discussed in Table 3.1, it is not hard to see that, the logical connective node "*LCTB*: Lung cancer or tuberculosis" gets the first priority

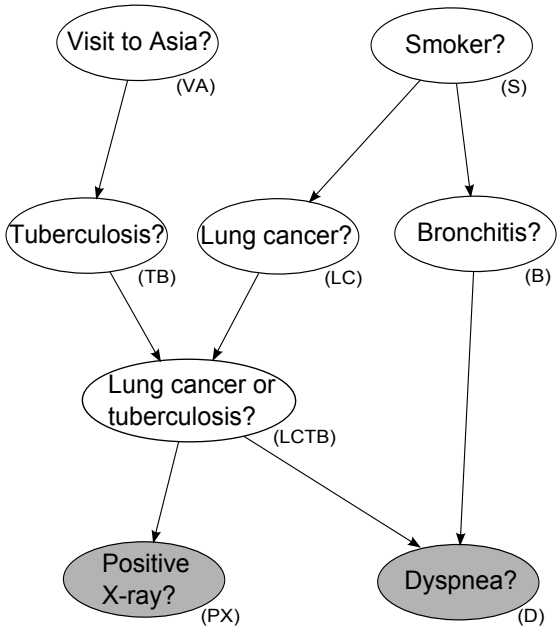
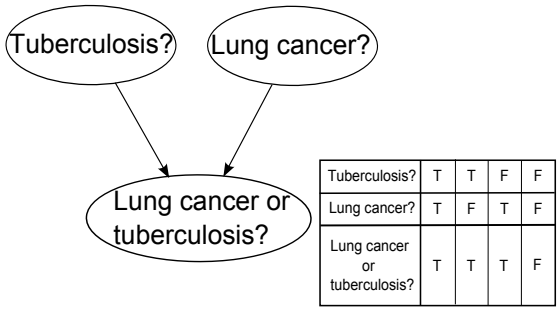


Figure 3.5: The directed acyclic graph of the Asia BN.

to be constrained by its logical output shown in Figure 3.6. This kind of constraint is absolutely certain, and the constraining NPTs are specified with point values. Therefore, the CO and MPL-C approach will directly use these values without learning.



Tuberculosis?	T	T	F	F
Lung cancer?	T	F	T	F
Lung cancer or tuberculosis?	T	T	T	F

Figure 3.6: The logical OR connective node and its output in Asia BN, which constrains the parameters $P(LCTB|TB,LC)$.

Next, based on discussions in (Lauritzen and Spiegelhalter, 1988) and with medical experts, we discovered widespread agreement and confidence in experts' ability to provide inequality constraints for some of the parameters relating to conditional probabilities of the signs and symptoms given a specific disease. Specifically, this refers to parameters associated with node "*PX*: Positive X-ray?" and node "*D*: Dyspnea?" (marked grey in Figure 3.5).

For the parameter $p(PX = true|LCTB = true)$ (i.e., the probability of a positive[6]

[6]A positive X-ray means that the X-ray shows an abnormality.

X-ray given lung cancer or tuberculosis) experts were confident (based on experience) that this probability is *very likely* and were happy to provide the inequality constraint $p(PX = true|LCTB = true) \geq 0.90$ (the 'ground truth' in (Lauritzen and Spiegelhalter, 1988) is actually 0.98). Similarly, for parameter $p(D = true|LCTB = true, B = true)$ (the probability of dyspnea given tuberculosis or cancer and also bronchitis) experts were happy to assert the inequality constraint greater than 0.80 (the ground truth is 0.90). These very simple and basic expert elicited judgments (which are summarized in Table 3.5) are all that were used.

Table 3.5: Details of 3 elicited judgments for the Asia BN and their corresponding 6 constraints.

Node Name	Elicited Judgments	Elicitation Precedence	Constraint Type
$LCTB$	$p(LCTB = true\|TB = true, LC = true) = 1$ $p(LCTB = true\|TB = true, LC = false) = 1$ $p(LCTB = true\|TB = false, LC = true) = 1$ $p(LCTB = false\|TB = false, LC = false) = 0$	1	Logical
PX	$p(PX = true\|LCTB = true) \geq 0.9$	2	Inequality
D	$p(D = true\|LCTB = true, B = true) \geq 0.8$	2	Inequality

Learning with real expert judgments After introducing these simple constraints, along with data samples of varying sizes (from 10 to 110) we get the estimated parameters. The results for this are shown in Figure 3.7, where we perform all learning approaches (i.e., MLE, MAP, CO and MPL-C).

In Figure 3.7, the x-coordinate denotes the data sample size from 10 to 110, and the y-coordinate denotes the average K-L divergence. For each data sample size, the experiments are repeated 10 times, and the results are presented with their mean and standard deviation. The results show the extent to which the MPL-C method outperforms the MLE and MAP approaches. More importantly, note that MPL-C also outperforms the directly competing CO approach. Specifically, MPL-C achieves very good learning performance with just 10 data samples, where the average K-L divergence is 0.12 ± 0.02 (95% confidence interval using the observed standard deviation), which is much smaller than the results of MLE (1.76 ± 0.52), MAP (0.60 ± 0.05) and CO (0.99 ± 0.36). Of most interest is the observation that the competing methods require vastly more data before they approach the accuracy that MPL-C achieves with very small data samples.

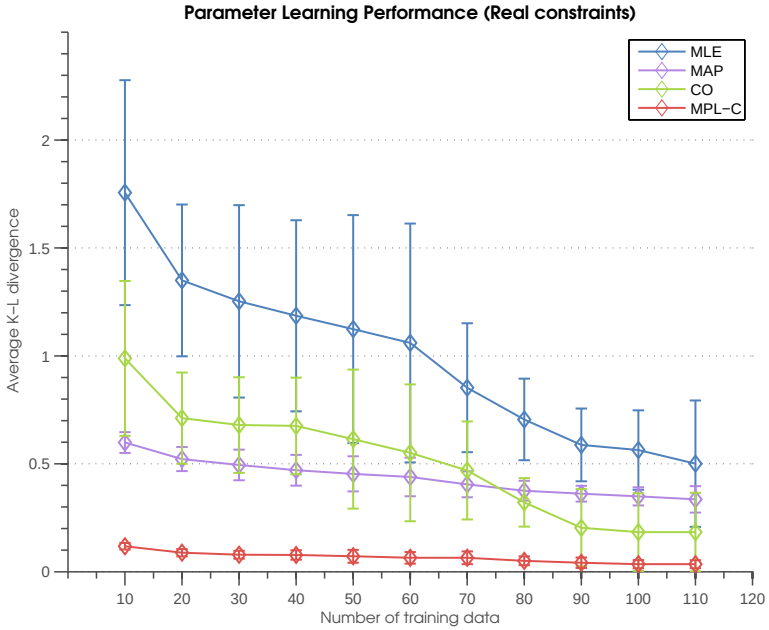
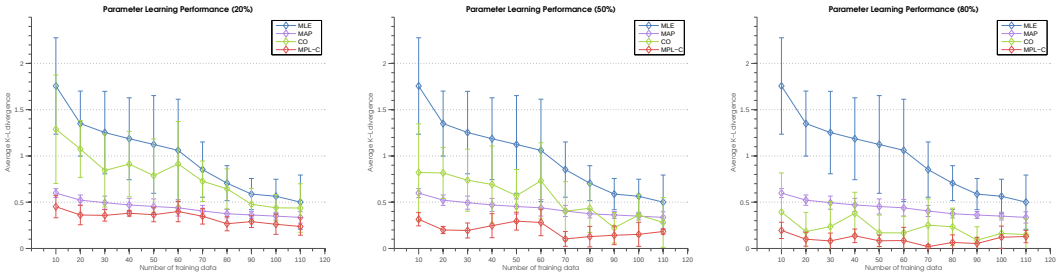


Figure 3.7: Learning results of MLE, MAP, CO and MPL-C for Asia BN with different training data sizes.

Learning with synthetic expert judgments This experiment investigates the learning performance of our algorithm with synthetic expert judgments. Specifically, the number of constraints chosen varies in different experiment settings (we consider the cases of 20%, 50%, and 80% parameters are constrained respectively). In each case the parameters are chosen at random. For each chosen parameter we introduce an approximate equality constraint generated with $\varepsilon = 0.1$. In other words, if the 'true' value for a parameter θ_{ijk} is 0.8, then the constraint will be $|\theta_{ijk} - 0.8| \leq 0.1$ The results are illustrated in Figure 3.8.



(a) 20 % (b) 50 % (c) 80 %

Figure 3.8: Learning results vs. different number of constraints for Asia BN: (a) shows the comparisons for different learning algorithm under 20% constraints ratio; (b) shows the same comparisons under 50% constraints ratio; (c) shows the experiments with 80% constraints ratio, i.e., where most of the parameters are constrained.

As shown in Figure 3.8, for all parameter learning methods, the average K-L di-

vergence shows a decreasing trend with increasing sample sizes, where the small variations are due to randomly select constrained parameters in each sample size. For parameter learning with constraints (CO and MPL-C), the average K-L divergence decreases along with number of constrained parameters' increase. However, the CO fails to outperform the baseline MAP algorithm when small subsets of parameters (i.e., 20% and 50%) are constrained, while the MPL-C method always outperforms the other three learning methods in all three different constrained ratios of parameters.

3.5.2 Different Standard BNs Experiments

In the second set of experiments, we continue the experiments on another 5 standard models (Weather, Cancer, Alarm, Insurance, Hailfinder) that have been widely used for evaluating different learning algorithms. The details of these BNs are listed in Table 3.4. For each BN, the sample sizes are fixed at 10 and 50 with 10 repetitions in each case to get the average learning performance. For the constraints we use the synthetic expert judgment approach (described above as part of the Asia BN experiment in Section 3.5.1). Since it is not usually feasible to get large numbers of real constraints for large BNs with thousands parameters, the ratio of constrained parameters is fixed at 20% for all BNs here.

Table 3.6: Results for MLE, MAP, CO and MPL-C in 5 standard BN parameter learning problems.

Name	Data	MLE	MAP	CO	MPL-C
Weather	10	0.82±0.57*	0.28±0.03*	0.61±0.67*	**0.23**±0.05
	50	0.32±0.26	0.70±0.18*	**0.31**±0.24	0.42±0.17
Cancer	10	1.70±0.49*	0.10±0.03	1.51±0.50*	**0.07**±0.03
	50	0.24±0.05*	**0.02**±0.01*	0.21±0.01*	**0.02**±0.01
Alarm	10	3.95±0.16*	0.92±0.01*	3.20±0.13*	**0.78**±0.02
	50	2.80±0.18*	0.73±0.03*	2.28±0.22*	**0.60**±0.03
Insurance	10	4.10±0.06*	1.80±0.01*	3.44±0.09*	**1.54**±0.03
	50	2.39±0.21*	1.38±0.03*	2.03±0.14*	**1.14**±0.02
Hailfinder	10	4.54±0.04*	0.77±0.01*	3.73±0.07*	**0.66**±0.01
	50	3.38±0.04*	0.57±0.01*	2.81±0.09*	**0.46**±0.01

Table 3.6 shows the average K-L divergence over NPT columns for different learning methods in each BN. The lowest average K-L divergence (best result) in each setting is presented in bold text format. Statistically significant improvements of the best result over competitors are indicated with asterisks * ($p \leq 0.05$). We can see that the MPL-C

method returns the best learning results in all experimental settings except one insignificant exception: in the Weather BN with 50 data samples, the CO algorithm achieves a slightly lower average K-L divergence 0.31 ± 0.24 compared with 0.42 ± 0.17 of MPL-C. However, due to the large standard deviations in these results (this is caused by the heavily unbalanced 'true' probabilities), it is not statistically significant to claim that CO outperforms MPL-C even in this case. In all the other cases in the table, the MPL-C result is statistically significant. Again, the improved performance of MPL-C against the other methods is especially pronounced when the number of samples is small (i.e., 10 in this case).

3.6 A Case Study

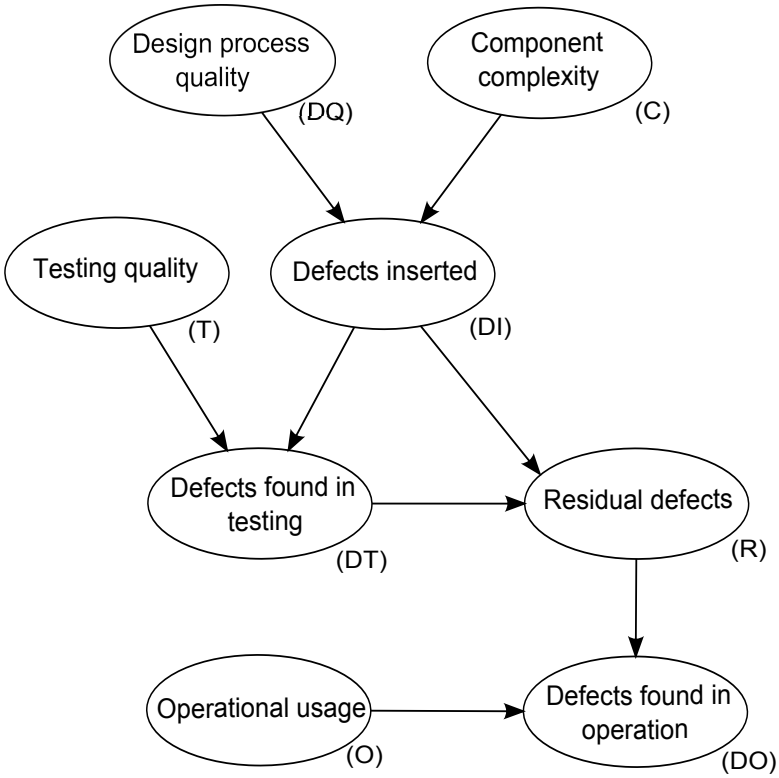


Figure 3.9: The directed acyclic graph of the software defects prediction BN.

This section describes experiments applied to a real software defects prediction BN (Fenton and Neil, 1999; Fenton et al., 2008) in which we incorporate real expert judgments. The software defects prediction BN is used to predict the quality of software in terms of defects found in operation based on observations that may be possible during the software development (such as component complexity and defects found in testing).

The basic version of this BN contains 8 nodes, and its structure is shown in Figure 3.9. The ground truth for the NPTs are presented in (Fenton and Neil, 2012), where each node has 3 states '*low*', '*medium*', and '*high*'. There are therefore 80 independent parameters in the model.

This model structure presented in Figure 3.9 is considered valid for multiple types of organization. While some of the NPTs will always be organization-specific others need to be 'tailored' to each particular organization. That makes this BN a particularly relevant test case for our method, especially as in practice it is known to be extremely difficult to obtain any more than a small number of relevant data samples.

We elicited 10 constraints from real software project experts and these are summarized in Table 3.7. Here, in order to simplify the notation we will use '*l*', '*m*' and '*h*' to represent variable states '*low*', '*medium*', and '*high*'.

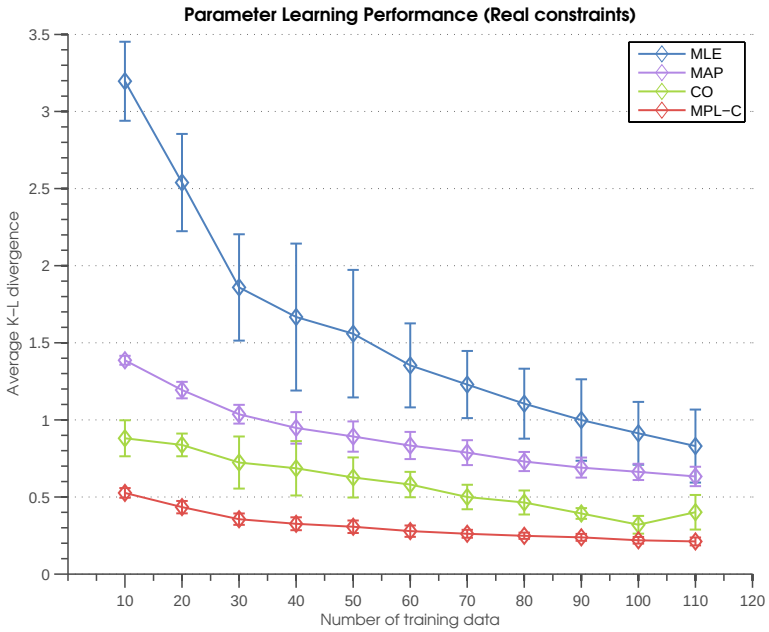


Figure 3.10: Learning results of MLE, MAP, CO and MPL-C for software defects prediction BN with different training data sizes.

Dependence on data sparsity After introducing these real constraints, along with data samples of varying sizes (from 10 to 110) we have the parameter learning results shown in Figure 3.10. Four lines are presented in this chart, where the blue and purple lines represent the learning results of baseline MLE and MAP approaches, green line denotes the results of the CO approach, and the red line shows the learning results of the MPL-C method.

Table 3.7: Details of 10 real expert judgments for the software defects prediction BN and their corresponding 19 constraints.

Index	Description of real expert judgments	Corresponding constraints
1	If design process quality is '*high*' and component complexity is '*low*' then the probability that defects inserted is '*low*' is $\geq 80\%$.	$p(DI = l \mid DQ = h, C = l) \geq 0.80$
2	If design process quality is '*low*' and component complexity is '*high*' then the probability that defects inserted is '*low*' is $\leq 5\%$.	$p(DI = l \mid DQ = l, C = h) \leq 0.05$
3	If defects inserted is '*low*' then the probability that defects found is '*low*' is $\geq 99\%$ (irrespective of testing quality).	$p(DT = l \mid DI = l, T = h) \geq 0.99$ $p(DT = l \mid DI = l, T = m) \geq 0.99$ $p(DT = l \mid DI = l, T = l) \geq 0.99$
4	When testing quality is '*high*' and defects inserted is '*high*' there is $a \leq 10\%$ probability defects found is '*low*'.	$p(DT = l \mid DI = h, T = h) \leq 0.10$
5	When testing quality is '*low*' then the probability that defects found is '*high*' is $\leq 5\%$ (irrespective of defects inserted).	$p(DT = h \mid DI = h, T = l) \leq 0.05$ $p(DT = h \mid DI = m, T = l) \leq 0.05$ $p(DT = h \mid DI = l, T = l) \leq 0.05$
6	If defects inserted is '*low*' then the probability that residual defects is '*low*' is $\geq 99\%$ (irrespective of defects found).	$p(R = l \mid DI = l, DT = h) \geq 0.99$ $p(R = l \mid DI = l, DT = m) \geq 0.99$ $p(R = l \mid DI = l, DT = l) \geq 0.99$
7	If defects inserted is '*high*' and defects found is '*low*' then the probability that residual defects is '*low*' is $\leq 1\%$.	$p(R = l \mid DI = h, DT = l) \leq 0.01$
8	If defects inserted is '*medium*' and defects found is '*high*' then the probability of residual defects is '*low*' is greater than the probability of '*medium*', and the probability of '*medium*' is greater than probability of '*high*'.	$p(R = l \mid DI = m, DT = h) \geq p(R = m \mid DI = m, DT = h)$ $p(R = m \mid DI = m, DT = h) \geq p(R = h \mid DI = m, DT = h)$
9	If operational usage is '*high*' and residual defects is '*high*' then the probability that number of defects found in operation is '*high*' is $\geq 99\%$.	$p(DO = h \mid O = h, R = h) \geq 0.99$
10	If operational usage is '*low*' then the probability that number of defects found in operation is '*high*' is $\leq 20\%$.	$p(DO = h \mid O = l, R = h) \leq 0.20$ $p(DO = h \mid O = l, R = m) \leq 0.20$ $p(DO = h \mid O = l, R = l) \leq 0.20$

As we can see from Figure 3.10, it is clear that the average K-L divergence of all learning algorithms decreases with increasing sample sizes. As expected, when the sample sizes increase, the performance gap between the algorithms decreases dramatically, which means the learning performance of all algorithms will converge with enough training samples. Moreover, the results show again that the MPL-C method

still achieves the best learning performance in the whole data range. Specifically, at 10 data samples, the average K-L divergence for MPL-C is 0.53±0.03, which is much smaller than the results of MLE (3.20±0.26), MAP (1.39±0.03) and CO (0.88±0.12).

Saved data samples To get a better idea of how the learning performance can be improved by MPL-C, we can examine the number of data samples that MLE, MAP and CO require in order to achieve the same average K-L divergence as MPL-C at a specific small sample size. The results for such a comparison are shown in Table 3.8.

The results indicate MPL-C requires much less samples to achieve good learning results than other learning algorithms. For example, MLE, MAP and CO need 250, 160 and 75 data samples respectively to achieve the same average K-L divergence as MPL-C at 10 data samples. Even for 20 data samples, the additional number required for the other methods is large (320, 230 and 105 respectively). This is a very important result because in a typical software development organization having 20 relevant data samples for this problem is considered a large data set and difficult to collect.

Table 3.8: Equivalent data sample size so that MLE, MAP and CO achieve the same performance as MPL-C in the software defects prediction BN.

Data Samples	K-L divergence (MPL-C)	Samples needed by MLE	Samples needed by MAP	Samples needed by CO
10	0.53±0.03	250	160	75
20	0.43±0.04	320	230	105
30	0.36±0.04	405	315	142
40	0.33±0.04	410	370	150
50	0.31±0.04	470	405	160

3.7 Summary

Purely data driven techniques (such as MLE and MAP) for learning the NPTs in BNs often provide inaccurate results even when the datasets are very large. That is why it is widely accepted that expert judgment should be used whenever it is available in order to supplement and improve the learning accuracy. However, reliable expert judgment is not only difficult to elicit, but also difficult to incorporate with existing data. We have described an automated method that addresses both of these concerns. It focuses on constraints that are easily described by experts and are incorporated into an extended

version of a multinomial parameter learning model. This model is an auxiliary BN associated with each node whose NPT we wish to learn. The auxiliary BN, which we called MPL-C (multinomial parameter learning with constraints) was implemented (the sample code using AgenaRisk API can be found in Appendix A) and evaluated experimentally against both pure data learning methods (namely MLE and MAP) as well as against the only relevant competing method that incorporates expert judgments with data (namely CO).

We have conducted experiments on 6 standard BN models as well as a real-world BN model that has been used by many technology companies worldwide (for predicting software defects). In all cases, we considered the impact of adding real expert constraints. The experiments demonstrate that, whereas the CO method clearly improves performance compared with conventional MLE and MAP algorithms when enough constraints are added, our MPL-C method achieves the best learning results in almost all experiment settings. MPL-C is especially accurate in comparison to the other methods in situations where the datasets are relatively small. Indeed, MPL-C needs much smaller datasets to achieve accurate learning results. Moreover, even a very small number of expert constraints dramatically improves accuracy under MPL-C.

The practical implications of these results are very important: in most real-world situations (such as that in which the software defects prediction BN is used) the availability of relevant data is extremely limited; even when it is possible to get relevant data it may be too expensive or time consuming to do so. Our results show that in these very common situations a small dataset, together with even a small number of expert judgment constraints, can result in accurate models using MPL-C. Indeed the accuracy can be much greater than what is achievable with a very large dataset and no expert constraints. Hence, the experimental results suggest that our MPL-C method may represent a significant step forward in parameter learning of BNs in the very common situation where there is sparse data but a small amount of expert judgment.

While the experimental results are extremely promising, there is an obvious area for improvement – incorporating constraints across NPT columns. In this chapter we have considered only constraints that affect a single column of NPT values. However, experts may be able to provide constraints between parameters in different columns, for example, in the software defects prediction BN, an expert could assert that “the

probability of defects inserted is '*high*' given '*high*' component complexity is greater than the probability of it is '*high*' given '*low*' component complexity". Given the remarkable boost in accuracy achieved by incorporating the limited types of constraints considered in this chapter, it is reasonable to conclude that incorporating these other types would also lead to greatly improved learning accuracy. This will be investigated in the next chapter.

Chapter 4

Parameter Learning with Exterior Constraints

Chapter 3 showed that incorporating parameter constraints can improve learning performance of BNs in real-world applications with limited data. However, the expert constraints were restricted to what we called 'interior' constraints, which apply only to constraints between entries in the same NPT column. But experts are often typically able to provide constraints that apply to entries of different columns. We call these exterior constraints. In this chapter we extend the auxiliary BN method (discussed in Chapter 3) to tackle parameter learning with exterior constraints. This extended model also addresses (i) how to estimate target parameters with both data and constraints, and (ii) how to fuse the weights from different causal relationships in a robust way. Finally, we demonstrate the successful applications to learn the 6 standard BN models described in Table 3.4 as well as the real-world software defects prediction BN (introduced in Section 3.6) with scarce data.

4.1 Interior and Exterior Constraints

In Chapter 3, we mainly discussed the parameter constraints restricted to a single probability table column; for example:

"$p(\text{cancer} = \textit{true} | \text{smoker} = \textit{true}) \geq 0.01$)" or

"$p(\text{cancer} = \textit{true} | \text{smoker} = \textit{true}) \geq p(\text{cancer} = \text{false} | \text{smoker} = \textit{true})$"

In this chapter we extend this to exterior parameter constraints (across NPT columns) such as:

"$p(\text{cancer} = \textit{true} | \text{smoker} = \textit{true}) \geq p(\text{cancer} = \textit{true} | \text{smoker} = \textit{false})$"

These two constraints are different according to the constrained parameters' parent state configurations.

Definition 4.1.1. For any variable X_i in a BN, if the two associated parameters θ_{ijk} and $\theta_{ijk'}$ ($k \neq k'$) in X_i share the same parent state configuration $\pi_i = j$, we call $\theta_{ijk} \geq \theta_{ijk'}$ or $\theta_{ijk} < \theta_{ijk'}$ an *interior constraint*.

We showed in chapter 3 that significant improvements to NPT learning can be achieved from a set of expert provided interior constraints. However, in many BNs, the available expert judgments are about constraining parameters with different parent state configurations. These constraints are referred to as exterior constraints, and defined as follows:

Definition 4.1.2. For any variable X_i in a BN, if the two associated parameters θ_{ijk} and $\theta_{ij'k}$ in X_i have different parent state configurations $\pi_i = j$ or $\pi_i = j'$ ($j \neq j'$), we call $\theta_{ijk} \geq \theta_{ij'k}$ or $\theta_{ijk} < \theta_{ij'k}$ an *exterior constraint*.

This kind of constraint is encoded in monotonic effects of DAGs which can greatly reduce the burden of expert judgment elicitation. Before examining exterior constraints in detail, we firstly discuss the monotonic effects in the next section.

4.2 Monotonic Effects

Definition 4.2.1. For any dependent relationship $X_j \rightarrow X_i$ in a BN with ordered categorical variables, if an increase in X_j leads to an increase in X_i no matter the values of other variables in $\pi_i \setminus \{X_j\}$, we call this *positive* monotonic effect $X_j \xrightarrow{+} X_i$. Otherwise, if an increase in X_j leads to a decrease in X_i no matter the values of other variables in $\pi_i \setminus \{X_j\}$, we call this *negative* monotonic effect $X_j \xrightarrow{-} X_i$.

A zero effect, denoted by arc sign $X_j \xrightarrow{0} X_i$ is defined analogously that an increase in X_j will not change the value of X_i. However, this effect is rare in real-world BNs and will not be discussed in this thesis. Finally, if there is no positive or negative monotonic effect between X_j and X_i, we call this *ambiguous* monotonic effect $X_j \xrightarrow{?} X_i$.

Positive and negative monotonic effects are widely seen in real-world BN applications. For example, smoking increases the risk of getting cancer, while medical treatment will reduce the cancer rate. Let $cdf(\cdot)$ denote the cumulative distribution func-

tion. Thus, the two kinds of monotonic effects can be formulated as exterior constraints as follows:

$$\begin{aligned} X_j \xrightarrow{+} X_i\text{: } cdf^{k_c}(\theta_{ijk}) \geq cdf^{k_c}(\theta_{ij'k}) \\ X_j \xrightarrow{-} X_i\text{: } cdf^{k_c}(\theta_{ijk}) \leq cdf^{k_c}(\theta_{ij'k}) \end{aligned} \tag{4.1}$$

where $1 \leq i \leq n,\ 1 \leq j \leq |\pi_i|,\ 1 \leq j' \leq |\pi_i|,\ 1 \leq k \leq k_c$.

Here both X_j and X_i are ordered categorical variables, j and j' are parent state indices satisfying the inequality relationships $1 \leq j < j' \leq |\pi_i|$. The k_c is the state index for which the cumulative distribution function is evaluated, and satisfies the condition $1 \leq k_c < r_i$. The arc signs ($\xrightarrow{+}$ and $\xrightarrow{-}$) specify the types of the monotonic effect.

As we can see, the negative effect represents the opposite monotonic influence compared with the positive effect. A renown extension of monotonic effects is the "enhanced" formalism (Renooij and van der Gaag, 1999; Renooij and Van der Gaag, 2008), which adds the cut-off value α ($0 \leq \alpha \leq 1$) in these inequality parameter constraints. On this formalism, the extension of equation (4.1) can represent,

- strongly positive monotonic effect, $cdf^{k_c}(\theta_{ijk}) - cdf^{k_c}(\theta_{ij'k}) \geq \alpha$, and
- weakly positive monotonic effect, $0 \leq cdf^{k_c}(\theta_{ijk}) - cdf^{k_c}(\theta_{ij'k}) \leq \alpha$.

The equation for strongly negative monotonic effect and weakly negative monotonic effect are defined analogously.

Models that are fully specified by qualitative monotonic effects ($\xrightarrow{+}, \xrightarrow{-}, \xrightarrow{?}$) are referred to as Qualitative Probabilistic Networks (QPNs) (Wellman, 1990). An efficient sign-propagation algorithm is achieved by restricting the maximal number of node-sign changes during the inference (Druzdzel and Henrion, 1993; Renooij and Van der Gaag, 2008). The inference results answer the question of how the observation of some variables changes the probability of other variables. The combination of QPNs and BNs is referred to as Semi-Qualitative Probabilistic Networks (SQPNs) (Renooij and van der Gaag, 2002), which means parts of the variables are represented by joint probability tables rather than qualitative constraints. Inference and learning in SQPNs is discussed in later work (de Campos and Cozman, 2005).

Same as previous work (Altendorf et al., 2005; Feelders and van der Gaag, 2006), in this thesis, we only use signs of qualitative probabilistic networks and their generated

exterior constraints (as shown in equation (4.1)) to constrain the probabilities in the standard BN parameter learning. The challenge here is dealing with the synergies of the monotonic effects – the interactions among monotonic effects. For example, the values of Y can be jointly influenced by the values of two parent nodes X_1 and X_2 in the local structure $X_1 \rightarrow Y \leftarrow X_2$.

Assume all nodes are binary ('T' and 'F'), previous work (de Campos and Cozman, 2005) has defined different kinds of synergies for X_2 and X_3 on their common effect Y, that is:

- **Positive Additive synergy**
  $p(y_T|x_{1T},x_{2T}) + p(y_T|x_{1F},x_{2F}) \geq p(y_T|x_{1T},x_{2F}) + p(y_T|x_{1F},x_{2T})$
- **Negative Additive synergy**
  $p(y_T|x_{1T},x_{2T}) + p(y_T|x_{1F},x_{2F}) \leq p(y_T|x_{1T},x_{2F}) + p(y_T|x_{1F},x_{2T})$
- **Positive Product synergy**
  $p(y_T|x_{1T},x_{2T})p(y_T|x_{1F},x_{2F}) \geq p(y_T|x_{1T},x_{2F})p(y_T|x_{1F},x_{2T})$
- **Negative Product synergy**
  $p(y_T|x_{1T},x_{2T})p(y_T|x_{1F},x_{2F}) \leq p(y_T|x_{1T},x_{2F})p(y_T|x_{1F},x_{2T})$

Recent work (Yang and Natarajan, 2013) has shown that synergies could improve the BN parameter learning performance with limited data. However, same as synergies discussed above, the synergies used in their work are homogeneous – that the involved monotonic effects either are positive or negative, i.e., in positive additive synergy, the monotonic effects encoded in two arcs ($X_1 \rightarrow Y$ and $X_2 \rightarrow Y$) are both positive. Real-world BNs usually contain nodes whose parents provide a mixture of positive and negative effects, which is also referred to as heterogeneous synergies.

The challenges of learning with these synergies are two-fold:

(i) The interactions of heterogeneous monotonic effects may produce ambiguous effect, i.e., the additive synery of one positive monotonic effect and one negative monotonic effect.

(ii) Increasing the number of node states would exponentially increase the number of generated constraints, which is computationally intractable.

To mediate this challenge, we introduce some assumptions to restrict constraint sizes and propose a model for generic synergy of monotonic effects in the next section.

4.3 A Generic Synergy of Monotonic Effects

In this section, we introduce a generative form of the exterior constraint equation, which supports homogeneous/heterogeneous synergies with different weights. This model is inspired by the previous "enhanced" formalism for QPNs (Renooij and van der Gaag, 1999; Renooij and Van der Gaag, 2008) via introducing an overall margin of the synergies.

Assume we have a BN with variables $V = \{Y, X_1, X_2, \ldots, X_n\}$ and the simple inverted naive structure, which means the variable Y is the shared child of $X_1, X_2, \ldots, X_n$. Then our generative exterior constraint is:

$$\begin{cases} cdf(p(Y|\pi_Y = j)) - cdf(p(Y|\pi_Y = j')) \geq M_{jj'} & if\, M_{jj'} > 0 \\ cdf(p(Y|\pi_Y = j)) - cdf(p(Y|\pi_Y = j')) \leq M_{jj'} & if\, M_{jj'} < 0 \end{cases} \tag{4.2}$$

where $M_{jj'} = \sum_{i=1}^{n} M^i_{jj'} = \sum_{i=1}^{n} w_i \cdot cl_i \cdot \varepsilon^i_{jj'}$ and $1 \leq j < j' \leq |\pi_Y|$.

$M_{jj'}$ represents the overall margin of the synergies, which is the summation of each single margin $M^i_{jj'}$. The $M^i_{jj'}$ contains three terms:

- $w_i \geq 1$ represents the global weight (the subjective confidence) of the dependent relationship $X_i \rightarrow Y$, its default value $w_i = 1$ indicates there is no subjective confidence on the influence. This term easily explains that some parent nodes could have higher effect on the child node, e.g., smoking behaviour has a higher effect on the risk of people getting tuberculosis than the factor that people has recently visited Asia.

- cl is the effect label ($cl_i = 1$ indicates the positive effect $X_i \xrightarrow{+} Y$; $cl_i = -1$ represents the negative effect $X_i \xrightarrow{-} Y$). This term is derived from the definition of monotonic effect (Definition 4.2.1).

- $\varepsilon^i_{jj'}$ is a local term that describes the confidence of the inequality introduced by state configuration gap in the synergy of monotonic effects. For example, in a single monotonic effect, $\varepsilon^i_{jj'}$ is a small positive value proportional to the state configuration distance in X_i under two indices j and j' in $\pi_Y = \{X_1, X_2, \ldots, X_n\}$.

The term $\varepsilon^i_{jj'}$ denotes the strength of the monotonic effect introduced by the difference between two parent state configurations. For a simple example[1], a company's customer loss rate is affected by four binary risk factors. The presence of any one risk factor increases the probability of the loss of a customer. Thus, all the dependent relationships encode positive monotonic effects. According to the statistic results, compared with the configuration that only one risk factor is presented, the customer loss rate is much higher in the configuration when four risk factors are present. Such difference is proportional to the difference between two state configurations, and can be modelled by $\varepsilon^i_{jj'}$.

To calculate $\varepsilon^i_{jj'}$, we need to introduce a function called $ind2sub(j,i)$, which converts the variable Y's parent configuration index j into the associated sub-index of its i-th parent. If Y only has one parent, the sub-index is same as the overall index, $ind2sub(j,i) = j$. Here we have:

$$\varepsilon^i_{jj'} = -\left(\frac{ind2sub(j,i) - ind2sub(j',i)}{\lambda |X_i|}\right) \tag{4.3}$$

Here $\lambda > 1$ is the trade-off parameter that controls the effect of the confidence introduced by the state configuration gap. Note, in this thesis and many real-world BN applications, lower variable state index corresponds to higher verbal value (more positive meaning), e.g., the *true* and *false* states of a event correspond to state index 1 and 2 respectively.

As shown in equation (4.2), the type ($\geq$ or $\leq$) of the exterior constraint is decided by the value of the margin. The margin is equal to zero ($M = 0$) only in the situation where the influences of different monotonic effects are intermediate in the shared child node. Thus, there is no associated exterior constraints.

Next, we present a simple example of our model: we assume the target variable Y is binary, and it has two binary parents X_1 and X_2 with 'T' and 'F' states. Thus we have four state configurations of $\pi_Y = \{X_1, X_2\}$:

$\pi_Y = 1:\ X_1 = x_{1T},\ X_2 = x_{2T}$

$\pi_Y = 2:\ X_1 = x_{1T},\ X_2 = x_{2F}$

$\pi_Y = 3:\ X_1 = x_{1F},\ X_2 = x_{2T}$

[1] https://www.eecs.qmul.ac.uk/norman/papers/the-problems-with-big-data.pdf

$\pi_Y = 4:\ X_1 = x_{1F},\ X_2 = x_{2F}$

Assume the first monotonic effect is positive $X_1 \overset{+}{\rightarrow} Y$, and the second monotonic effect is negative $X_2 \overset{-}{\rightarrow} Y$. Therefore, we have 6 combinations of two indices in $|\pi_Y|$. These combinations and their generated exterior constraints are listed as below:

$$cdf(p(Y|\pi_Y = 1)) - cdf(p(Y|\pi_Y = 4)) \geq w_1 \cdot \varepsilon^1_{14} - w_2 \cdot \varepsilon^2_{14} = \frac{w_1}{2\lambda} - \frac{w_2}{2\lambda}$$
$$cdf(p(Y|\pi_Y = 1)) - cdf(p(Y|\pi_Y = 3)) \geq w_1 \cdot \varepsilon^1_{13} - w_2 \cdot \varepsilon^2_{13} = \frac{w_1}{2\lambda}$$
$$cdf(p(Y|\pi_Y = 1)) - cdf(p(Y|\pi_Y = 2)) \leq w_1 \cdot \varepsilon^1_{12} - w_2 \cdot \varepsilon^2_{12} = -\frac{w_2}{2\lambda}$$
$$cdf(p(Y|\pi_Y = 2)) - cdf(p(Y|\pi_Y = 4)) \geq w_1 \cdot \varepsilon^1_{24} - w_2 \cdot \varepsilon^2_{24} = \frac{w_1}{2\lambda}$$
$$cdf(p(Y|\pi_Y = 2)) - cdf(p(Y|\pi_Y = 3)) \geq w_1 \cdot \varepsilon^1_{23} - w_2 \cdot \varepsilon^2_{23} = \frac{w_1}{2\lambda} + \frac{w_2}{2\lambda}$$
$$cdf(p(Y|\pi_Y = 3)) - cdf(p(Y|\pi_Y = 4)) \leq w_1 \cdot \varepsilon^1_{34} - w_2 \cdot \varepsilon^2_{34} = -\frac{w_2}{2\lambda}$$

In addition, there is no subjective judgments on their weights, i.e. $w_1 = w_2 = 1$. Thus, the margin of the first equation above equals to zero ($\frac{w_1}{2\lambda} - \frac{w_2}{2\lambda} = 0$), and this equation is discarded. Also, because Y is binary, this means $Y = \{y_T, y_F\}$. Therefore, each of above equation could be expanded into two exterior constraints, for example, the fifth equation above could be expanded as:

$$p(y_T|x_{1T}, x_{2F}) - p(y_T|x_{1F}, x_{2T}) \geq \frac{1}{\lambda}$$
$$p(y_T|x_{1T}, x_{2F}) + p(y_F|x_{1T}, x_{2F}) - p(y_T|x_{1F}, x_{2T}) - p(y_F|x_{1F}, x_{2T}) = 0$$

Note the equality only happens when y reaches the full range (the biggest) value in $cdf(p(y))$.

As we know, the size of total parent configurations $|\pi_Y| = \prod_{i=1}^{n} |X_i|$ increases exponentially with an increase of parent nodes, thus we have many combinations of two indices in $|\pi_Y|$ and large computational complexity. A detailed computational analysis is provided in the next section.

4.4 The MPL-EC Model

In this section, we present the extended MPL-C model (Multinomial Parameter Learning with Exterior Constraints, MPL-EC) to encode the constraints in equation (4.2).

4.4.1 Model Construction

For any monotonic effect, we need to introduce a set of shared children nodes to model the introduced constraints $C_1, C_2, \ldots, C_r$ (see Figure 4.1). The size of the constraints set is equal to the number of states (ranges from 1 to r) in variable Y. In order to simplify

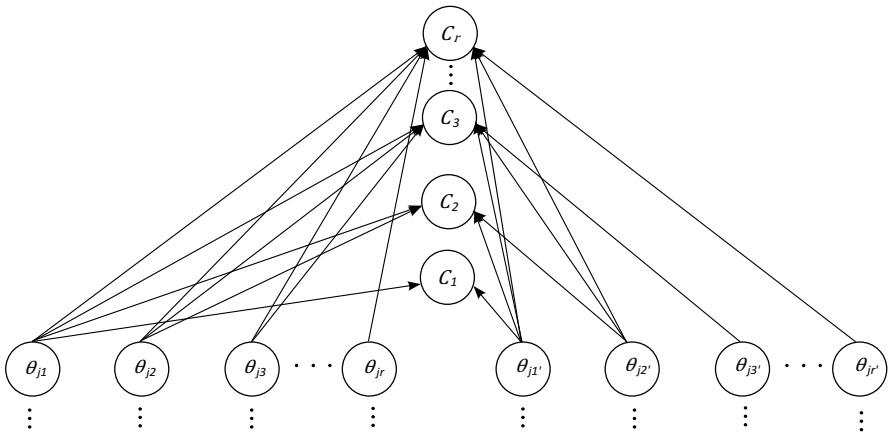
the notation, we use θ_{jk} and $\theta_{j'k}$ ($k = 1$ to r) to represent parameters in Y under two different state configurations of X_i (j and j'). Therefore, for a single positive monotonic effect $X_i \xrightarrow{+} Y$, we have the following arithmetic constraints encoded in the MPL-EC model to constrain the parameters under two state configurations of X_i:

$$\begin{cases} C_1 : \theta_{j1} - \theta_{j'1} \geq w_i \cdot cl_i \cdot \varepsilon^i_{jj'} \\ C_2 : \theta_{j1} + \theta_{j2} - \theta_{j'1} - \theta_{j'2} \geq w_i \cdot cl_i \cdot \varepsilon^i_{jj'} \\ C_3 : \theta_{j1} + \theta_{j2} + \theta_{j3} - \theta_{j'1} - \theta_{j'2} - \theta_{j'3} \geq w_i \cdot cl_i \cdot \varepsilon^i_{jj'} \\ \vdots \\ C_r : \sum_{k=1}^{r} (\theta_{jk} - \theta_{j'k}) = 0 \end{cases} \tag{4.4}$$

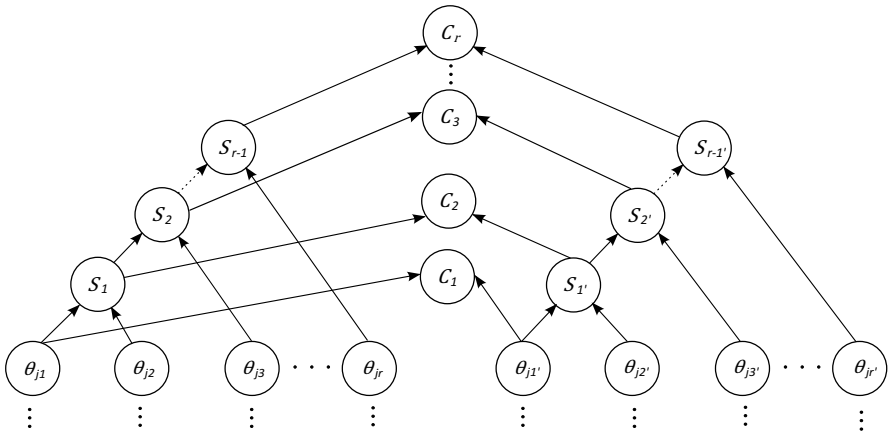
In the last exterior constraint equation C_r, two sides of the relative relationship are equal to each other. As we can see there are an additional $(1+n)n$ edges when we introduce n constraint nodes. To reduce the model complexity it must be replaced by an equivalent model whose structure has a restricted number of parents.

Previous work has proposed a binary factorization algorithm Neil et al. (2012) to improve the efficiency of the DDJT algorithm. This idea can also be applied here to produce an alternative model of the straightforward MPL-EC. The new model is called binary summation model, which introduces an additional $2(n-1)$ auxiliary nodes $(S_1, S_{1'}, ..., S_{r-1}, S_{r-1'})$ that only encode the simple sum arithmetic equations to model the summations of its parents. This model has the same number of edges as the straightforward model, but the maximal number of parents is fixed as two in this model. This avoids the parent state combination explosion problem. The detail of its structure can be found in Figure 4.1 (b).

After inference in MPL-EC models (Figure 4.1 (b)) with the observed data statistics, we can get the updated parameters θ_{jk} and $\theta_{j'k}$. However, because there are multiple parent configurations of j and j', we have more than one estimation of θ_{jk} and $\theta_{j'k}$. Thus, weighted average is applied to all estimations of $\theta_{jk}/\theta_{j'k}$ with greater weight given to those estimations produced by greater constraint margin $|M_{jj'}|$.



(a) The straightforward model of introducing exterior constraints



(b) The binary summation model of introducing exterior constraints

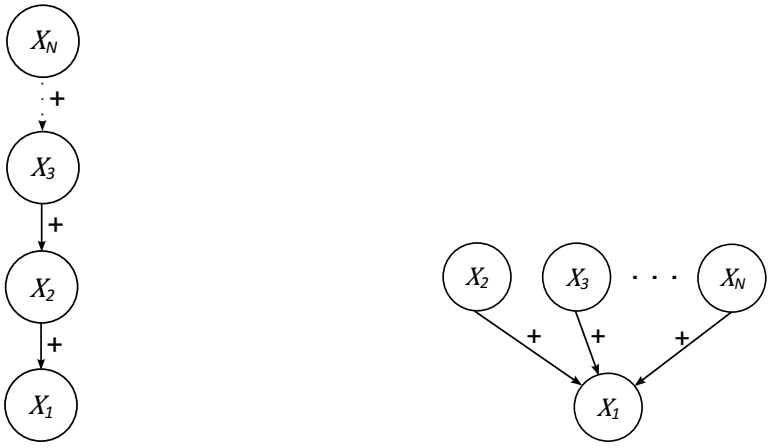
Figure 4.1: The straightforward MPL-EC model and its alternative binary summation model. Due to the space limitation, the MPL-EC model presented here only display the part for modeling introduced constraints, the left part for modeling multinomial parameter learning is not displayed, which is the same as MPL-C in Chapter 3.

4.4.2 Computational Complexity Analysis

The bounded number of parents ensures the MPL-EC model has the same treewidth as the MPL-C model. Therefore, the bottleneck in terms of efficiency of the MPL-EC learning lies in the total number of exterior constraints generated from the monotonic effects. Assume there are n nodes in a BN (each node has r states), and $n-1$ positive monotonic effects. The total number of generated exterior constraints e satisfies:

$$\frac{1}{2}(r^3-r^2)(n-1)=\frac{r!}{2!(r-2)!}r(N-1)\leq e\leq\frac{r^{(n-1)}!}{2!(r^{(n-1)}-2)!}r=\frac{1}{2}(r^{(n^2-2n+2)}-r^n) \tag{4.5}$$

The total number of exterior constraints e equals to the product of total number of parent configurations and the number of child node states. The lower bound and upper bound of e correspond to two network structures in Figure 4.2 (a) and 4.2 (b) respectively.



(a) The lower bound structure representation **(b)** The upper bound structure representation

Figure 4.2: BN representations for generating lower and upper bound number of constraints.

As we can see from the equation (4.5), the increase of n would greatly increase the generated exterior constraints, and make the MPL-EC learning problem hard to solve. We will demonstrate this empirically in the experiments.

4.4.3 An Illustrative Example of MPL-EC

In this subsection, we use another simple example to demonstrate the exterior constraints and its generated MPL-EC model. This example encodes the simplest single positive causal connection: $X \xrightarrow{+} Y$, where two nodes involved are both binary with 'T' and 'F' states. Therefore, we have two parameter columns under two parent state instantiations to estimate in Y, which are $p(Y|x_T)$ and $p(Y|x_F)$. Its MPL-EC model is shown in Figure 4.3.

The detail of the exterior constraints encoded in the constraint nodes of MPL-EC is:

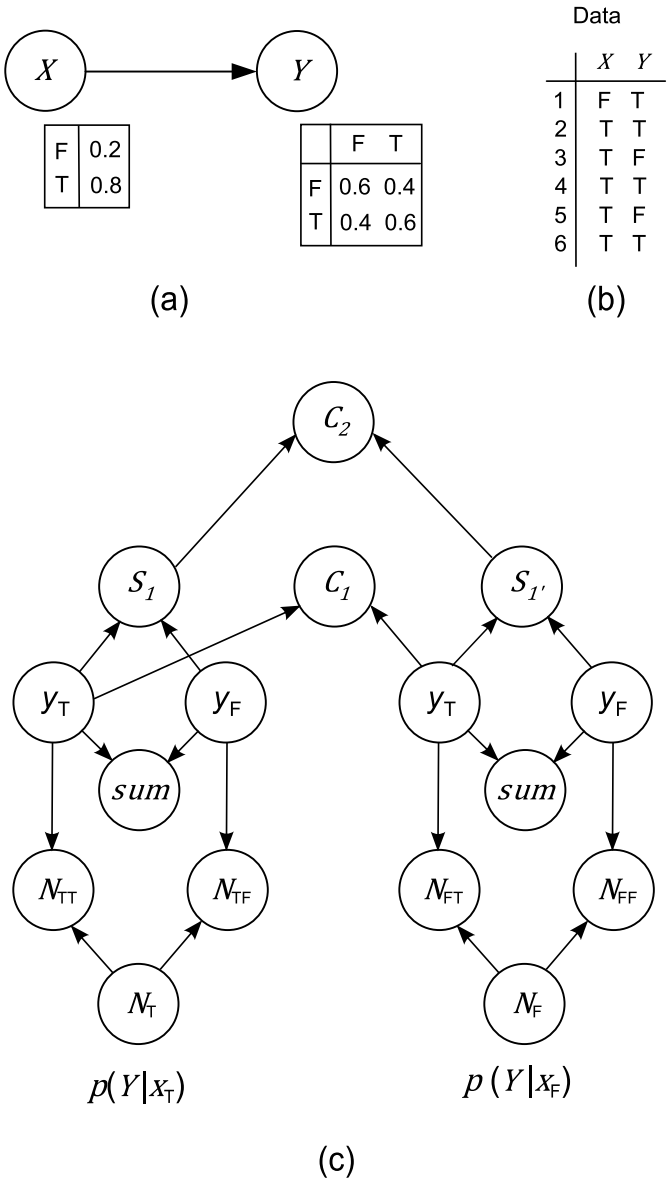


Figure 4.3: The two-node BN and its training data in the simple example: (a) The DAG and its associated NPTs. (b) The 6 data records for the two variables in the BN. (c) The MPL-EC model for estimating the parameters in Y. The constraint nodes are modelled as binary (*true*/*false*) nodes with expressions that specify the constraint relationships between its parents. For example, the expression statement for C_1 is: $if(p(y_T|x_T) - p(y_T|x_F) \geq w \cdot cl \cdot \varepsilon, true, false)$.

$$\begin{cases} S_1 : p(y_T|x_T) + p(y_F|x_T) \\ S_{1'} : p(y_T|x_F) + p(y_F|x_F) \\ C_1 : p(y_T|x_T) - p(y_T|x_F) \geq w \cdot cl \cdot \varepsilon \\ C_2 : S_1 - S_{1'} \geq w \cdot cl \cdot \varepsilon \end{cases} \tag{4.6}$$

where $cl = 1$, $\varepsilon = \frac{1}{2\lambda}$. according to above definition, and w represents the subjective confidence whose value can be chosen empirically from the domain knowledge.

Based on the statistics on the dataset (Figure 4.3 (b)) and previous definition, we have: $N_T = 5$ ($N_{TF} = 2, N_{TT} = 3$) under the condition of $X = x_T$, and $N_F = 1$ ($N_{FF} = 0$, $N_{FT} = 1$) in the state initiation of $X = x_F$. Therefore, the MLE of Y are $p(y_T|x_T) = 0.6$ and $p(y_T|x_F) = 1$. As we can see, the estimation of $p(y_T|x_F)$ is far away from the ground truth (0.6 and 0.4) due to the scarce data observations under the $X = x_F$ condition.

With the above data observations, we now can set the evidence for certain nodes including constraint nodes (all are set as '*true*' observations), number of trials, total numbers, and the summation of all the estimated parameters. Based on this evidence, the inference in the MPL-EC is to compute the discretized posterior marginals of each of the unknown nodes $y_{F/T}$ (these are the nodes without evidence) via the DDJT algorithm described in Algorithm 3.1. This algorithm alternates between two steps: 1) performing dynamic discretization, which searches and splits the regions with the highest relative entropy error determined by a bounded K-L divergence with the current approximated estimates of the marginals; 2) performing junction tree inference, which updates the posterior of the marginals. At convergence, the mean value of $y_{F/T}$ will be assigned as the final corresponding NPT cell values. After inference with the model in Figure 4.3 (c), we have $p(y_T|x_T) = 0.67$ and $p(y_T|x_F) = 0.50$, which are more reasonable than the MLE results.

4.5 Examples of Monotonic Effects in Some Well-known BNs

Incorporating exterior constraints encoded in monotonic effects could help the BN parameter learning. Here, given the true NPTs in Asia, Weather, Cancer, Alarm, Insur-

ance and Hailfinder BNs, we investigate the monotonic effect for each edge of these BNs using Algorithm 4.1. The descriptions of these BNs are listed in Table 3.4.

Algorithm 4.1 describes the method of eliciting positive/negative monotonic effect labels from the BN with known NPTs. According to the definition 4.2.1, a positive/negative monotonic effect only shows the monotonic influence between two nodes connect by the dependent edge. For the parents not connected by the edge in DAG, their influences on the child will be marginalized. For example, $p(X_i = k|X_{i'} = j)$ and $p(X_i = k|X_{i'} = j')$ (Line 14 and 15, Algorithm 4.1) are marginalized conditional probability of $p(X_i|X_{i'})$, where the influences of variables in set $\pi_i \setminus \{X_{i'}\}$ are marginalized. After marginalizing out the influences of unrelated variables, the final monotonic effect label for each edge is decided by the definition. In Algorithm 4.1, the *numel* function (Line 33 and 36) is used to count the total number of elements in a set.

Table 4.1 summarises the elicited monotonic effects for the 6 standard BNs. As we can see, monotonic effects exist in all of them. Specifically, all the edges in Asia, Weather and Cancer BNs encode either positive or negative monotonic effects. In the Alarm, Insurance and Haifinder BNs, the positive/negative monotonic effects are partially existing, specifically there are 52.2%, 46.2% and 47.0% edges encode such monotonic effects in each BN respectively. The details are shown in Figures 4.4–4.7.

Table 4.1: Monotonic effects in Asia, Weather, Cancer, Alarm, Insurance and Hailfinder BNs

Name	Nodes	Arcs	$\overset{+}{\rightarrow}$	$\overset{-}{\rightarrow}$	$\overset{?}{\rightarrow}$
Asia	8	8	8	0	0
Weather	4	4	3	1	0
Cancer	5	5	5	0	0
Alarm	37	46	17	7	22
Insurance	27	52	13	11	28
Hailfinder	56	66	26	5	35

$\overset{+}{\rightarrow}$, $\overset{-}{\rightarrow}$ and $\overset{?}{\rightarrow}$ indicate *positive*, *negative* and *ambiguous* monotonic effect respectively.

INPUT : Bayesian network $\{V, G\}$
OUTPUT: Monotonic effect labels $M_{ii'}$

for *each node* $i = 1$ **to** n **do**
  if $\pi_i = \emptyset$ **then**
    root node;
  else
    for *node* $i' = i+1$ **to** n **do**
      if $X_{i'} \rightarrow X_i \notin G$ **then**
        no such edge;
      else
        tmplabel = $\emptyset$;
        for *each parent configuration* $j = 1$ **to** $|\pi_i|$ *in* i' **do**
          for $j' = j+1$ **to** $|\pi_i|$ **do**
            cdf1 = 0, cdf2 = 0;
            for *each state* $k = 1$ **to** $r_i - 1$ **do**
              cdf1 = cdf1 + $p(X_i = k|X_{i'} = j)$, ;
              cdf2 = cdf2 + $p(X_i = k|X_{i'} = j')$;
              if *cdf1* $\geq$ *cdf2* **then**
                tmpindex(k) = 1;
              else
                tmpindex(k) = -1;
              end
            end
            if *sum(tmpindex)* $= r_i - 1$ **then**
              tmplabel = $\{tmplabel, 1\}$;
            else
              if *sum(tmplabel)* $= -(r_i - 1)$ **then**
                tmplabel = $\{tmplabel, -1\}$;
              else
                tmplabel = $\{tmplabel, 0\}$;
              end
            end
          end
        end
        if *sum(tmplabel) = numel(tmplabel)* **then**
          $X_{i'} \overset{+}{\rightarrow} X_i$;
        else
          if *sum(tmplabel)= −numel(tmplabel)* **then**
            $X_{i'} \overset{-}{\rightarrow} X_i$;
          else
            $X_{i'} \overset{?}{\rightarrow} X_i$;
          end
        end
      end
    end
  end
end
return $M_{ii'} = \left\{ X_{i'} \overset{+,-,?}{\rightarrow} X_i \mid i, i' = 1, 2, ..., n \right\}$;

Algorithm 4.1: The algorithm of eliciting monotonic effects from BNs

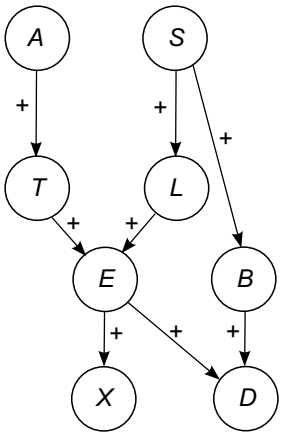


Figure 4.4: The monotonic effect labels in the Asia BN (8 positive monotonic effects and 0 negative monotonic effect).

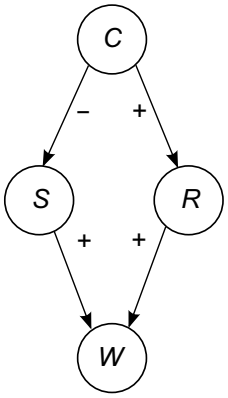


Figure 4.5: The monotonic effect labels in the Weather BN (3 positive monotonic effects and 1 negative monotonic effect).

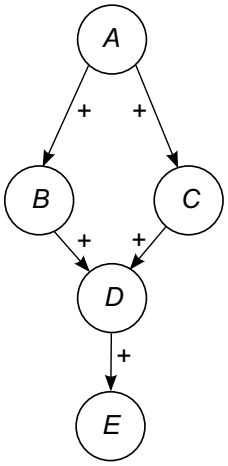


Figure 4.6: The monotonic effect labels in the Cancer BN (5 positive monotonic effects and 0 negative monotonic effect).

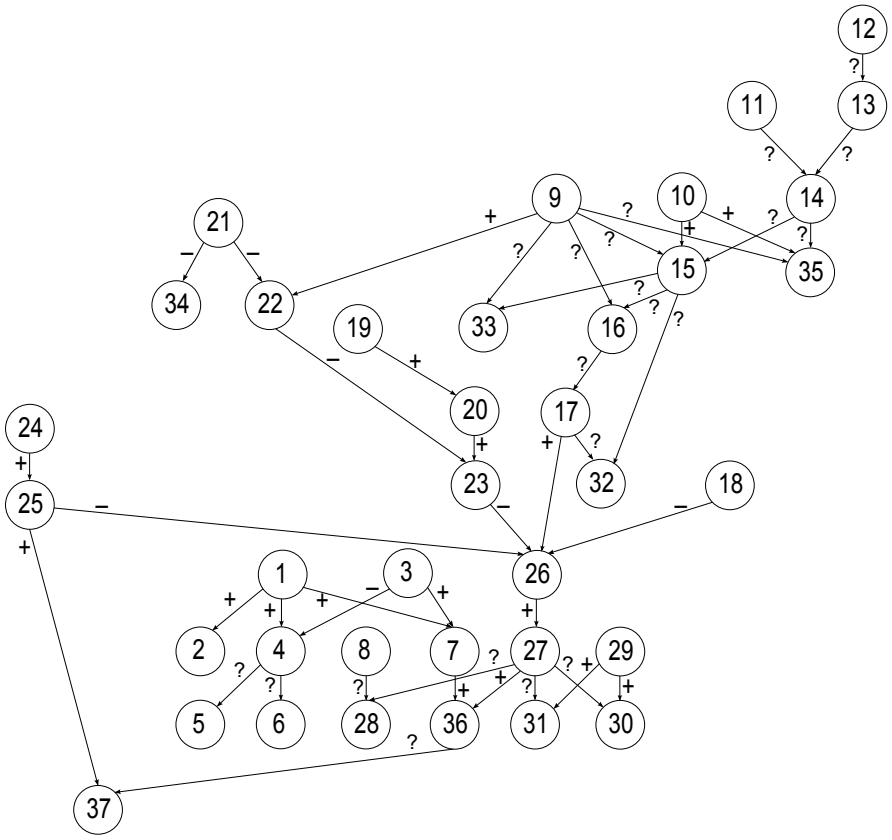


Figure 4.7: The monotonic effect labels in the Alarm BN (17 positive monotonic effects and 7 negative monotonic effects).

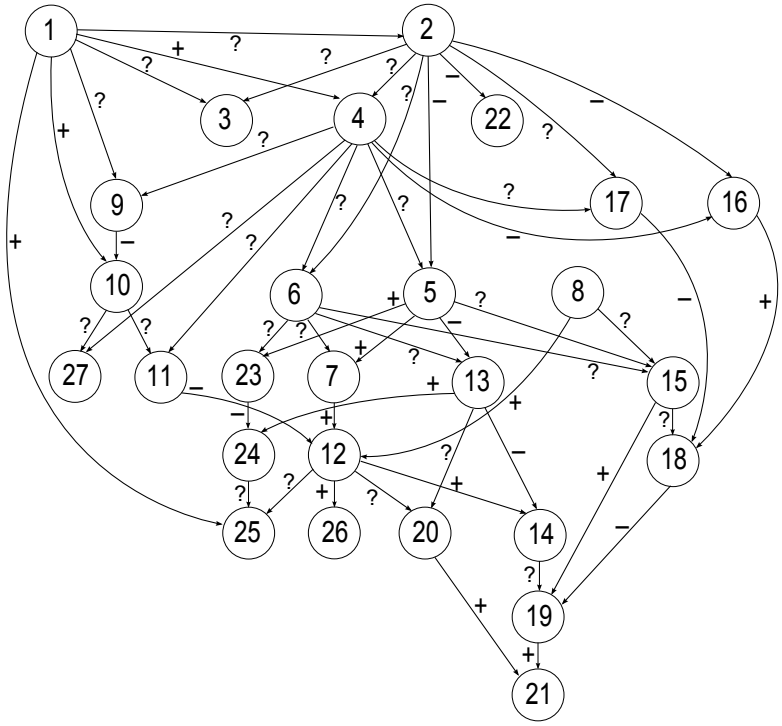


Figure 4.8: The monotonic effect labels in the Insurance BN (13 positive monotonic effects and 11 negative monotonic effects).

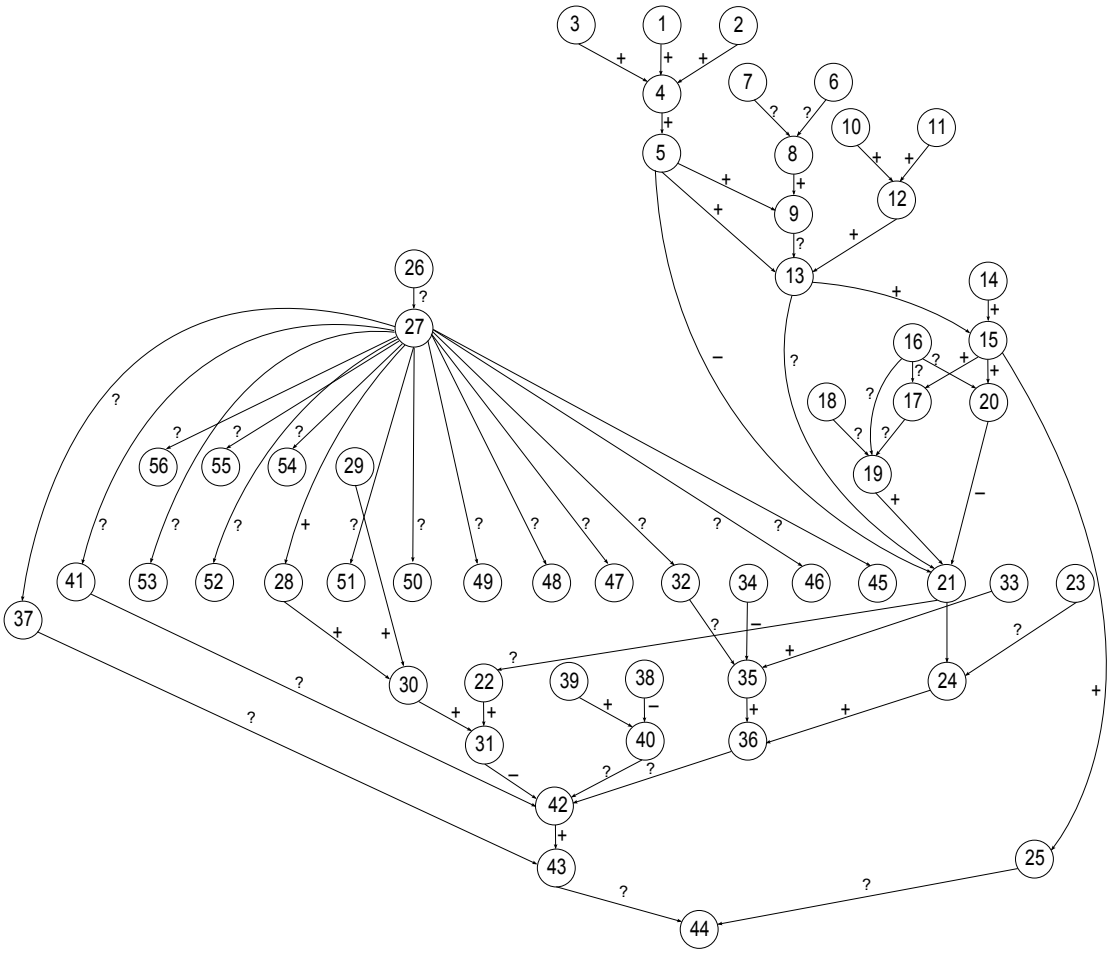


Figure 4.9: The monotonic effect labels in the Hailfinder BN (26 positive monotonic effects and 5 negative monotonic effects).

4.6 Experiments

In this section, we continue to use the 6 standard BNs (Table 3.4) to test the parameter learning performance with exterior constraints. The structure and monotonic effects of these BNs are analysed in Section 4.5. Based on the same experiment setting in Section 3.5, we introduce AUC measurement and consider the following experimental tasks:

- Investigate the effectiveness of the introduced margin in the generic synergy, especially the local term whose value is proportional to the state configuration gap in the synergy.

- Compare the MPL-EC with the state-of-the-art CO algorithm and conventional parameter learning algorithms. Especially, assume the ground truth is unknown, and measure the performance via calculating AUC values (2.23) for different classification tasks.

- Investigate the effect of synthetic wrongly elicited constraints on the final learning results.

- Empirically analyse the computational complexity of the MPL-EC method.

In the following experiments, we will evaluate our MPL-EC method in each of these cases.

4.6.1 The Performance of Introduced Margin

In the first set of experiments, we use the 6 standard BNs to show if the introduced margin in the generic synergy will affect the final learning results. By doing so, we compare the performance of the original MPL-EC and MPL-EC (without margin). Each experiment is repeated 10 times under two data sizes: 10 samples and 50 samples. The distance between learnt BNs and ground truth are measured by average K-L divergence. Thus, lower divergence indicates better learning performance.

In parameter learning, all the monotonic effects and their synergies (either homogeneous or heterogeneous) are converted into exterior constraints by equation (4.2). The generated margin contains three terms: global weight, monotonic effect label and local term. They are set as follows:

- Because the subjective confidence of monotonic effects are hard to acquire in these 6 BNs, we do not consider the influence of these. Thus, the global weight term $w_i = 1$.

- The monotonic effect label term $cl = \{1, -1\}$ has already been elicited in Figures 4.5–4.9, and can be directly used here.

- The local term $\varepsilon^i_{jj'}$ that describes the confidence of the inequality introduced by state configuration gap in the synergy is calculated by equation (4.3).

The results are presented in Figure 4.10, the height of bars show the average K-L divergence between learnt BNs and true BNs. As we can see, the MPL-EC (without margin) shows worse learning results in both of the data sample size settings. Therefore, the estimates of MPL-EC(without margin) are far from the true values, (average K-L divergence of 0.71 and 0.54 for sample sizes 10 and 50 respectively). However, after introducing the margin terms in parameter learning in the MPL-EC model, the

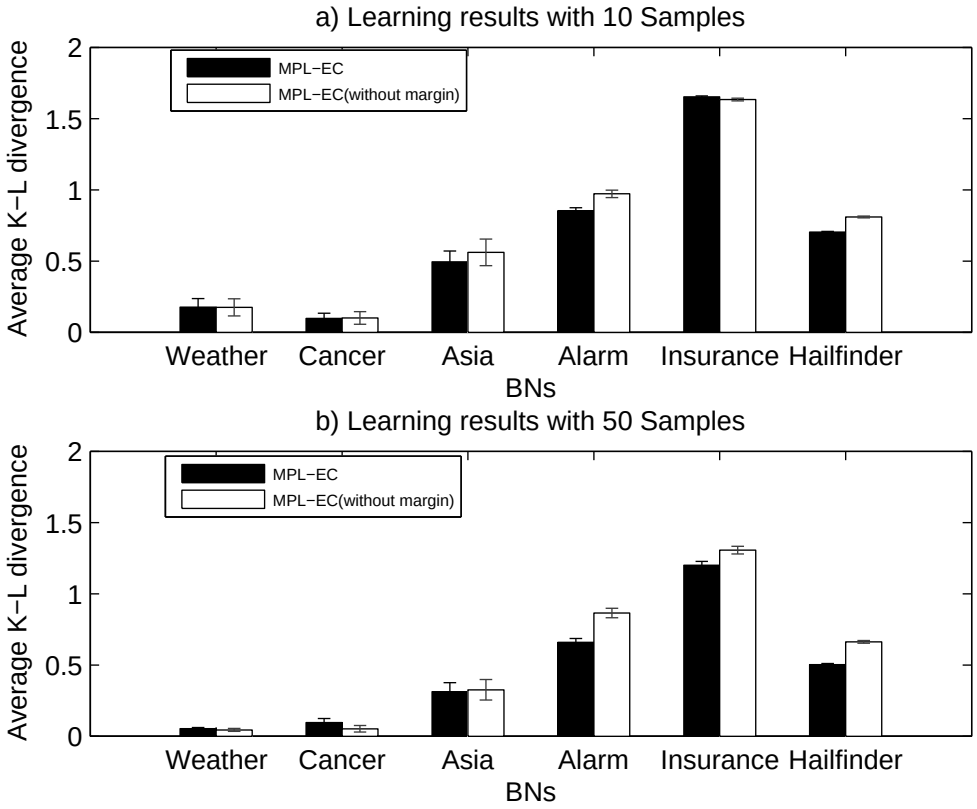


Figure 4.10: The parameter learning performance of MPL-EC (black bars) and MPL-EC without introduced synergy margin (white bars).

MPL-EC reduces the average K-L divergence between the estimated values and true values to 0.66 and 0.47 respectively. These results demonstrate the benefit of introducing margin terms in parameter learning, and show the correctness of our analysis on local weight terms. Next, we will compare the performance of all parameter learning algorithms in different BN parameter learning problems.

4.6.2 The Overall Performance

We continue to use the same experiment setting but considering two measurement methods for the learning results: (i) the average K-L divergence (2.21) over NPT columns in each BN, and (ii) the average AUC value (2.23) of the classification task in each BN.

Table 4.2 shows the average K-L divergence over NPT columns for different learning methods in each BN. The lowest average K-L divergence (best result) in each setting is presented in bold text format. Statistically significant improvements of the best result over competitors are indicated with asterisks * ($p \leq 0.05$).

Table 4.2: Results (average K-L divergence) for MLE, MAP, CO and MPL-EC in 6 standard BN parameter learning problems.

Name	Data	MLE	MAP	CO	MPL-EC
Weather	10	0.57±0.41*	0.27±0.06*	**0.14**±0.04	0.18±0.06
	50	0.05±0.03	0.08±0.02*	**0.04**±0.02	0.05±0.01
Cancer	10	1.65±0.37*	0.11±0.04	0.12±0.06	**0.10**±0.04
	50	0.50±0.23*	**0.04**±0.02	**0.04**±0.03	0.10±0.03*
Asia	10	1.93±0.73*	0.62±0.07*	0.84±0.30*	**0.50**±0.08
	50	0.86±0.30*	0.41±0.05*	0.38±0.18	**0.31**±0.06
Alarm	10	3.94±0.19*	0.94±0.02*	1.59±0.08*	**0.85**±0.02
	50	2.80±0.19*	0.75±0.02*	1.20±0.09*	**0.66**±0.03
Insurance	10	4.06±0.11*	1.80±0.01*	2.40±0.05*	**1.65**±0.01
	50	2.52±0.09*	1.40±0.02*	1.76±0.04*	**1.20**±0.03
Hailfinder	10	4.53±0.03*	0.77±0.00*	1.32±0.03*	**0.70**±0.00
	50	3.40±0.06*	0.57±0.01*	0.85±0.03*	**0.50**±0.01

Table 4.2 summarises the results, where our MPL-EC outperforms the conventional MLE algorithm in all settings. Compared with the state-of-the-art CO algorithm, MPL-EC also improves performance on 9 out of 12 experiments, with an average margin of 36.4% (the average reduction of K-L divergence). There are three insignificant exceptions in the Weather and Cancer BNs where the CO algorithm achieves a slightly lower average K-L divergence compared with MPL-EC. These exceptions may be caused by the bias parameter prior in MPL-EC in these BNs. In conclusion, the overall good results of MPL-EC show the potential benefit of using MPL-EC in parameter learning with scarce data and provided monotonic effect labels.

AUC measurement Next, we use estimated BNs and 500 sampled testing data to predict the state of a specific node given observing all other nodes in the BN. That is to say, we now have 12 BN classifiers, 6 of them are learnt from 10 samples and 6 of them are learnt from 50 samples. In each experiment repeat, one node is randomly selected as the query node, and all other nodes are observed, standard junction tree algorithm is performed to inference the final predictions.

Table 4.3 summarises the results, with higher AUC values indicating better classification performance. The AUC value around 0.5 indicates the classifier has the same performance as completely random guessing. As we can see, the results still show our MPL-EC outperforms the state-of-the-art CO algorithm (improvement on 10 out of 12 experiments, with an average increased AUC margin of 24.6%). However, in the

Table 4.3: Results (AUC) for MLE, MAP, CO and MPL-EC in 6 standard BN parameter learning problems.

Name	Data	MLE	MAP	CO	MPL-EC
Weather	10	0.80±0.08*	0.90±0.04	0.90±0.04	**0.91**±0.03
	50	0.90±0.03	0.90±0.03	**0.91**±0.03	0.90±0.03
Cancer	10	0.46±0.13*	0.70±0.17	0.66±0.16	**0.73**±0.15
	50	0.70±0.13	**0.80**±0.15	**0.80**±0.15	0.79±0.14
Asia	10	0.84±0.14	**0.94**±0.08	0.82±0.15	**0.94**±0.09
	50	0.83±0.19	**0.84**±0.19	0.79±0.20	**0.84**±0.18
Alarm	10	0.51±0.02*	0.83±0.26	0.51±0.06*	**0.92**±0.08
	50	0.59±0.07*	0.90±0.12	0.63±0.10*	**0.91**±0.09
Insurance	10	0.50±0.00*	0.74±0.12	0.51±0.02*	**0.79**±0.13
	50	0.55±0.04*	0.82±0.13	0.62±0.08*	**0.83**±0.12
Hailfinder	10	0.50±0.00*	0.72±0.12	0.50±0.00*	**0.73**±0.13
	50	0.50±0.00*	0.89±0.13	0.51±0.02*	**0.89**±0.13

AUC measurement, the MAP also gets good results. The average absolute difference between MAP and MPL-EC results is just 0.02, which suggest these two approaches have similar classification accuracy.

In a real-world classification task, the true BN is unknown and the specific query node is provided. In this situation, we cannot use K-L divergence measurement. And the AUC measurement is the best choice to measure the learning algorithm that returns the best classification performance.

Running time analysis As discussed in Section 4.4, the computational complexity is mainly decided by the total number of introduce exterior constraints. Here, we list the average computational time of each learning task for different learning algorithms, the details are presented in Table 4.4.

The results are shown in Table 4.4, from which we can make the following observations. (i) MLE and MAP are very efficient compared with CO and MPL-EC. (ii) For learning algorithms with constraints, the computational time increases with the BN sizes. This finding is consist with the theoretical analysis in Section 4.4. (iii) The different computational time between CO and MPL-EC is caused by different parameter estimation methods. The gradient descent method used in CO is more efficient than the DDJT inference method in MPL-EC. Specifically, according to the result, the average running time of CO is 103.2 times faster than MPL-EC.

Table 4.4: Running time (seconds) for MLE, MAP, CO and MPL-EC in 6 standard BN parameter learning problems.

Name	Data	MLE	MAP	CO	MPL-EC
Weather	10	0.0	0.0	1.3	52.0
	50	0.0	0.0	2.4	29.3
Cancer	10	0.0	0.0	1.4	74.9
	50	0.0	0.0	2.6	99.6
Asia	10	0.0	0.0	2.0	178.8
	50	0.0	0.0	4.0	157.7
Alarm	10	0.1	0.0	85.9	23791.3
	50	0.4	0.0	205.2	23355.0
Insurance	10	0.0	0.0	210.9	39019.7
	50	0.0	0.0	408.6	30426.6
Hailfinder	10	0.0	0.0	891.4	150786.6
	50	0.0	0.0	2235.3	154008.0

4.6.3 The Influence of Error Labels

As shown in above experiments, incorporating exterior constraints generated from monotonic effects can increase the learning performance. However, when such monotonic effect labels are wrongly elicited (that is usually inevitable in real-world applications), their influences on the final learning performance should be further investigated. To this end, we generate error labels for previously elicited monotonic effects. Specifically, in each BN the elicited positive effect labels are converted into negative labels, and elicited negative effect labels are converted into positive labels. The final results are measured by average K-L divergence.

As we can see from the results (Table 4.5), the performance of CO and MPL-EC are both worse than previous results that learnt with correct monotonic effect labels (Table 4.2). For example, the average K-L divergence of MPL-EC for each BN increased 0.34 compared with the previous results. Now, the MAP achieves the best learning performance, it beats other learning algorithms almost in all cases. Specifically, it outperforms the MPL-EC with an average margin of 28.2%. These results show that wrongly elicited monotonic effect labels and their generated exterior constraints could harm the learning performance of MPL-EC. To this end, in real-world applications, the elicited labels should be carefully checked before use.

Table 4.5: Results for MLE, MAP, CO and MPL-EC with error monotonic effect labels in 6 standard BN parameter learning problems.

Name	Data	MLE	MAP	CO	MPL-EC
Weather	10	0.57±0.41*	**0.27**±0.06	0.52±0.06*	0.80±0.09*
	50	**0.05**±0.03	0.08±0.02	0.39±0.03*	0.83±0.29*
Cancer	10	1.65±0.37*	**0.11**±0.04	0.54±0.45*	0.35±0.04*
	50	0.50±0.23*	**0.04**±0.02	0.30±0.25*	0.43±0.16*
Asia	10	1.93±0.73*	**0.62**±0.07	0.81±0.08*	0.97±0.05*
	50	0.86±0.30*	**0.41**±0.05	0.72±0.06*	0.86±0.10*
Alarm	10	3.94±0.19*	**0.94**±0.02	2.72±0.12	1.04±0.02*
	50	2.80±0.19*	**0.75**±0.02	2.33±0.06*	0.90±0.02*
Insurance	10	4.06±0.11*	**1.79**±0.01	2.46±0.04*	1.80±0.01*
	50	2.52±0.09*	1.40±0.02	1.85±0.05*	**1.39**±0.03
Hailfinder	10	4.53±0.03*	**0.77**±0.00	1.37±0.04*	0.81±0.00*
	50	3.40±0.06*	**0.57**±0.01	0.94±0.02*	**0.65**±0.01*

4.7 A Case Study

In this section, we continue to use the software defects prediction BN discussed in Chapter 3 to test its learning performance now with exterior constraints. Compared to the experiment settings in Chapter 3, the difference here is that the experts are only required to identify the monotonic effect labels in this BN. Thus, the elicitation efforts and expenses are greatly reduced in this setting.

Figure 4.11 represents the structure of the BN, the signs on the edges indicate whether the associate monotonic effects are positive or negative. These monotonic effects are elicited from real expert judgments, e.g., as design process quality (DQ) goes from '*low*' to '*high*', the defects inserted (DI) go from '*high*' to '*low*', this encodes a negative monotonic effect.

Figure 4.12 shows the learning results. As we can see, the MPL-EC outperforms all other algorithms in every scenario. Compared with the state-of-art CO algorithm, our MPL-EC significantly improves the parameter learning performance, i.e., the MPL-EC outperforms the CO in all training sample sizes, with an overall 47.1% K-L divergence reduction.

4.8 Summary

As discussed in previous chapters, when data is scarce, purely data driven BN learning is inaccurate. Whereas Chapter 3 introduced the idea of interior constraints and the

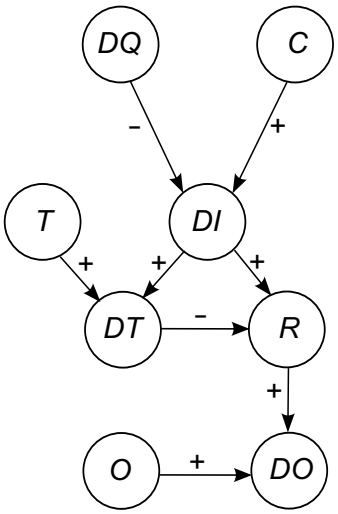


Figure 4.11: The monotonic effect labels in the software defects prediction BN (6 positive monotonic effects and 2 negative monotonic effects).

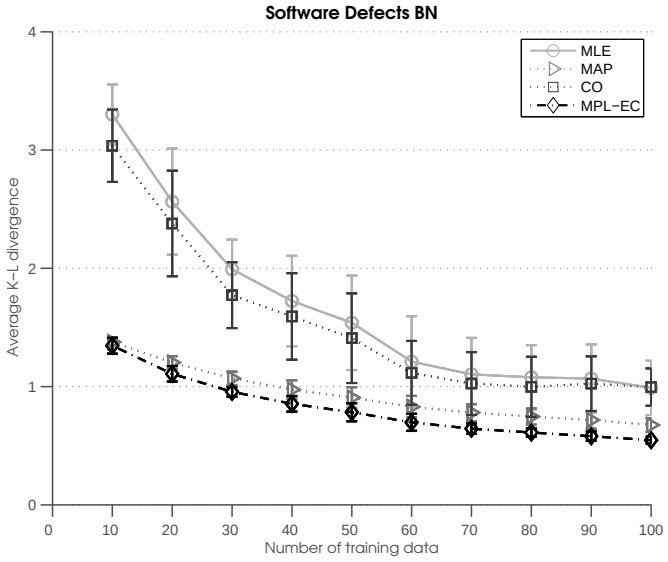


Figure 4.12: Learning results of MLE, MAP, CO and MPL-EC for software defects prediction BN with different training data sample sizes.

MPL-C model to address this problem, in this chapter the MPL-EC method presented tackles this problem by leveraging a set of exterior constraints elicited from experts. This method is an auxiliary BN, which encodes all the information (i.e., data observations, parameters we wish to learn, and exterior constraints encoded in monotonic effects) in parameter learning. By using the generic synergy model and converting the parameter learning problem into a Bayesian inference problem, we are able to perform robust and effective parameter learning even with heterogeneous monotonic effects and

zero data observations in some cases. This MPL-EC method applies with categorical variables, and is robust to very limited training data sizes.

Experiments with 6 standard BNs show that MPL-EC consistently outperforms the MLE and MAP algorithms and former constrained parameter learning algorithm – CO with respect to the K-L divergence measurement. With randomly selected query variables, MPL-EC and MAP achieve similar AUC results, and greatly outperform the results of MLE and CO. Moreover, the time complexity analysis suggests that large number of monotonic effects would introduce great number of exterior constraints, and require more computational cost in both CO and MPL-EC. The experimental results show that MPL-EC is very slow compared to all other algorithms. Finally, experiments with a real-world software defects prediction BN show the practical value of this method.

So far we have assumed expert judgment is available to enhance parameter learning. However, if there is no expert judgment in the problem domain, both MPL-C and MPL-EC would fail to learn a good model with limited data. Thus, we need to investigate other sources of available information to help the parameter learning. We address this in the following two chapters.

Chapter 5

Parameter Transfer Learning

When no expert judgments are available, learning BNs from scarce data is a major challenge in real-world applications. Transfer learning techniques attempt to address this by leveraging data from different, but related problems. The idea behind transfer learning is to improve the accuracy of a target BN by making use of one or more related source BNs. For example, the target BN may be a model for diagnosis of a particular disease based on limited data in one district or country. If there are other (source) BNs with similar variables and objectives but from a different district or country, then it make senses to exploit such models to improve the accuracy of the target BN. Transfer learning does this by providing methods for both determining suitability of the data in the source and its transfer to the target.

In this chapter we introduce the first two-step general-purpose BN parameter transfer learning framework to reason about both network and fragment relatedness. Our framework addresses (i) how to find the most relevant source network and network fragments to transfer, and (ii) how to fuse source and target parameters in a robust way. In addition to improving target task performance, explicit reasoning allows us to diagnose network and sub-graph relatedness across BNs, even if latent variables are present, or if their state space is heterogeneous. This is important in some applications where relatedness itself is an output of interest. Experimental results demonstrate the superiority of our framework at limited data and various source relevance levels compared to single task learning and other state-of-the-art parameter transfer methods.

5.1 Limitations

The obvious practical limitation of transfer learning – which limits the applicability of all work in this area including this work – is that the relatedness is never truly known. The necessary assumptions to overcome this introduce inevitable bias into the results. In this thesis we assume there is at least one source domain or a subset of a domain that is sampled from similar distributions as the target, and that this can be transferred to help learn the target BN parameters. However, determining relatedness in a data driven way means there is an inevitable confirmation bias in the sense that the source BNs most likely to be selected are those that most closely match the current target estimate. This limits the extent that the source can 'change' the target when it is more 'correct' than the current noisy target estimates.

If the chosen source and the target are not sampled from similar distributions, directly applying parameters learned in another domain may be impossible or result in negative transfer: the underlying tasks may have major quantitative or qualitative differences (e.g., care procedures vary across hospitals). This limits the effectiveness of existing methods such as that in (Luis et al., 2010) – referred to as CPTAgg in this thesis. Our framework will address this by robustly measuring relatedness in a piecewise way. Before giving the detail of our framework, we firstly introduce the formal definitions used in parameter transfer learning.

5.2 Formal Definition of Parameter Transfer Learning

In Chapter 2, we have defined the BN parameter learning setting, which has data D combined with V and G to form the problem domain $\mathscr{D} = \{V, G, D\}$. Within a domain $\mathscr{D}$, the goal of parameter learning is to determine parameters for all $p(X_i|\pi_i)$. Given data D, the estimation of NPT parameters θ is conventionally solved by the MLE, $\hat{\theta} = \arg\max_\theta \log p(D|\theta)$. We denote this setting Single Task Learning (STL).

In parameter transfer learning, we have one target domain $\mathscr{D}^t$, and a set of sources $\{\mathscr{D}^s\}_{s=1}^S$, $S \geq 1$. The target domain and each source domain have training data,

$D^t = \{d_1^t, d_2^t, \ldots, d_N^t\}$ and $D^s = \{d_1^s, d_2^s, \ldots, d_N^s\}$.

In most cases the target domain is relatively scarce: $0 < N^t \ll N^s$. Following the definition of transfer learning in (Pan and Yang, 2010), we define the BN parameter transfer learning:

Definition 5.2.1. ***BN parameter transfer learning*** Given a set of source domains $\{\mathscr{D}^s\}$ and a target domain $\mathscr{D}^t$, BN parameter transfer learning aims to improve the parameter learning accuracy of the BN in $\mathscr{D}^t$ using the knowledge in $\{\mathscr{D}^s\}$.

This task corresponds to the problem of estimating the target domain NPTs θ^t given all the available domains:

$$\hat{\theta}^t = \arg\max_{\theta^t} p(\theta^t|\mathscr{D}^t,\{\mathscr{D}^s\}) \tag{5.1}$$

If the networks correspond ($V^t = V^s, G^t = G^s$) and relatedness is assumed, then this is trivially MLE with count-aggregation. But in general $\mathscr{D}^s \neq \mathscr{D}^t$ due to different training data sets with different statistics and thus varying relatedness; and potentially heterogeneous state spaces V. We consider the case where dimensions/variables in each domain do not correspond, i.e. $V_s \neq V_t$. They may be disjoint $V_s \cap V_t = \emptyset$, or partially overlap $V_s \cap V_t \neq \emptyset$. However any correspondence between them is not assumed given (variables names are not used). Then this parameter transfer learning problem is much harder.

In order to learn a target domain $\mathscr{D}^t$ leveraging sources $\{\mathscr{D}^s\}$ with *piecewise relatedness*[1], or heterogeneity $V^t \neq V^s$ and $G^t \neq G^s$, we transfer at the level of BN *fragments*.

Definition 5.2.2. ***BN fragments*** A BN of domain $\mathscr{D}$ can be divided into a set of sub-networks (denoted *fragments*) $\mathscr{D} = \{\mathscr{D}_i\}$ by considering the graph G. Each fragment $\mathscr{D}_i = \{V_i, G_i, D_i\}$ is a single root node or a node with its direct parents in the original BN, and encodes a single NPT from the original BN. The number of fragments is the number of variables in the original BN.

[1]Relatedness is typically computed at domain or instance level granularity. In contrast, here we consider that relevance may vary *within-domain* – such that different subsets of features/variables may be relevant to different source domains.

5.3 A Two-step General-Purpose Parameter Transfer Learning Framework

To realize flexible BN parameter transfer, the target domain and source domains are all broken into fragments $\mathscr{D}^t = \{\mathscr{D}^t_j\}$, $\{\mathscr{D}^s\} = \{\{\mathscr{D}^s_k\}\}$. Assuming for now no latent variables in the target domain, then each fragment j can be learned independently $\hat{\theta}^t_j = \arg\max_{\theta^t_j} p(\theta^t_j|\mathscr{D}^t_j, \{\{\mathscr{D}^s_k\}\})$. To leverage the bag of source domain fragments $\{\{\mathscr{D}^s_k\}\}$ in learning each θ^t_j, we consider each source fragment $\mathscr{D}^s_k$ as potentially relevant. Then we propose a two-step general-purpose parameter transfer learning framework. Specifically, in the first step, for each target fragment, every source fragment is evaluated for relatedness and the best fragment mapping is chosen. Once the best source fragment is chosen for each target, a domain/network-level relatedness prior is re-estimated by summing the relatedness of its fragments to the target. We refer to this step as the ***fitness step***. In the second step, the knowledge from the best source fragment for each target is then fused according to its estimated relatedness. We refer to this step as the ***fusion step***. To realize this strategy, four issues must be addressed:

(1) which source fragments are transferable?

(2) how to deal with variable name mapping?

(3) how to quantify the relatedness of each transferrable source fragment in order to find the best one?

(4) how to fuse the chosen source fragment?

We next address these issues in the context of a deeper examination of the two steps.

5.3.1 The Fitness Step

The key processes of the fitness step are described in Figure 5.1. Next, we will discuss their details in turn.

Fragment Compatibility For a target fragment j and putative source fragment k with discrete and finite state space[2], we say they are *compatible* if they have the same

[2]For fragments with continuous state spaces, we say they are *compatible* if they have the same structure.

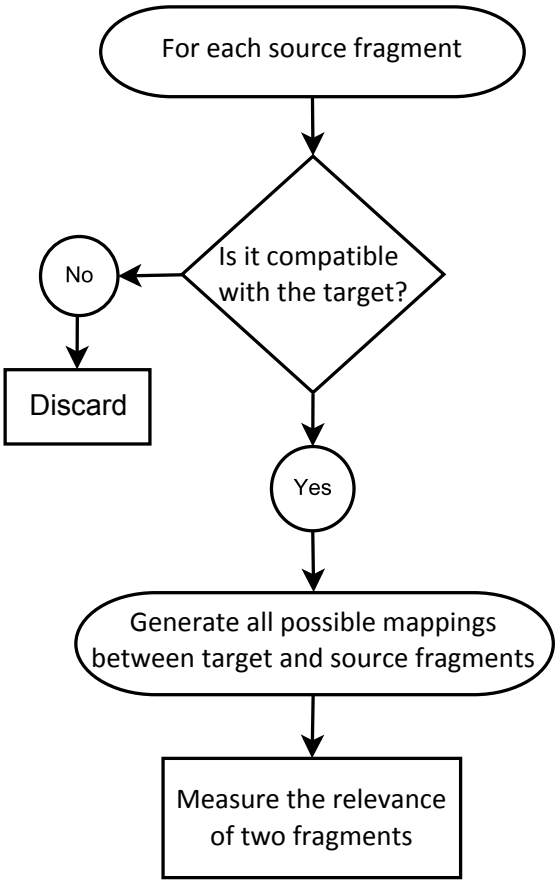


Figure 5.1: The flowchart of the fitness step of the parameter transfer learning framework.

structure and state space. That is, the same number of states and parents' states[3], so

$$compatible(\mathscr{D}_j^t, \mathscr{D}_k^s) = \begin{cases} 1 & if\ G_j^t = G_k^s \ \&\&\ \dim(\theta_j^t) = \dim(\theta_k^s) \\ 0 & \text{otherwise} \end{cases} \tag{5.2}$$

This definition of compatibility could be further relaxed quite straightforwardly (e.g., allowing target states to aggregate multiple source states) at the expense of additional computational cost. For example, if the target variable contains two states (*true* and *false*), and one source variable contains three states (*low*, *medium* and *high*), we can try multiple aggregations of the source states to generate three mappings to the target:

(i) mapping $low - true$ and $\{medium, high\} - false$,

(ii) mapping $medium - true$ and $\{low, high\} - false$, and

[3]This assumes that the number of parameters is proportional to the number of rows in the node probability table, and no parametric dimension reduction is used.

(iii) mapping $high-true$ and $\{low, medium\}-false$.

However, while relaxing the condition of compatibility would improve the range of situations where transfer can be exploited, it would also increase the cost of the algorithm by increasing the number of allowed permutations, as well as decreasing robustness to negative transfer (by potentially allowing more 'false positive' transfers from irrelevant sources). This is an example of pervasive trade-off between maximum exploitable transfer and robustness to negative transfer (Torrey and Shavlik, 2009).

Fragment Permutation Mapping For two fragments j and k determined to be compatible, we still do not know the mapping between variable names. For example, if j has parents $[a,b]$ and k has parents $[d,c]$, the correspondence could be $a-d, b-c$ or $b-d, a-c$. The function $permutations(G^t_j, G^s_k)$ returns an exhaustive list of possible mappings P_m that map states of k to states of j.

Here we provide an illustrative example of fragment-based parameter transfer: the target is a three node BN shown in the left part of Figure 5.2 (a), and the source is a eight node BN shown in the right part of Figure 5.2 (a). In Figure 5.2 (b), there are two source fragments ($\{T^s, L^s, E^s\}$ and $\{E^s, B^s, S^s\}$) which are *compatible* with the target fragment. Thus, there are four *permutations* of compatible source fragments (assuming binary parent nodes). All four of these options are then evaluated for *fitness*, and the best fragment and permutation is picked (shown with dashed triangle in Figure 5.2 (b)).

Relevance Measurement To measure the relevance/relatedness between compatible target and source fragments $\mathscr{D}^t_j$ and $\mathscr{D}^s_k$, we introduce a function $fitness(\mathscr{D}^t_j, \mathscr{D}^s_k, p(H^s))$, where $p(H^s)$ is a domain-level relatedness prior[4]. In this section, for notational simplicity we will use t and s to represent the j-th target and k-th source domain fragments under consideration.

A systematic and robust way to compare source and target fragments for relevance is to compute the probability that the source and target data share a common NPT (hypothesis H_1) versus having distinct NPTs (hypothesis H_0). Consistent with the simplification of fragment notation, here H^s_1 only refers to the dependent hypothesis between $\mathscr{D}^t_j$ and $\mathscr{D}^s_k$. The Bayes model comparison for hypotheses $H \in \{H^s_1, H^s_0\}$ is:

[4]We consider a discrete random variable indexing the related source s among S possible sources. So $p(H^s)$ is a S-dimensional *Multinomial* distribution encoding the relatedness prior.

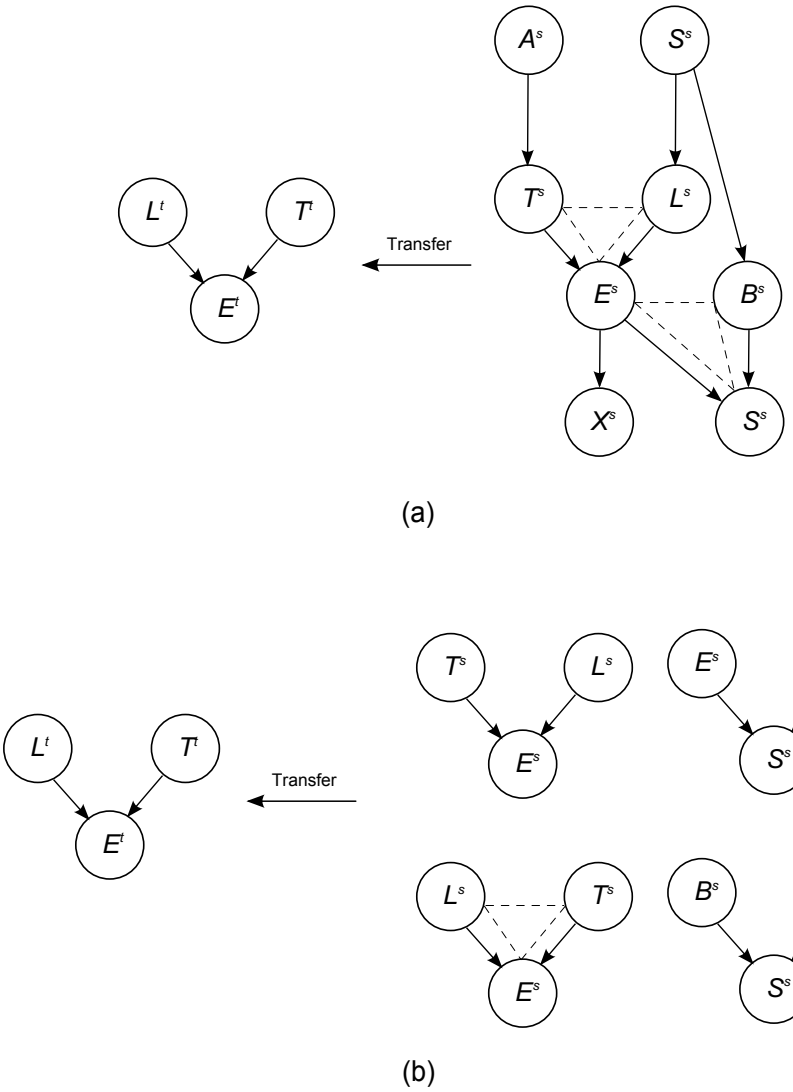


Figure 5.2: A simple example to show the fragment compatibility measurement, and the permutations of all possible parental nodes in a BN fragment. (a) The dashed triangle represents source fragments $\{T^s, L^s, E^s\}$ and $\{E^s, B^s, S^s\}$, which are *compatible* with the target fragment. (b) All the *permutations* of compatible source fragment, and the most fit one $\{L^s, T^s, E^s\}$.

$$
\begin{aligned}
p(H_1^s|D^s, D^t) &\propto \int p(D^t|\theta)p(\theta|D^s, H_1^s)p(H_1^s)d\theta, \\
p(H_0^s|D^s, D^t) &\propto \int p(D^t|\theta^t)p(\theta^t|H_0^s)p(H_0^s)d\theta^t.
\end{aligned}
\tag{5.3}
$$

where we have made the following conditional independence assumptions: $D^s \perp H_1^s$, $D^t \perp \{D^s, H_1^s\}|\theta$ and $\theta^t \perp D^s|H_0^s$.

For discrete likelihoods $p(D|\theta)$ and Dirichlet priors $p(\theta|H)$, integrating over the unknown NPTs θ, the required marginal likelihood is the Dirichlet compound multinomial (DCM) or multi-variate Polya distribution:

$$p(D^t|D^s,H_1^s) = \frac{\Gamma(A^{X^s})}{\Gamma(N^{X^t}+A^{X^s})}\prod_{c=1}^{C}\frac{\Gamma(n_c^{X^t}+\alpha_c^{X^s})}{\Gamma(\alpha_c^{X^s})} \tag{5.4}$$

where $c = 1,\ldots,C$ index variable states, $n_c^{X^t}$ is the number of observations of the c-th target parameter value in data D^t, and $N^{X^t} = \sum_c n_c^{X^t}$; $\alpha_c^{X^s}$ indicates the aggregate counts from the source domain and distribution prior, and $A^{X^s} = \sum_c \alpha_c^{X^s}$. Assuming that transfer is equally likely a priori within source domains, but not across source domains, we set the prior for the shared-parameter hypothesis to the estimated domain similarity $p(H_1^s)$ (Algorithm 5.1) and independent-parameter hypothesis prior to its complement, $p(H_0^s) = 1 - p(H_1^s)$.

Maximal $fitness(\cdot)$ is achieved when the target data are most likely to share the same generating distribution as the source data. We call our proposed fitness function **BMC**. The presentation thus far addresses discrete data with Dirichlet conjugate priors. Analogous computations can be derived for continuous data with Gaussian likelihood with Normal-Inverse-Gamma conjugate priors:

$$p(D^t|D^s,H_1^s) = \prod_{i=1}^{N}\left(\frac{1}{\sqrt{\pi}}\frac{\Gamma\left(\frac{2\alpha_m+1}{2}\right)}{\Gamma\left(\frac{2\alpha_m}{2}\right)}\sqrt{\frac{\Lambda}{2\alpha_m}}\left(1+\frac{\Lambda(d_i^t-\mu_m)^2}{2\alpha_m}\right)^{-\left(\frac{2\alpha_m+1}{2}\right)}\right) \tag{5.5}$$

where $\Lambda = \frac{\alpha_m k_m}{\beta_m(k_m+1)}$, the hyperparameters μ_m, k_m, α_m and β_m are updated based on the source data D_k^s, which contains M samples with center at $\bar{d}^s$:

$$\begin{cases} \mu_m = \frac{k_0\mu_0+M\bar{d}^s}{k_0+M} \\ k_m = k_0+M \\ \alpha_m = \alpha_0+\frac{M}{2} \\ \beta_m = \beta_0+\frac{1}{2}\sum_{i=1}^{M}(d_i^s-\bar{d}^s)^2+\frac{k_0M(\bar{d}^s-\mu_0)^2}{2(k_0+M)} \end{cases} \tag{5.6}$$

Transfer Prior The final outstanding component of BMC is how to define the transfer prior $p(H^s)$. We assume that transfer is equally likely a priori within a given source

domain, but that different source domains may have different prior relatedness. Thus, we set the transfer prior for a particular fragment pair to the prior for the corresponding source network, i.e., $p(H^s_{jk1}) = p(H^s)$. The fragment transfer prior $p(H^s_{jk})$ is then normalised as $p(H^s_{jk0}) = 1 - p(H^s_{jk1})$.

5.3.2 The Fusion Step

Once the best source fragment $\mathscr{D}^s_k$ is found for a given target fragment $\mathscr{D}^t_j$, the next challenge is how to optimally fuse them. Our solution (denoted **BMA**) is to infer the target NPT, integrating over uncertainty about whether the selected source fragment is indeed relevant or not (i.e., if they share parameters or not, corresponding to hypotheses H_1 and H_0 in Section 5.3.1).

We perform Bayesian model averaging, summing over these possibilities. Specifically, we ask $p(\theta^t|D^t,D^s) = \sum_H p(\theta^t,H|D^t,D^s)$ which turns out to be:

$$\begin{aligned} p(\theta^t|D^t,D^s) &= p(H_1|D^t,D^s)Dir(\theta;\alpha+N^{X^t}+N^{X^s}) \\ &\quad +p(H_0|D^t,D^s)Dir(\theta;\alpha+N^{X^t}) \end{aligned} \tag{5.7}$$

where $p(H_1|D^t,D^s)$ and $p(H_0|D^t,D^s)$ come from equation (5.3). This means the strength of fusion is automatically calibrated by the estimated relevance. Since there is no closed form solution for the sum of Dirichlets, we approximate equation (5.7) by moment matching. For conditional Gaussian nodes, the weighted sum is also approximated by moment matching.

Moment matching (also known as Assumed Density Filtering (ADF)) is to approximate a mixture such as equation (5.7) by a single distribution whose mean and variance is set to the mean and variance of the weighted sum. The estimated relatedness provides the weights $w_1 = p(H_1|D^t,D^s)$, $w_0 = p(H_0|D^t,D^s)$. Assuming the posterior mean and variance of the parameters in the related and unrelated condition are u_1, v_1 and u_0, v_0 respectively. Then the approximate posterior mean is $u = w_1u_1 + w_0u_0$, and variance is $v = w_1(v_1 + (u_1 - u)^2 + w_0(u_0 - u)^2$ (Murphy, 2012). For Gaussian distributions we can use this directly. For Dirichlet distributions with parameter vector α, the variance parameter $v = 1/\sum\alpha$, and the mean parameter vector is $u = v\alpha$.

5.3.3 The Algorithm

The detail of our two-step general-purpose parameter transfer learning framework is given in Algorithm 5.1. Each target fragment is compared to all permutations of compatible source fragments and evaluated for relevance using BMC fitness. The most relevant source fragment and permutation is assigned to each target fragment. The network-level relevance prior is re-estimated based on aggregating the inferred fragment relevance for that source: $p(H^s) \propto \sum_{jk} p(H^s_{jk}|\mathscr{D}^t_j, \mathscr{D}^s_k)$. This way of updating the source network prior reflects the inductive bias that fragment should be transferred from fewer distinct sources, or that a source network that has already produced many relevant fragments is more likely to produce further relevant fragments and should be preferred.

Finally, the most relevant source fragment for each target is fused using BMA. We refer to our BMC+BMA framework as BNPTL (Bayesian network parameter transfer learning) in this thesis. If there are missing or hidden data in the target domain, we start by running the standard EM algorithm in the target domain, to infer the states of each hidden variable. We use these expected counts to fill in D^t when applying our framework.

Properties Our framework has a few favorable properties worth noting:

- If there is *no* related source fragment, then the most related source fragment will have estimated relatedness approaching zero, and no transfer is performed ($p(H_1|D^t, D^s) \approx 0$ in equation (5.7)). The framework is thus robust to irrelevant sources (as explored in Subsection 5.4.6 and 5.4.7).

- Although we rely on an EM procedure to estimate fragment and source relatedness, starting from a uniform prior $p(H^s)$, our algorithm is deterministic and we use only one run to get results.

- Explicitly reasoning about both fragment and network level relatedness allows the exploitation of heterogeneous relevance both within and across source domains.

INPUT : Target network $\mathscr{D}^t$, Sources $\{\mathscr{D}^s\}$
OUTPUT: $\theta^t = \left\{\theta^t_j\right\}$ and $p(H^s)$

Initialize the domain-level relatedness $p(H^s)$ (uniform);
repeat
  for *each target fragment* j **do**
    for *each source network* s *and fragment* k **do**
      if $compatible(\mathscr{D}^t_j, \mathscr{D}^s_k)$ **then**
        $P = permutations(G^t_j, G^s_k)$;
        for *permutation* $m = 1$ **to** M **do**
          measure relatedness:
          $fitness(\mathscr{D}^t_j, P^{sk}_m(\mathscr{D}^s_k), p(H^s)) = p(H^s_{jk1}|D^t_j, P^{sk}_m(D^s_k))$;
        end
      end
    end
  end
  for *each source network* s **do**
    Re-estimate network relevance: $p(H^s) \propto \sum_{jk} p(H^s_{jk}|D^t_j, D^s_k)$;
  end
until *convergence* $\Delta p(H^s) \leq 1 \times 10^{-4}$;
for *each target fragment* j **do**
  Find the best source and permutation:
  $k', s', m' = \arg\max_{k,s,m} p(H^s_{jk1}|D^t_j, P^{sk}_m(D^s_k))$;
  $\theta^t_j = fusion(\mathscr{D}^t_j, P^{s'k'}_{m'}(\mathscr{D}^{s'}_{k'}))$;
end
return $\theta^t = \left\{\theta^t_j\right\}$ *and* $p(H^s)$

Algorithm 5.1: A two-step general-purpose BN parameter transfer learning framework (BNPTL)

5.4 Experiments

In this section, we compare against existing strategies for estimating relatedness and fusing source and target data. For relatedness estimation, we introduce two alternatives to **BMC**:

- **Likelihood**: The similarity between the fragments is the log-likelihood of the target data under the ML source parameters $\hat{\theta}^s$, $\sum_l \log p(d^t_l|\hat{\theta}^s)$.

- **MatchCPT**: The dis-similarity between the fragments is the K-L divergence (KLD) between their ML parameter estimates $KL(\hat{\theta}^t, \hat{\theta}^s)$ (Dai et al., 2007; Selen and Jaime, 2011; Luis et al., 2010).

For fusing source and target knowledge, we introduce two competitors to our **BMA**:

- **Basic:** Use the estimated source parameter directly $\hat{\theta}^s_j$. A reasonable strategy if relevance is perfect and the source data volume is high, but does not exploit target data and it is not robust to imperfect relevance.

- **Aggregation**: A weighted sum reflecting the relative volume of source and target data (equation (12) in (Luis et al., 2010)), it exploits both source and target data, but is less robust than BMC to varying relevance.

Neither Basic nor Aggregation is robust to varying relevance across and within sources (they do not reflect the goodness of fit between source and target), or situations in which no source node at all is relevant (e.g., given partial overlap of the source and target domain).

5.4.1 Overview of Relatedness Contexts

Before presenting the experimental results, we first highlight the variety of possible network-relatedness contexts that may occur. Of these, different relatedness scenarios may be appropriate depending on the particular application area.

- **Structure and Variable Correspondence** In some applications, the source and target networks may be known to correspond in structure, share the same variable names, or have provided variable name mappings. In this case the only ambiguity in transfer is which of multiple potential source *networks* is the most relevant to a target. Alternatively, structure/variable name correspondence may not be given. In this case there is also ambiguity about which *fragment* within each source is relevant to a particular target NPT.

- **Cross-network relevance heterogeneity** There may be multiple potential source networks, some of which may be relevant and others irrelevant. The most relevant source should be identified for transfer, and irrelevant sources ignored.

- **Continuous versus discontinuous relevance** When there are multiple potential source networks, it may be that relevance to the target varies continuously (e.g., if each network represents a slightly different segment of demographic of

the population), or it may be that across all the sources some are fully relevant and others totally irrelevant. In the latter case it is particularly important not to select an irrelevant source, as significant negative transfer is then likely.

- **Piecewise Relevance** Relevance may vary piecewise within networks as well as across networks. Consider a target network with two sub-graphs A and B: A may be relevant to a fragment in source 1, and B may be relevant to a fragment in source 2. For example, in the case of networks for hospital decision support, different hospitals may share different subsets of procedures – so their BNs may correspond in a piecewise way only. A target hospital network may then ideally draw from multiple sources. Note that this may happen either because (i) sub-graphs in the target are structurally compatible with different sub-graphs in the multiple sources (which need not be structurally equivalent to each other), or (ii) in terms of quantitative NPT fit, fragments in the target may each be better fit to different sources.

Our BNPTL framework aims to be robust to all the identified variations in network relatedness. In the following experiments, we will evaluate BN transfer in each of these cases.

5.4.2 Transfer with Known Correspondences

In this section, we first evaluate transfer in the simplest setting, where structure/variable name correspondence is assumed to be given. This setting is the same as (Luis et al., 2010): the transfer only happens between target/source nodes with the same node index $X_i^t = X_i^s$, where $X_i^t \in V_t$, $X_i^s \in V_s$ and $V_t = V_s$, $G_t = G_s$. (In our framework this is easily modelled by providing the prior $p(H_{jk1}^s) = 0$, and hence $p(H_{jk0}^s) = 1$, for non-corresponding pairs $j \neq k$.) This setting has the least risk of negative transfer, because there is no chance of transferring from an irrelevant source NPT.

We continue using six standard BNs (Table 3.4) to compare our approach (BMC fitness with BMA (BNPTL)) to the state-of-art (MatchCPT fitness with Aggregation fusion (CPTAgg) (Luis et al., 2010)). In this case we use "soft noise" to simulate continuously varying relatedness among a set of sources. For each reference BN three sets of 200, 300 and 400 samples respectively are drawn. These sample sets are used to learn three sources of increasing network-level relatedness to the true BN. Subsequently, 100

samples of each source copy are drawn and used as the actual source data. Because node correspondences are known in this experiment, another baseline is simply to aggregate all target and source data. This method is referred to as ALL, and also will be compared. Results are quantified by average K-L divergence between estimated and true NPTs. In each experiment we run 10 trials with random data samples and report the mean and standard deviation of the K-L divergence.

Table 5.1: Performance (known correspondences) of STL, ALL and transfer learning methods: CPTAgg, BNPTL^{np} and BNPTL.

Name	STL	ALL	CPTAgg	BNPTL^{np}	BNPTL
Weather	0.02±0.02*	**0.01** ±0.00	**0.01** ±0.00	**0.01** ±0.00	**0.01** ±0.00
Cancer	0.33±0.31*	**0.01** ±0.00	0.12±0.09*	0.10±0.07*	0.10±0.05*
Asia	0.85±0.18*	0.36±0.04	0.68±0.27	0.30 ±0.12	**0.24** ±0.14
Insurance	1.82±0.16*	1.05±0.09*	1.47±0.17*	0.77 ±0.05	**0.76** ±0.04
Alarm	2.43±0.15*	1.70±0.10*	2.19±0.13*	0.64 ±0.02	**0.63** ±0.02
Hailfinder	2.85±0.03*	1.98±0.02*	2.44±0.04*	**0.97** ±0.07	**0.97** ±0.04
Average	1.38±0.14	0.85±0.04	1.15±0.12	0.47 ±0.05	**0.45** ±0.05

The results are presented in Table 5.1, with the best result in bold, and statistically significant improvements of the best result over competitors indicated with asterisks * ($p \leq 0.05$). Compared with CPTAgg, BNPTL achieves 60.9% average reduction of K-L divergence compared to the ground truth. These results verify the greater effectiveness of BNPTL even in the known correspondence setting, where the assumptions of CPTAgg are not violated. To demonstrate the value of our network-level relevance prior $p(H^s)$, we also evaluate our framework without this prior (denoted BNPTL^{np}). The comparison between BNPTL and BNPTL^{np} demonstrates that the network-level relevance does indeed improve transfer performance. In this case it helps the model to focus on the higher quality/more relevant 400-sample source domain: even if for a particular fragment a less relevant source domain may have seemed better from a local perspective.

The ALL baseline also achieves good results in the Cancer and Weather networks. We attribute this to these being smaller BNs (nodes ≤ 5), so all the source parameters are reasonably well constrained by the source samples used to learn them, and aggregating them all is beneficial. However, in large BNs with more parameters, the difference between the 200 and 400 sample source networks becomes more significant, and it becomes important to select a good source instead of aggregating everything including the

noisier less related sources. In real-world settings, we may not have node/structure correspondence. Thus we do not assume this information is available in all the following sections.

5.4.3 Dependence on Target Network Data Sparsity

In this section, we explore the performance for varying number of target samples, focusing on the Asia network. Here the target and source domain are both generated from the Asia network, and the relatedness of the source domain varies (soft noise). For relatedness, we consider 3 conditions for the source domains: (i) one 200-sample source Asia network, (ii) one 200-sample source and one 300-sample source Asia network, and (iii) one 200-sample source, one 300-sample source, and one 400-sample source Asia network. This results in a varying number of source fragments (8, 16 and 24) in each setting. The latter cases potentially contain stronger cues for transfer – if a good decision is made about which source network to transfer from. To unpack the effectiveness of our contributions, we investigate all combinations for different fitness methods and fusion methods under these settings.

In each sub chart of Figure 5.3, the x-axis denotes the number of target domain training instances, and the y-axis denotes the average K-L divergence between estimated and true parameter values. The blue line represents standard MLE learning, green denotes transfer by MatchCPT fitness, purple shows transfer with likelihood fitness, and red line the results using our BMC fitness function. The columns represent Basic (source only), Aggregation and BMA fusion. Comparing rows shows the performance of all transfer methods (except MLE) improves with more source fragments. Furthermore, algorithms with our BMC fitness function (red) achieve the best results in almost all situations. Even the simple basic fusion method gets reasonable learning results (< 0.50) using the BMC fitness function to choose among the 24 source fragments. Also, our BMA fusion (right column, Figure 5.3 (c,f,i)) significantly outperforms other fusion methods. For instance, when there are 8 source fragments (top row, Figure 5.3 (a,b,c)), the average performance of BMC fitness function in BMA fusion increased 41.4% and 43.0% compared with the same fitness function in Basic fusion and Aggregation fusion settings. Although these margins decrease with increasing source fragments, our BNPTL (BMC+BMA) is generally best.

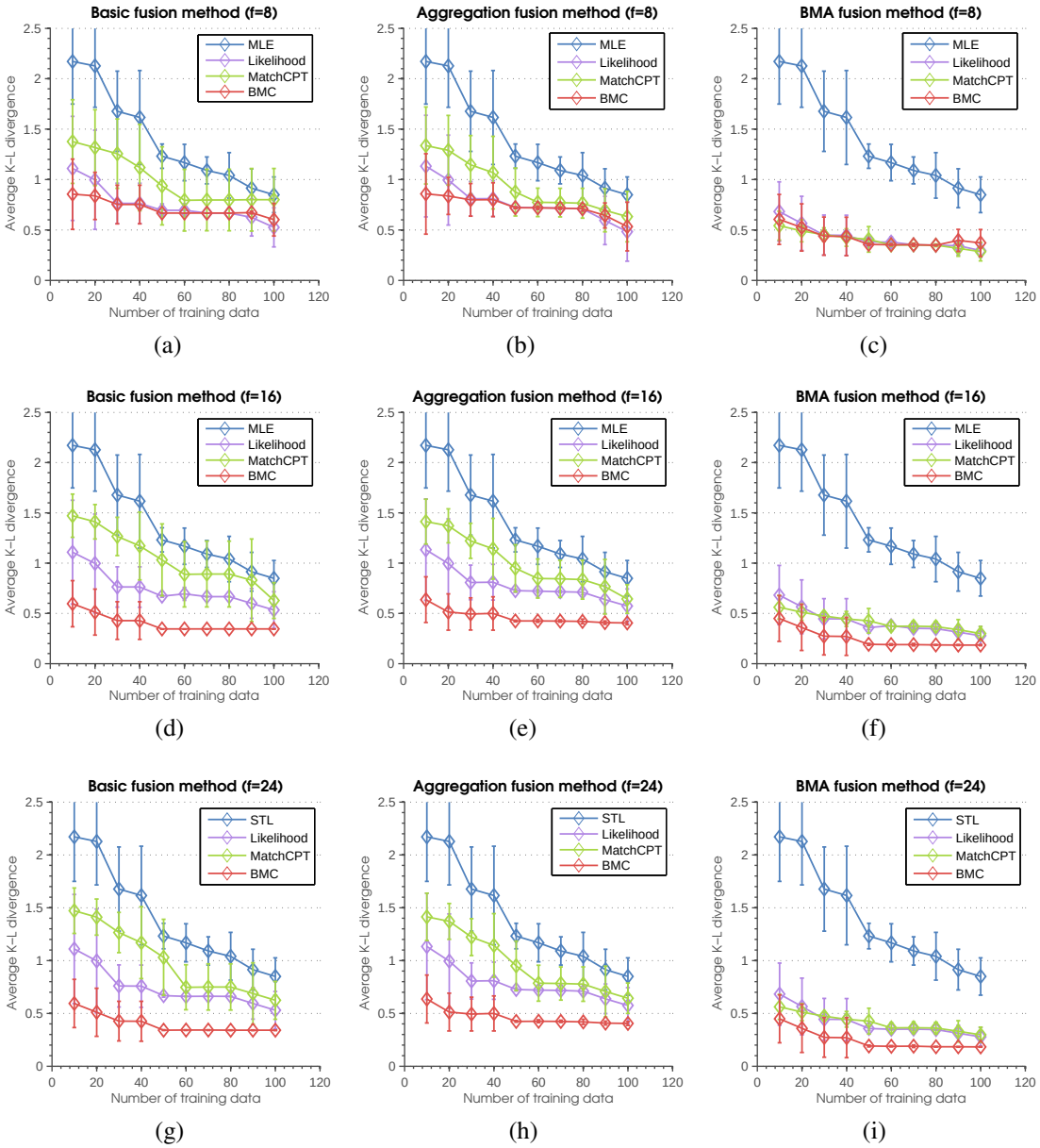


Figure 5.3: Transfer performance of varying target data volume and source relatedness (soft noise). Top, middle and bottom rows: transfer learning of 8, 16 and 24 source fragments respectively. Columns: Basic, Aggregation and BMA fusion.

5.4.4 Illustration of Network and Fragment Relatedness Estimation

To provide insight into how network and fragment relatedness is measured in BNPTL, we continue to use the Asia network and its three sources (soft noise).

Network Relatedness Figure 5.4 shows the estimated relatedness prior $p(H^s)$ for each source s over EM iterations. As we can see the network-level relatedness converges after about 10 iterations, with the relatedness estimates being in order of the

actual source relevance.

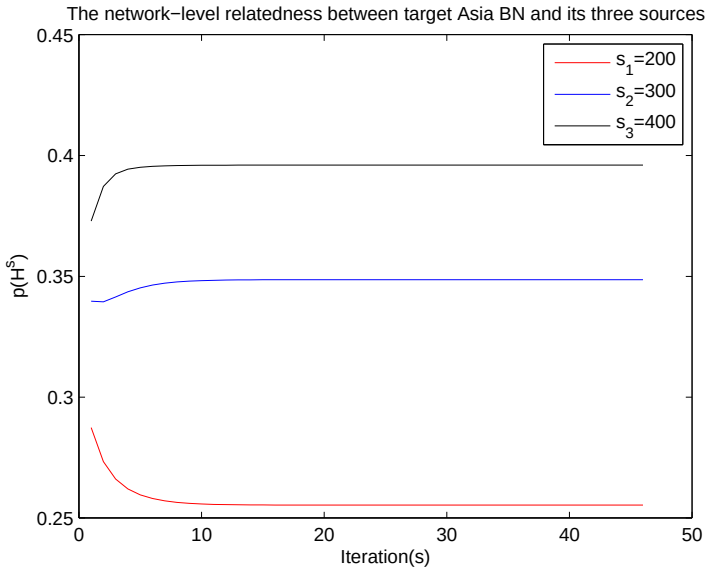
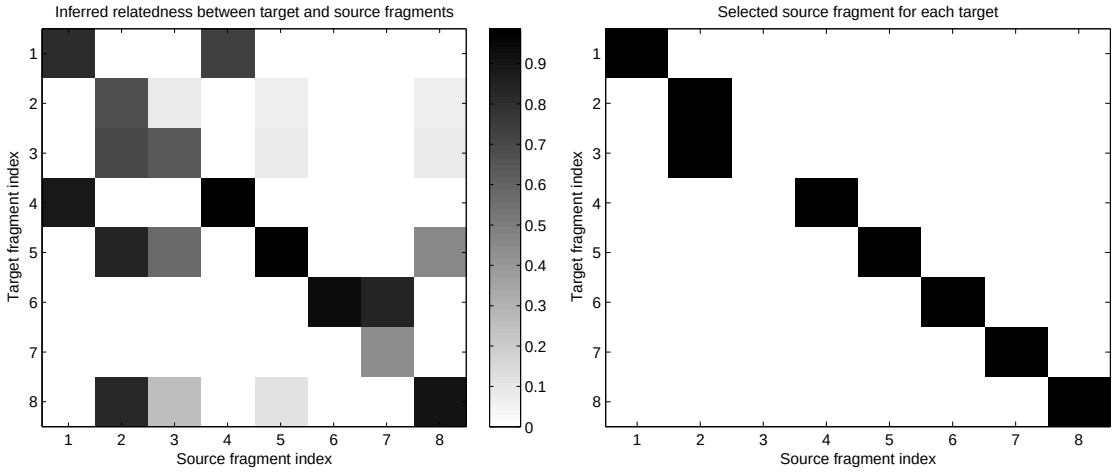


Figure 5.4: The estimated network relatedness $p(H^s)$ between target Asia BN and its three source copies of varying quality/relatedness.



(a) Inferred probability of fragment relatedness **(b)** Final selected fragment

Figure 5.5: The inferred fragment relatedness and final selected fragment in Asia BN.

Fragment Relatedness To visualize the inferred fragment relatedness, we record the estimated relatedness between every fragment in the target and every fragment in source 3 of the Asia network. This is plotted as a heat map in Figure 5.5(a), where the y-axis denotes the index of the target fragment, and x-axis denotes the index of the source fragment. Darker color indicates higher estimated relatedness $p(H^s_{jk1}|\mathscr{D}^t_j, \mathscr{D}^s_k)$ between two fragments j and k. Some incompatible source fragments have zero relatedness automatically. For each target fragment, the most related (darkest) source fragment is

selected for BMA fusion. Although there is some uncertainty in the estimated relatedness (more than one dark cell per row), overall all but one target fragment selected the correct corresponding source fragment (Figure 5.5(b).

5.4.5 Robustness to Hidden Variables

Table 5.2: Performance (unknown correspondences and hidden variables) of STL and transfer learning methods: CPTAgg and BNPTL.

Name	Hidden Vars	STL	CPTAgg	BNPTL
Weather	None	0.03±0.02	**0.02** ±0.02	**0.02** ±0.02
	1	0.55±0.07*	**0.41** ±0.00	0.45±0.01*
	2	0.59±0.00*	**0.45** ±0.01	0.49±0.01*
Cancer	None	0.33±0.31	0.14±0.09	**0.09** ±0.08
	1	0.33±0.28	0.12±0.09	**0.09** ±0.09
	2	0.39±0.27	0.20±0.08	**0.15** ±0.06
Asia	None	0.85±0.18*	0.73±0.22*	**0.31** ±0.09
	1	0.93±0.18*	0.87±0.27*	**0.42** ±0.15
	2	1.17±0.17*	0.93±0.27	**0.63** ±0.26
Insurance	None	1.82±0.16*	1.51±0.13*	**0.76** ±0.06
	3	1.96±0.15*	1.56±0.11*	**0.87** ±0.05
	5	2.08±0.13*	1.66±0.11*	**1.01** ±0.05
Alarm	None	2.43±0.15*	2.13±0.12*	**0.66** ±0.06
	3	2.48±0.14*	2.20±0.14*	**0.64** ±0.01
	5	2.47±0.14*	2.20±0.09*	**0.79** ±0.06
Hailfinder	None	2.85±0.03*	2.47±0.02*	**1.03** ±0.07
	5	2.84±0.03*	2.47±0.02*	**1.00** ±0.05
	10	2.86±0.03*	2.49±0.03*	**1.06** ±0.04

In this section, we use the same sampled target and sources as in Table 5.1, but we introduce additional hidden variables in the target. We learn the target parameters by:

- conventional single task BN learning (STL),
- MatchCPT fitness with Aggregation fusion (CPTAgg) (Luis et al., 2010) (note that CPTAgg does not apply to latent variables, but we use their fitness and fusion functions in our framework), and
- our BNPTL.

Three conditions are considered:
(i) fully observed target data,
(ii) small number of hidden variables and
(iii) medium number of hidden variables.

In the hidden data conditions, the specified number of target network nodes are chosen uniformly at random on each trial, and considered to be unobserved, so the data for these nodes are not used.

Table 5.2 summarises the average K-L divergence per parameter. In summary, the transfer methods outperform conventional EM with MLE (STL) in all settings. Compared with the state-of-the-art CPTAgg, BNPTL also improves performance on 15 out of 18 experiments (Table 5.2), with an average margin of 53.6% (the average reduction of K-L divergence). That means, of the total set of individual target NPTs, 84.3% showed improvement in BNPTL over CPTAgg.

5.4.6 Exploiting Piecewise Source Relatedness

Thus far, we simulated source relevance varying smoothly at the network level – all nodes within each source network were similarly relevant. So all fragments should typically be drawn from the source estimated to be most relevant. In contrast, for this experiment, we investigate the situation where relatedness varies in a *piecewise* fashion. In this case, to effectively learn a target network, different fragments should be drawn from different source networks. This is a setting where transfer in Bayesian networks is significantly different from transfer in conventional flat machine learning models (Pan and Yang (2010)).

To simulate this setting, we initialise a source network pool with three copies of the network, before introducing piecewise "hard noise", so that some compatible fragments are related and others are totally unrelated. Specifically, we choose a proportion (25% and 50%) of each source network's NPTs uniformly at random and randomise them to make them irrelevant (by drawing each entry uniformly from [0,1] and renormalizing). This creates a different subset of *compatible* but *(un)related* fragments in each network. Thus piecewise transfer - using different fragments from different sources is essential to achieve good performance.

We consider two evaluation metrics here: the accuracy of the fragment selection - whether each target fragment selects a (i) corresponding and (ii) non-corrupted fragment in the source, and accuracy of the learned NPTs in the target domain.

Table 5.3 presents the results, where our model consistently outperforms CPTAgg in Weather, Cancer and Asia networks. Although the fragment selection accuracy of

Table 5.3: The fragment selection performance of CPTAgg and BNPTL. The numbers 25% and 50% indicate different proportions of irrelevant fragments in the sources. Note that chance here is much lower than 75/50% due to unknown network correspondence.

Name	25% random NPTs				
	Fragment Accuracy		KLD		
	CPTAgg	BNPTL	STL	CPTAgg	BNPTL
Weather	61.0%*	**90.0** %	0.03±0.02*	0.01±0.00	**0.01** ±0.00
Cancer	94.8%	**96.0** %	0.33±0.31	0.14±0.09	**0.07** ±0.05
Asia	78.0%*	**97.5** %	0.85±0.18*	0.67±0.14*	**0.18** ±0.00
Insurance	**82.4** %	70.7%*	1.82±0.16*	1.01±0.04*	**0.74** ±0.02
Alarm	**61.7** %	58.8%*	2.43±0.15*	1.60±0.27*	**0.57** ±0.02
Hailfinder	**75.5** %	62.4%*	2.85±0.03*	2.04±0.03*	**0.79** ±0.02
Average	75.6%	**79.2** %	1.38±0.14	0.91±0.10	**0.39** ±0.02
	50% random NPTs				
Weather	57.0%*	**74.5** %	0.03±0.02*	0.01±0.00	**0.01** ±0.00
Cancer	79.2%	**82.4** %	0.33±0.31	0.13±0.07	**0.08** ±0.04
Asia	61.5%*	**80.8** %	0.85±0.18*	0.42±0.19	**0.20** ±0.01
Insurance	**65.9** %	51.9%*	1.82±0.16*	0.97±0.05	**0.90** ±0.04
Alarm	**51.0** %	46.4%*	2.43±0.15*	1.38±0.17*	**0.63** ±0.04
Hailfinder	**65.7** %	49.9%*	2.85±0.03*	2.07±0.03*	**0.43** ±0.02
Average	63.4%	**64.3** %	1.38±0.14	0.83±0.09	**0.38** ±0.03

BNPTL failed to outperform the CPTAgg in Insurance, Alarm and Hailfinder networks due to the greater data scarceness in their target networks, the general good performance (K-L divergence) of BNPTL verifies that the framework still can exploit source domains with piecewise relevance. Meanwhile the fragment selection accuracy of BNPTL explains how this robustness is obtained (irrelevant fragments (equation (5.3)) are not transferred (equation (5.7))). In addition to verifying that our transfer framework can exploit different parts of different sources, this experiment demonstrates that it can further be used for diagnosing which fragments correspond or not (equation (5.3)) across a target and a source – which is itself of interest in many applications.

5.4.7 Robustness to Irrelevant Sources

The above experiments verify the effectiveness of our framework under conditions of varying source relatedness, but with homogeneous networks $V_t = V_s$. In this section we verify robustness to two extreme cases of partially and fully irrelevant heterogeneous sources.

Partially irrelevant In this setting, we use the same six networks from the BN reposi-

Table 5.4: Performance (domain-partially-irrelevant) of STL and transfer learning methods: CPTAgg and BNPTL.

Name	Domain Accuracy		KLD		
	CPTAgg	BNPTL	STL	CPTAgg	BNPTL
Weather	80.0%*	**100.0** %	0.03±0.02*	**0.01** ±0.00	**0.01** ±0.00
Cancer	80.0%*	**92.0** %	0.33±0.31	0.11±0.07	**0.07** ±0.04
Asia	77.5%*	**85.0** %	0.85±0.18*	0.49±0.15*	**0.18** ±0.01
Insurance	**97.8** %	**97.8** %	1.82±0.16*	0.82±0.03*	**0.51** ±0.02
Alarm	**94.1** %	82.7%*	2.43±0.15*	1.64±0.06*	**0.70** ±0.03
Hailfinder	99.3%*	**100.0** %	2.85±0.03*	1.74±0.01*	**0.84** ±0.02
Average	88.1%	**92.9** %	1.38±0.14	0.80±0.05	**0.38** ±0.02

tory, and consider each in turn as the target, and copies of all six networks as the source (thus five are irrelevant and one is relevant). Therefore the majority of the potential source fragments come from 5 irrelevant domains. Table 5.4 presents the results of transfer learning in these conditions. We evaluate performance with two metrics: (i) percentage of fragments chosen from the correct source domain, and (ii) the usual K-L divergence between the estimated and ground truth parameters in the target domain.

As shown in Table 5.4, our BNPTL clearly outperforms the previous state-of-the-art CPTAgg in each case. This experiment verifies that our framework is robust even to a majority of totally irrelevant source domains, and is achieved via explicit relatedness estimation ($p(H_1^s)$ in Algorithm 5.1 and equation (5.3)).

Fully irrelevant In this setting, we consider the extreme case where the source and target networks are totally different $G_t \neq G_s, V_t \cap V_s = \emptyset$. Note that since the source and target are apparently unrelated, it is not expected that positive transfer should typically be possible. The test is therefore primarily whether negative transfer (Pan and Yang, 2010) is successfully avoided in this situation where all source fragments may be irrelevant. Note that since the sources are totally heterogeneous, prior work CPTAgg (Luis et al., 2010) does not support this experiment. We therefore compare our algorithm to a variant using BMC fitness and Basic fusion function (denoted BMCBasic) and target network only STL.

The results are shown in Table 5.5, from which we make the following observations. (i) BNPTL is never noticeably worse than STL. This verifies that our framework is indeed robust to the extreme case of no relevant sources: $p(H_0^s|D^t, D^s)$ is correctly inferred in equation (5.3), thus preventing negative transfer from taking place (equation

Table 5.5: Performance (domain-fully-irrelevant) of STL and transfer learning methods: BMCBasic and BNPTL. The symbol $\leftarrow\!\!\prec$ represents the transfer relationship: target $\leftarrow\!\!\prec$ source. Here 'Other' represents the six BN repository networks with the target removed.

Transfer Setting	STL	BMCBasic	BNPTL
Asia $\leftarrow\!\!\prec$ Other	0.85±0.18*	0.34±0.02*	**0.19** ±0.03
Weather $\leftarrow\!\!\prec$ Other	**0.03** ±0.02	0.21±0.01*	0.04±0.01
Cancer $\leftarrow\!\!\prec$ Other	0.33±0.31	0.23±0.01*	**0.08** ±0.02
Alarm $\leftarrow\!\!\prec$ Other	2.43±0.15	2.59±0.11*	**2.27** ±0.14
Insurance $\leftarrow\!\!\prec$ Other	**1.82** ±0.16	2.28±0.13*	**1.82** ±0.15
Hail.$\leftarrow\!\!\prec$ Other	**2.85** ±0.03	3.12±0.03*	2.86±0.03
Average	1.38±0.14	1.46±0.05	**1.21** ±0.06

(5.7)). (ii) In some cases, BNPTL noticeably outperforms STL, demonstrating that our model is flexible enough to achieve positive transfer when there exists some randomly matched distributions in the fully heterogeneous sources. (iii) In contrast, BMCBasic is worse than STL overall demonstrating that these properties are unique to our approach.

5.5 Summary

In this chapter, we proposed a two-step general-purpose BN parameter learning transfer framework, which tackles the scarce data problem by leveraging a set of source BNs. By making an explicit inference about relatedness per domain and per fragment, we are able to perform robust and effective transfer even with heterogeneous state spaces and piecewise source relevance.

Our approach applies when there are latent variables, and is robust to widely range degrees of source network relevance, automatically adjusting the strength of fusion to take this into account. Moreover, it is able to provide estimated domain and fragment-level relatedness as an output, which is of interest in many applications (e.g., in the medical domain to diagnose differences in procedures between hospitals). A variety of experiments (considering different data scarceness and piecewise-relatedness) show that BNPTL consistently outperforms single task STL and previous transfer learning algorithms.

In transfer with totally irrelevant sources, the results of our approach are no worse than the competitors, because the developed fitness and fusion function "fix" the selected irrelevant sources. The transfer without relatedness information might work if we can properly identify the related variables with the use of information coming not

only from the target samples of a given variable or a parent configuration alone. For example, because of the multiple distributions of a NPT, we can use the appearance together of apparently related distributions to identify relatedness. However, root nodes would have only one distribution, and using its associated target data and learned distributions to decide relatedness is statistically impossible in general.

Chapter 6

A Generic Framework for Parameter Learning with All Information

To address the challenge of learning accurate BNs when training data is scarce, we have so far introduced two approaches that have been proven useful:

(i) introducing expert judgments (Chapter 3 and 4) and

(ii) transferring knowledge from related domains (Chapter 5).

In this chapter, we present the first generic framework that combines both approaches to improve BN parameter learning. It serves to integrate both knowledge transfer and expert constraints. Experimental results demonstrate improved accuracy of the new method on a variety of benchmark BNs, showing its potential to benefit many real-world problems.

6.1 MPL-TC Model

Chapters 3–5 discussed two methods (constrained parameter learning and parameter transfer learning) to address the challenge of accurately learning BN parameters with limited data or no relevant training data. While incorporating either parameter constraints or transfer learning from related data in source domains can improve parameter estimation accuracy, there exists no generic learning framework to synergistically exploit the benefits of both approaches. Achieving this is non-trivial because typical approaches to transfer (Luis et al., 2010) and to constrained learning (Zhou et al., 2014a)

use very different formalisations. Here we generalise the MPL-C model[1] for learning with expert constraints to also exploit knowledge from related source domains via a bootstrap approach. The new model called MPL-TC (Multinomial Parameter Learning model with Transferred prior and Constraints) synergistically exploits both forms of external knowledge to improve learning performance in the target BN.

Assuming for now no latent variables in the target domain, then each target fragment i can be learned independently:

$$\hat{\theta}_i^t = \arg\max_{\theta_i^t} p(\theta_i^t | C_i^t, \mathscr{D}_i^t, \{\{\mathscr{D}_{i'}^s\}\}) \tag{6.1}$$

where C_i^t denotes target constraints.

In Chapter 5, we proposed a two-step BN parameter transfer learning framework. Based on that, we treat the selected source as the target parameter priors and then introduce them into the MPL-C model. Thus, the fusion step is implemented in the Bayesian way. The MPL-TC model contains three main steps:

(i) Compatibility and Permutation Mapping: for each BN fragment in the target domain, find its closest source fragment and permutation in the source domain;

(ii) Bayesian Fusion Step: transfer the selected source fragment by converting the source data statistics into prior distributions of parameters in the target MPL-C model (see Chapter 3);

(iii) MPL-TC Inference: perform the inference in this auxiliary model (now called as MPL-TC) to learn the target parameters.

The details can be found in Algorithm 6.1. Next, we will discuss these steps in turn.

Compatibility and Permutation Mapping As same as the transfer setting of BNPTL, MPL-TC also transfers at fragment-level. Therefore, in MPL-TC, each source fragment is firstly checked for compatibility with the target. Then the compatible fragments are permuted to generate all the possibles mappings between the target and source fragments. Finally, the relevance is measured for each valid mapping. However, because now we are provided with some target parameter constraints C_i^t, there are

[1]The MPL-EC model is simple extension of MPL-C which only address exterior constraints, as shown in Chapter 4, introducing such constraints generated from monotonic causalities will introduce heavy computation cost. Thus we mainly consider elicited interior constraints in this chapter.

INPUT : Target domain $\mathscr{D}^t$, source domains $\{\mathscr{D}^s\}$ and target constraints C^t.
OUTPUT: The target parameters $\hat{\theta}^t$.

1 **for** *each target fragment i* **do**
2 **for** *each source network s and fragment i'* **do**
3 **if** $compatible(C_i^t, \mathscr{D}_i^t, \mathscr{D}_{i'}^s)$ **then**
4 $P = permutations(G_i^t, G_{i'}^s)$;
5 **for** *permutation* $m = 1$ **to** M **do**
6 measure relatedness:
 $fitness(\mathscr{D}_i^t, P_m^{si'}(\mathscr{D}_{i'}^s)) = p(H_{ii'1}^s | D_i^t, P_m^{si'}(D_{i'}^s))$;
7 **end**
8 **end**
9 **end**
10 **end**
11 **for** *each target fragment i* **do**
12 Find the best source fragment and permutation:
 $\arg\max_{i',s,m} p(H_{ii'1}^s | D_i^t, P_m^{si'}(D_{i'}^s))$;
13 **for** *each parent state configuration j* **do**
14 **for** *each state value k* **do**
15 $\left\{\theta_{i'jk}^s\right\} = bootstrap(100, @MLE, P_m(D_{i'j}^s))$;
16 Fit the $\left\{\theta_{i'jk}^s\right\}$ with $\zeta_{i'jk}^s = TNormal(\mu_{ij'k}, \sigma_{ij'k}, 0, 1)$;
17 **end**
18 Generate the auxiliary model in the target:
 $\Psi_{ij}^t = mplc(D_{ij}^t, C_{ij}^t, \zeta_{i'jk}^s)$;
19 Inference to get the parameters estimation: $\hat{\theta}_{ij}^t = inference(\Psi_{ij}^t)$;
20 **end**
21 **end**
22 **return** $\hat{\theta}^t = \left\{\hat{\theta}_{ij}^t\right\}$

Algorithm 6.1: Multinomial parameter learning with transferred prior and constraints (MPL-TC)

some differences in the fragment compatibility check. Specifically, the source parameter $\theta_{i'}^s$ associated in compatible source fragment should fall in the constrained value ranges $\Omega_{C_i^t}$.

Based on previous equation (5.2), we have the new *compatible* function (Line 3, Algorithm 6.1) in MPL-TC:

$$compatible(C_i^t, \mathscr{D}_i^t, \mathscr{D}_{i'}^s) = \begin{cases} 1 & if\ G_i^t = G_{i'}^s\ \&\&\ \theta_{i'}^s \in \Omega_{C_i^t} \\ & \quad \&\&\ \dim(\theta_i^t) = \dim(\theta_{i'}^s) \\ 0 & \text{otherwise} \end{cases} \tag{6.2}$$

where i and i' index the fragments in target and source domain respectively.

Bayesian Fusion Step To fuse the selected source fragment with the target fragment in the second step of our algorithm, we perform the bootstrap approach in the source to generate the priors of target parameters. Bootstrap is a re-sampling method to measure the quality of true samples (Duval, 1993). In this chapter, we are interested in the quality of selected source parameters. We cannot access infinite training samples of selected source; instead we only have a sample of it (the selected best mapping source sample $D_{i'j}^s$), which means the MLE of $\theta_{i'jk}^s$ is not accurate.

From a specific subset of source samples, i.e., $D_{i'j}^s$, only one estimate of the MLE for a parameter of interest $\theta_{i'jk}^s$ can be obtained. In order to reason about the population, we need some sense of the variability of the estimated MLE. Thus, we apply the simplest bootstrap method – sampling from the $D_{i'j}^s$ to form a new sample (called a "re-sample" or bootstrap sample) that is also of size $|D_{i'j}^s|$. The bootstrap sample is taken from the original using sampling with replacement. This process is repeated multiple times (100 or 1000), and for each of these bootstrap samples we compute the MLE of $\theta_{i'jk}^s$ (each of these are called bootstrap estimates). We now have a set of bootstrap estimates, which are used to fit a *TNormal* distribution[2] to encode how much the source MLE varies.

MPL-TC Inference In our MPL-TC approach, these *TNormal* distributions ($\{\zeta_{i'jk}^s\}$) are used to replace the *uniform* parameter priors on θ_{ijk}^t (Figure 3.2) of MPL-C models in the target domain. Thus the transferred prior, target training samples and constraints are now all encoded in the target MPL-TC models (Line 18, Algorithm 6.1). After observing the sources, the target data statistics $(N_{ij}^t, N_{ij1}^t, ..., N_{ijr_i}^t)$ and available constraints (the constraint nodes are all observed with '*true*' values), we can update (by $inference(\cdot)$ function in Algorithm 6.1) these auxiliary models to get the target parameter posteriors:

[2]The abbreviation of *Truncated Normal* distribution.

$$p(\hat{\theta}^t_{ij1},...,\hat{\theta}^t_{ijr_t}|N^t_{ij},N^t_{ij1},...,N^t_{ijr_i},C^t_1,...,C^t_M,\zeta^s_{i'jk},...,\zeta^s_{i'jr_{i'}},sum)$$

Because the auxiliary BNs are hybrid models, the update/inference (Line 19, Algorithm 6.1) uses the DDJT algorithm as discussed in Chapter 3. The time complexity of the inference is exponential in model treewidth, which restricts the applicability at some point. However, approximate inference could be used with dynamic discretization to improve the time efficiency.

6.2 Illustrative Examples

Bootstrap Fitting of TNormal Priors Figure 6.1 demonstrates an example of fitted *TNormal* distributions for MLE estimation $\theta^s_{i'jk} = 0.2$ learned from sample sizes 10 and 100. As we can see, although the parameter estimations of two source samples are the same, the estimation from the large sample are clearly more reliable than the estimation from the small sample, where the fitted *TNormal* distribution in $|D^s_{i'j}| = 100$ is much sharper than the distribution in $|D^s_{i'j}| = 10$. Moreover, the number of bootstrap replicates does not change the results much. Here we use 1000 replicates in all subsequent experiments.

Fragment Transfer Here we provide an illustrative example of our framework for fragment-based parameter transfer and the target parameter estimation: the target is a three node BN shown in the left part of Figure 6.2 (a), and the source is an eight node BN shown in the right part of Figure 6.2 (a). We aim to estimate the NPT of S^t, which has four parent state configurations – π^t_1, π^t_2, π^t_3 and π^t_4. As we can see, there are two source fragments ($\{T^s,L^s,E^s\}$ and $\{E^s,B^s,S^s\}$) which are *compatible* with the target fragment (shown with dashed triangle in Figure 6.2 (a)). Thus, there are four *permutations* of compatible source fragments (assuming binary parent nodes). All four of these options are then evaluated for *fitness*, and the best fragment and permutation is picked ($\{B^s,E^s,S^s\}$). In Figure 6.2 (b), we generate four auxiliary BNs (MPL-TC model) for each target parameter column, the right part of Figure 6.2 (b) shows the MPL-TC model of the first parameter column, which is used to estimate θ^t_{11} and θ^t_{12}. The constraints and data statistics in the target domain are modelled by grey nodes. The parameter priors (white nodes) are *TNormal* distributions fitted from source bootstrap samples. Finally, these priors are updated by observing the target data statistics and constraints to get the posterior parameter estimates.

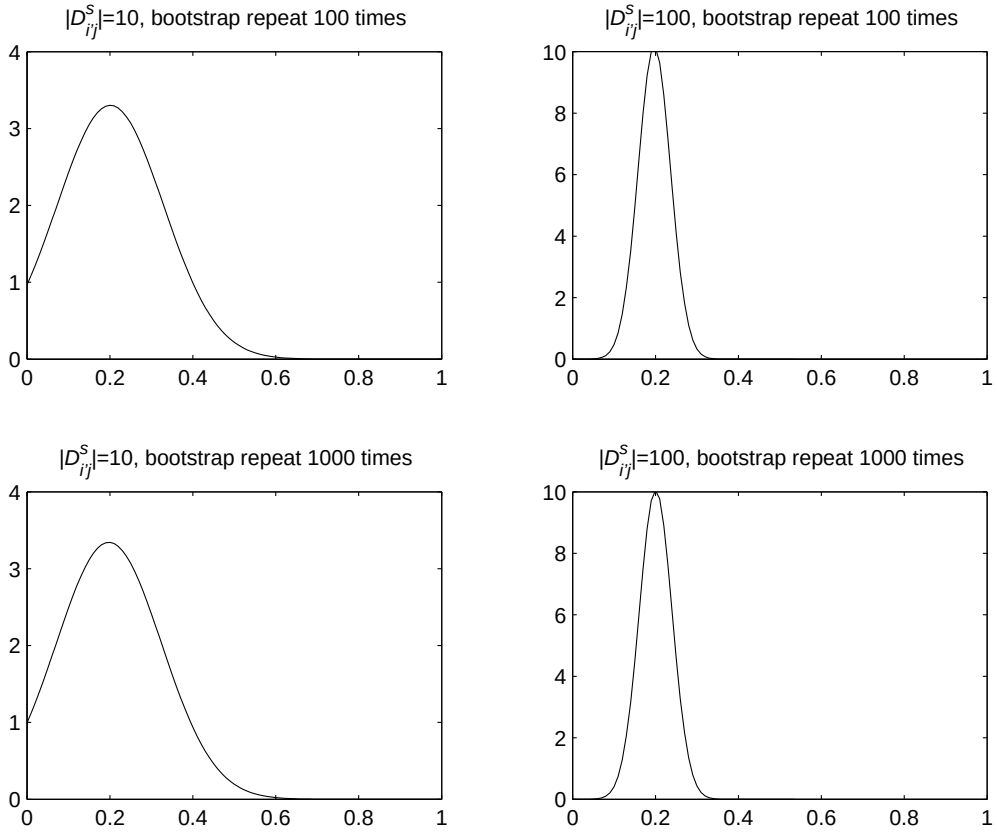


Figure 6.1: The fitted *TNormal* distributions for a parameter of interest with different source sample sizes 10 and 100. The original source MLE estimation of this parameter is 0.2, which means there are 2 and 20 appearances of this parameter in sample sizes 10 and 100 respectively. The bootstrap process in each case is repeated 100 and 1000 times.

6.3 Experiments

As same as previous experiment setting, in all cases, we assume that the structure of the model is known and that the 'true' NPTs that we are trying to learn are those that are provided as standard with the benchmark BN models. For the purpose of the experiment we are not given these true NPTs but instead are given a limited number of sample observations which are randomly generated based on the true target NPTs. To introduce noise between the target and source for simulating varying relatedness, the source datasets are also sampled from the true NPTs but with 'soft' and 'hard' noise conditions:

(i) soft: for each BN, three sources are learnt from 200, 300 and 400 samples respectively to simulate continuously varying relatedness among the sources;

(ii) hard: choose a proportion (20%) of each source's fragments uniformly at random

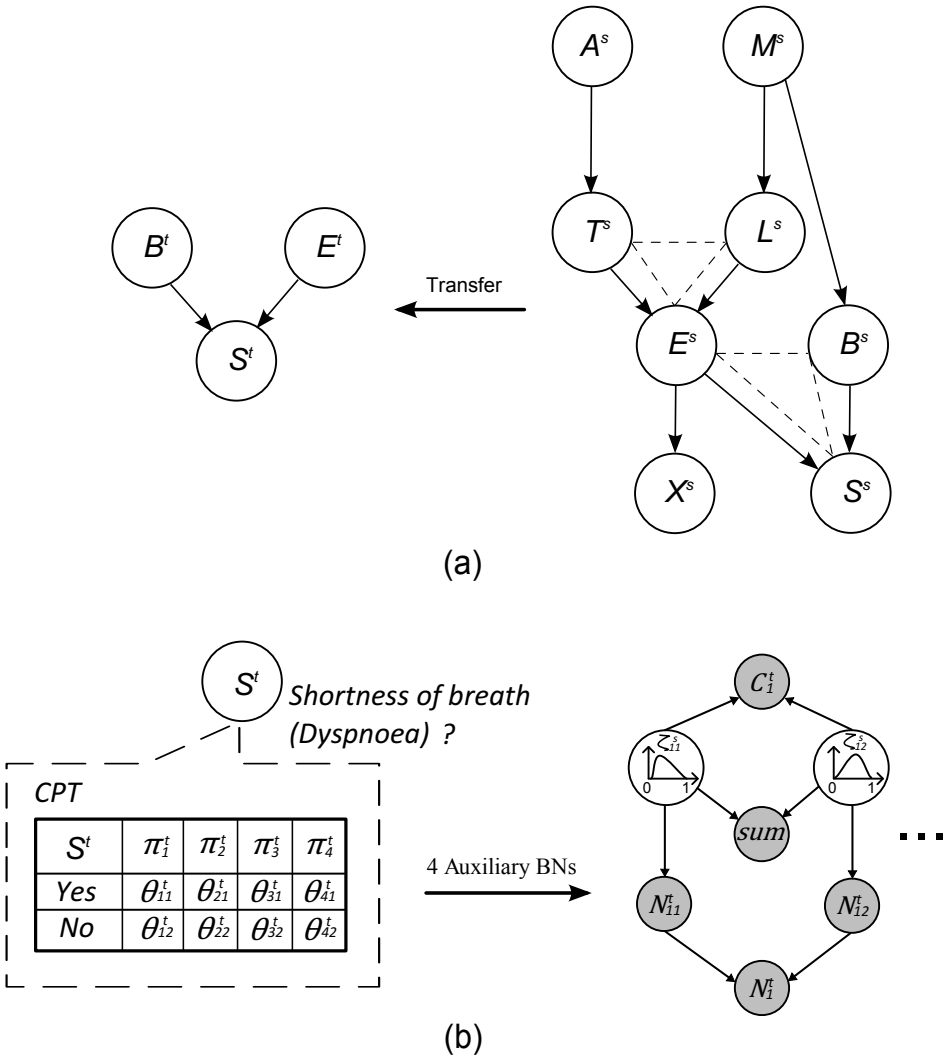


Figure 6.2: A simple example to show the framework of multinomial parameter learning with transferred prior and constraints. (a) The dashed triangle represents source fragments $\{T^s, L^s, E^s\}$ and $\{E^s, B^s, S^s\}$, which are *compatible* to the target fragment. (b) The structure representation of the MPL-TC model for estimating θ^t_{11} and θ^t_{12} is in the first target parameter column, whose parameter priors (θ^s_{11} and θ^s_{12}) are converted form the most fit source fragment $\{B^s, E^s, S^s\}$ via bootstrap.

and randomise their data/NPTs to make them irrelevant.

This results in a different subset of compatible but (un)related fragments in each source. Introducing these two types of sampling noise makes the sources similar but different to the target, and hence simulates the kind of source-target relations that may exist in practice.

The constraints are elicited from the true NPTs (so they are certainly correct) and randomly assigned to parameters in the network. Following the method of constraints generation in (Liao and Ji, 2009), for each true parameter θ_{ijk}, we create a constraint:

$$\min((1+\varepsilon)\theta_{ijk},1) > \theta^{t}_{ijk} > \max((1-\varepsilon)\theta_{ijk},0)$$

where the $\varepsilon = 0.05$ in the experiments. These elicited constraints are encoded in the MPL-TC auxiliary models, which are built with BN software AgenaRisk.

We compare our MPL-TC^{+5} against the following algorithms and settings (the upper right superscript value associated with learning algorithms represents the number of constraints used in these algorithms):

- MLE and MAP, conventional BN parameter learning algorithms.
- MPL-C^{+5}, state-of-the-art parameter learning algorithm with five constraints (Zhou et al., 2014a).
- CPTAgg, state-of-the-art parameter transfer learning algorithm (Luis et al., 2010).
- MPL-TC^{+0}, our MPL-TC algorithm with zero constraints.

The resulting learned NPTs are evaluated against the true NPTs by the average K-L divergence measure (equation (2.21)). The smaller the K-L divergence is, the closer the estimated NPT is to the true NPT. Each experiment is repeated 10 times, and the results are reported with the mean and standard deviation of the K-L divergences between estimated and true NPTs.

6.3.1 Experiments on the Cancer BN

As explained in Table 3.4, there are 10 independent parameters to learn in the Cancer model. In the target domain, training samples under different sparsity levels (10 to 100 samples) are drawn from the true Cancer BN.

Overall Results Figure 6.3 presents the results of all learning algorithms under varying data volumes in the target Cancer BN. It is clear that the average K-L divergence of all learning algorithms decreases with increasing target sample size. With increasing sample sizes, the performance gap between the algorithms decreases[3]. Moreover, our MPL-TC^{+5} always outperforms all other the competitors, which demonstrates the effectiveness of our framework.

[3]Given enough target training samples, the learning performance of all algorithms converge.

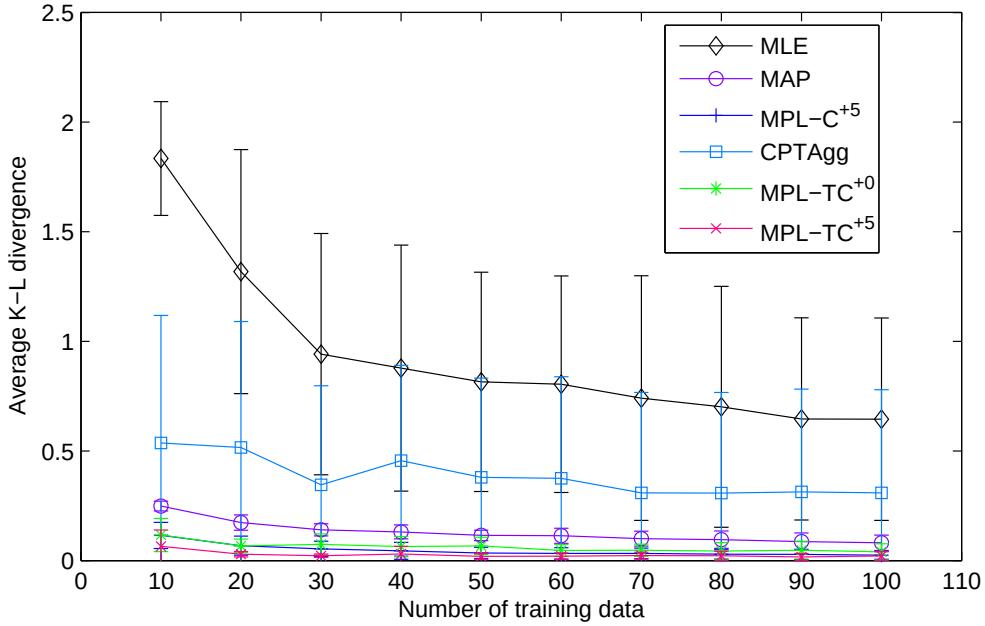


Figure 6.3: Parameter learning performance in the Cancer BN under different levels of data sparsity. Lower is better.

Considering models without parameter constraints (MLE, CPTAgg, MAP and MPL-TC^{+0}): MPL-TC^{+0} provides overall K-L divergence reductions (performance improvements) of 93.4%, 84.1% and 52.3% compared with MLE, CPTAgg and MAP respectively, thus demonstrating the efficacy of knowledge transfer.

After introducing 5 sampled constraints, MPL-TC^{+5} achieves even greater reductions in comparison with MLE, CPTAgg and MAP, which are 97.1%, 93.0% and 79.1% respectively. Due to the benefit of introducing parameter constraints, MPL-C^{+5} algorithm also outperforms MLE, CPTAgg and MAP. However, MPL-TC^{+5} still outperforms MPL-C^{+5} with 42.0% average K-L divergence reduction.

According to the results, the MPL-TC^{+5} greatly outperforms the conventional MLE and MAP algorithms, and the CPTAgg and MPL-C^{+5} that only use transfer or constraints alone. This demonstrates the complementarity of both constraints and sources of external knowledge when learning with scarce target data.

Varying Number of Constraints To investigate how the number of introduced constraints affect the learning performance of MPL-C and MPL-TC, we vary the number of sampled constraints in parameter learning (shown in Figure 6.4). As we can see, K-L divergence decreases with more constraints for both MPL-C and MPL-TC in both data sparsity settings. However, when the number of constraints is small, our MPL-TC

greatly outperforms MPL-C due to the benefit of transferred parameter priors. When the number of constraints increases to 10 (every parameter is constrained), the learning results of MPL-C and MPL-TC both converge to zero in both settings.

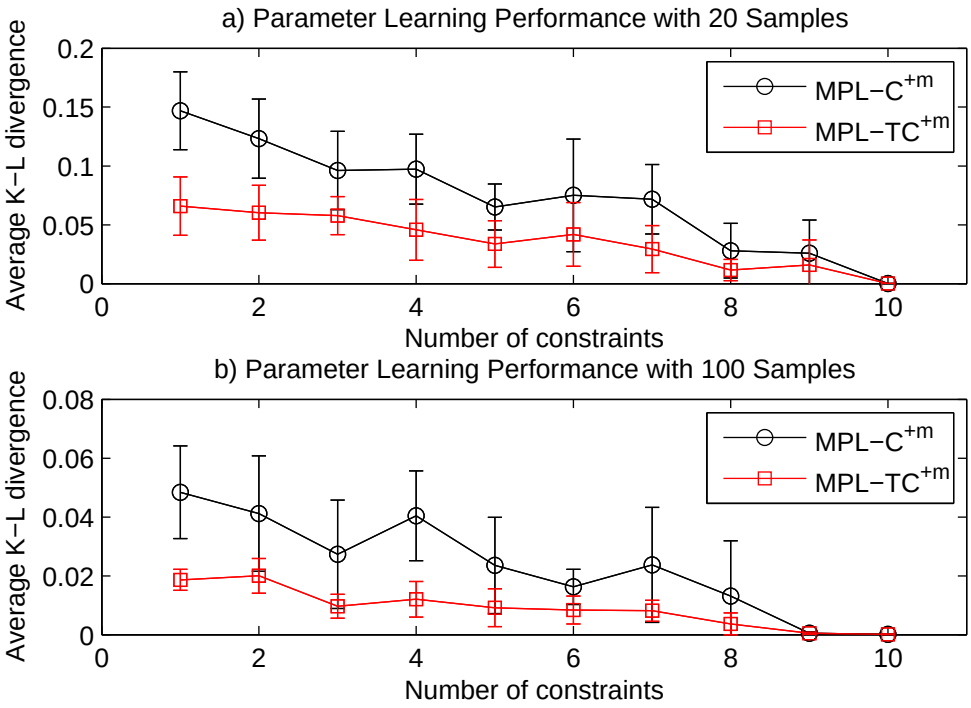


Figure 6.4: Performance of MPL-C and MPL-TC when varying the number of constraints ($m = 1, ..., 10$).

Priors vs. Posteriors To provide insight into the mechanisms of our framework, we investigate the differences between MPL-TC (Priors) (transferred *TNormal* mean values) and MPL-TC^{+5} (Posteriors) (the updated parameter posteriors after inference given the target data and parameter constraints) for each parameter in the Cancer BN. The results are presented in Figure 6.5, where the heights of the bars represent the absolute differences between estimated values and true NPT values.

As we can see, the MPL-TC (Priors) shows inaccurate transfer in both of two settings: parameters 7–9 in Figure 6.5 (a) and parameters 3–7 and 9 in Figure 6.5 (b). This is caused by the bias in target samples and noise in source domains. Therefore, the estimates of MPL-TC (Priors) are far from the true values, (average K-L divergence of 0.65 and 0.40 for sample sizes 20 and 100 respectively). However, after performing MAP learning in the MPL-TC^{+5} model, the MPL-TC^{+5} (Posteriors) reduces the average K-L divergence between the estimated values and true values to 0.09 and 0.03 respectively. These results demonstrate the robustness of the Bayesian learning in MPL-TC^{+5}, and

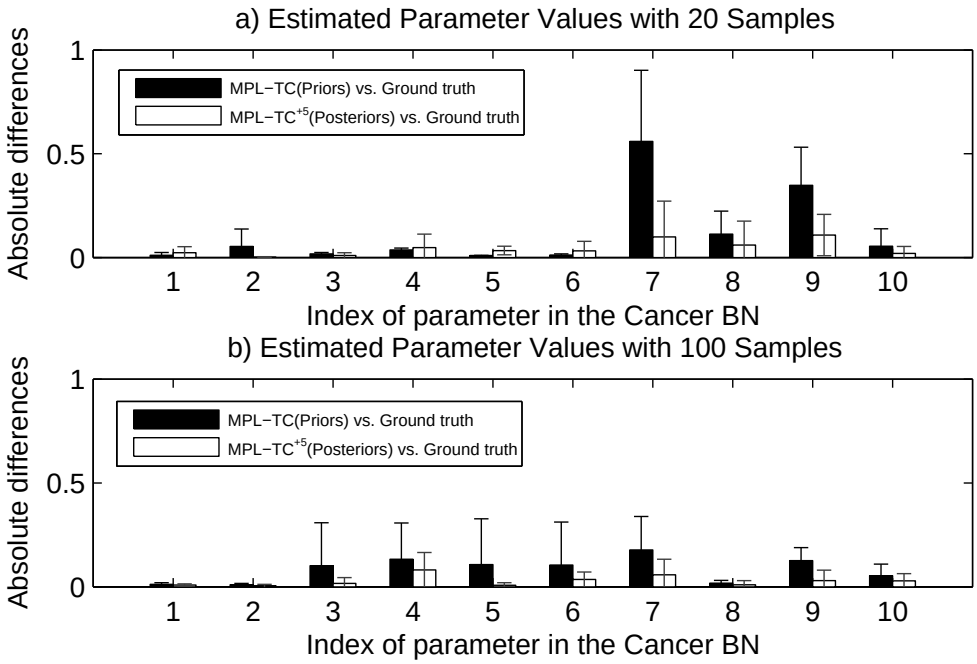


Figure 6.5: The differences between estimated probability values (MPL-TC(Priors) and MPL-TC^{+5}(Posteriors)) and ground truth for all parameters in the Cancer BN.

the importance of systematically inferring the new parameters given available data and constraints. Next, we will compare the performance of all these algorithms in different BN parameter learning problems.

6.3.2 Experiments on Standard BNs

To enrich the experiments, we evaluate the algorithms on 12 standard BNs[4] (details in Table 6.1). For each BN, 100 training samples and 5 constraints are drawn from the true NPTs in the target domain.

Overall Table 5.2 summarises the average K-L divergence per parameter. The best results are presented in bold, and the comparison is statistically significant at the 5% significance level. In summary, the MPL-C^{+5} and MPL-TC^{+5} methods outperform conventional MLE and MAP in 11 out of 12 settings, the only exception is the learning performance in Weather BN, where the learning results of these methods converge with enough training samples[5]. These results demonstrate the benefit of learning with both sources of external knowledge. Compared with the state-of-the-art MPL-C^{+5}, MPL-TC^{+5} wins in every setting (including the Weather BN, where MPL-TC^{+5} achieves

[4] `http://www.bnlearn.com/bnrepository/`

[5] As shown in Table 5.2, the Weather BN only contains 9 parameters to learn, therefore 100 training samples are already enough to train a good model.

Table 6.1: Parameter learning performance (average K-L divergence) in 12 standard Bayesian networks.

Name	Nodes	Edges	Para	MLE	MAP	MPL-C^{+5}	CPTAgg	MPL-TC^{+0}	MPL-TC^{+5}
Alarm	37	46	509	2.36±0.10	0.66±0.01	0.61±0.02	1.61±0.08	**0.42**±0.02	**0.42**±0.01
Andes	223	338	1157	1.03±0.06	0.17±0.01	0.15±0.01	0.65±0.05	**0.08**±0.00	**0.08**±0.00
Asia	8	8	18	0.57±0.16	0.34±0.04	0.28±0.03	0.31±0.05	0.22±0.02	**0.18**±0.03
Cancer	5	5	10	0.86±0.35	0.09±0.04	0.07±0.05	0.54±0.11	0.05±0.01	**0.03**±0.01
Earthquake	5	4	10	1.50±0.82	0.15±0.04	0.13±0.03	0.35±0.22	0.11±0.01	**0.10**±0.01
Hailfinder	56	66	2656	2.85±0.01	0.46±0.00	0.41±0.00	1.98±0.01	**0.31**±0.01	**0.31**±0.01
Hepar2	70	123	1453	3.18±0.13	0.33±0.01	0.33±0.01	2.58±0.15	0.30±0.01	**0.29**±0.00
Insurance	27	52	984	1.95±0.18	1.17±0.03	1.07±0.03	0.93±0.06	**0.75**±0.03	**0.75**±0.02
Sachs	11	17	178	1.74±0.29	0.78±0.04	0.71±0.05	0.98±0.08	**0.50**±0.03	**0.50**±0.02
Survey	6	6	21	0.35±0.20	0.05±0.01	0.05±0.01	0.24±0.15	0.04±0.01	**0.03**±0.01
Weather	4	4	9	0.02±0.02	0.03±0.00	0.02±0.00	0.02±0.00	0.02±0.00	0.02±0.00
Win95pts	76	112	574	3.59±0.07	0.81±0.01	0.78±0.02	3.20±0.10	0.67±0.02	**0.64**±0.01

even smaller average K-L divergence – 0.018 of MPL-TC^{+5} vs. 0.020 of MPL-C^{+5}). Over all BNs, MPL-TC^{+5} gets 83.2%, 33.5% and 26.9% average reduction of K-L divergence compared with MLE, MAP and MPL-C^{+5} respectively.

Transfer vs. No Transfer Considering transfer learning only, both CPTAgg and MPL-TC^{+0} outperform conventional MLE, which demonstrates the benefit of introducing source domain knowledge. However, due to a simplistic relatedness model and NPT fusion heuristic, CPTAgg even fails to outperform MAP in some settings. In contrast, our MPL-TC^{+0} outperforms CPTAgg and MAP with 74.1% and 31.2% average reduction of K-L divergence over all the settings. In addition, after introducing both transferred parameter priors and target constraints, our MPL-TC^{+5} shows additional improved learning performance over CPTAgg and MPL-TC^{+0} (75.0% and 3.5% average reduction of K-L divergence).

Importance of Transfer vs. Constraints As we can see, our MPL-TC^{+0} outperforms MPL-C^{+5}, which indicates the transferred prior is more helpful than a moderate number (i.e., 5) of constraints in improving parameter learning performance in these experiments. Given the burden of constraint elicitation in the real world, we used a realistic limited number of constraints. Of course if sufficient constraints were available, MPL-C would perform better (cf Figure 6.4) and this result would be reversed. But in this case, the transferred prior makes a greater contribution in improving performance – despite the noise process between source and target domain, and the imperfect es-

timation of relevance. This is especially true in the larger BNs, where the constraints are scarcer relative to the number of parameters to learn. This also explains why MPL-TC^{+0} and MPL-TC^{+5} have similar results in the Alarm, Andes, Hailfinder, Insurance and Sachs BNs.

6.4 A Real Medical Case Study

The previous experiments demonstrated the effectiveness of our MPL-TC under controlled data, constraints and relatedness conditions. In this section we explore its application to learn BN parameters of a real medical network, where the "true" constraints and relatedness are unknown, and data volume and relatedness reflect the conditions of real-world medical tasks.

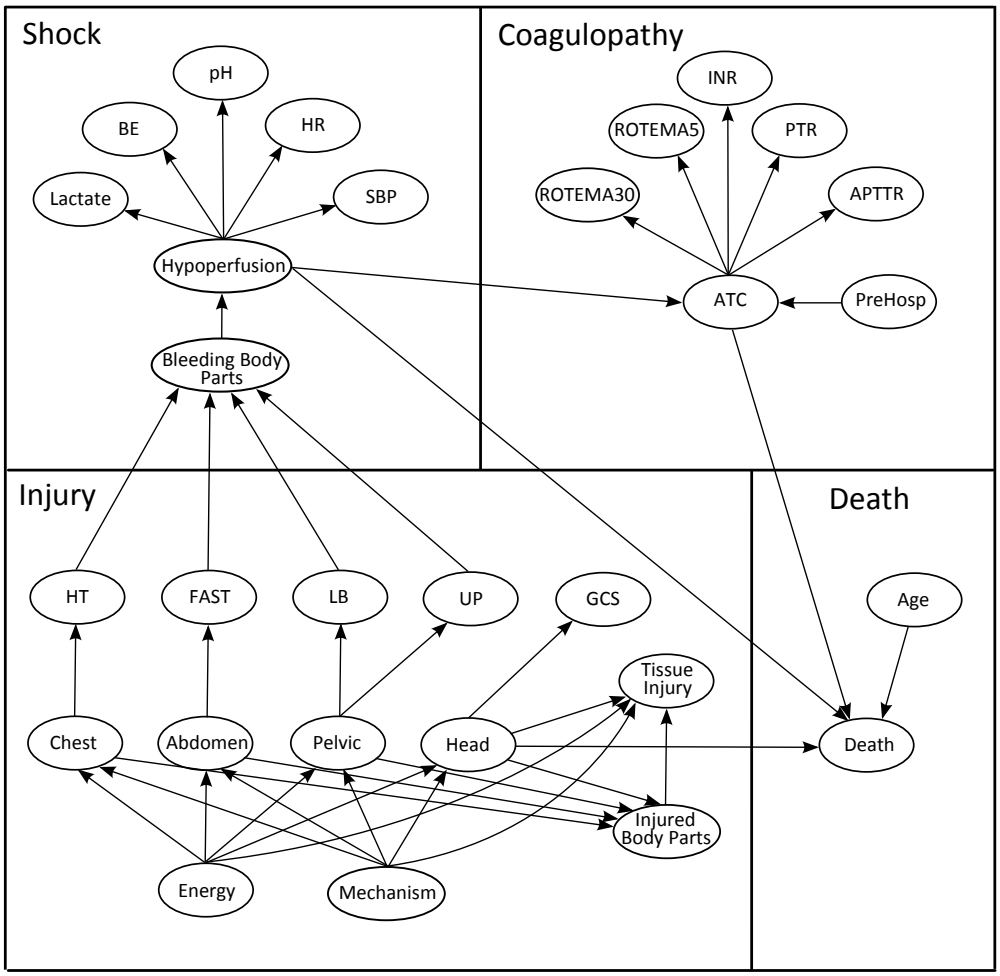


Figure 6.6: The Trauma Care Bayesian network, which contains four main parts: "Injury", "Shock", "Coagulopathy" and "Death".

The Trauma Care (TC) dataset (Yet et al., 2014) derives from a BN structure designed by trauma care specialists, and relates to procedures in hospital emergency

rooms. The full details of the BN (whose graph is shown in Figure 6.6) and datasets are proprietary to the hospitals involved. This BN contains 18 discrete variables (of which 3 are hidden) and 11 Gaussian variables[6] that are grouped into 4 parts:

(i) the degree of overall tissue injury,

(ii) the degree of hypoperfusion resulting from blood loss for the patient,

(iii) the risk of developing acute traumatic coagulopathy, and

(iv) the risk of death for the patient.

Here, a well learnt BN is important because rapid and accurate identification of hidden risk factors and conditions modelled by the network are important to support a doctors' decision making about treatments which reduce mortality rate (Karaolis et al., 2010).

In this experiment, we are provided with both expert judgments elicited from real experts in this field and two distinct datasets. Hence we are able, in a real-world setting, to evaluate the MPL-TC method.

The expert judgments are shown in Table 6.3, which constrain the variables: 'Death', 'ATC', 'Age', 'Hypoperfusion', and 'Head' (their details can be found in Table 6.2).

Table 6.2: Details of constrained variables.

Variable	Description	States
Death	The risk of patient's death in 48 hours.	No Yes
ATC	Acute traumatic coagulopathy.	No Yes
Age	Patient's age.	M: $45 < Age < 65$ O: $Age \geq 65$ Y: $Age \leq 45$
Hypoperfusion	The degree of decreased blood flow through an organ.	Compensated None Uncompensated
Head	Severe head injury of patient.	No Yes

One dataset is composed primarily of data from a large inner city hospital with extensive data (1022 instances) and the second dataset is composed of data from a smaller

[6]The details of these variables can be found in `http://www.traumamodels.com/`.

Table 6.3: The elicited expert judgments for the Trauma Care BN.

Judgment	Description	Constraints
1	ATC occurs will result in the death of patient with *very high* probability.	$p(Death = Yes \mid ATC = Yes) \approx 0.93$
2	Old patient has higher risk of death than young patient.	$p(Death = Yes \mid Head = O) \geq p(Death = Yes \mid Age = Y)$
3	Uncompensated hypoperfusion will *very likely* result in the death of patient.	$p(Death = Yes \mid Hypo. = U.) \approx 0.90$
4	Severe head injury will *likely* result in the death of patient.	$p(Death = Yes \mid Head = Yes) \approx 0.71$

Table 6.4: Prediction performance (AUC) for the Trauma Care BN. The query variable is "Death".

Algorithm	STL	ALL	CPTAgg	BNPTL	MPL-C	MPL-TC
AUC	0.77*	0.93	0.80*	0.94	0.91*	**0.97**

hospital and city in another country (30 instances). The smaller hospital would like an effective decision support model. However, using their own data to learn the model would be insufficient, and using the large dataset directly may be sub-optimal due to (i) differences in statistics of injury types in and out of major cities city, (ii) differences in procedural details across the hospitals and (iii) differences in demographic statistics across the cities/countries. Therefore, the TC BN from the inner city hospital can be used to generate the parameter priors of BN in the small hospital. Finally, elicited constraints are used with data instances in the small hospital to train the target TC BN model.

We perform cross-validation in the target domain of the small hospital, using half the instances (15) to train the model, and half to evaluate the model. To evaluate the model we instantiate the evidence variables in the target domain test set, select one of the variables of interest (*Death*), and query this variable. AUC values are calculated for the query variable. The results are presented in Table 6.4, with the best result in bold, and statistically significant improvements of the best result over competitors indicated with asterisks * ($p \leq 0.05$).

As shown in Table 6.4, all learning algorithms mentioned in this thesis have been compared. Here STL denotes single task learning from the small hospital data, ALL indicates the baseline of concatenating all the source and target data together before STL. CPTAgg and BNPTL are two parameter transfer learning algorithms. MPL-C

is the learning algorithm incorporating constraints for this TC BN. Because there is one exterior constraint (Judgment 2, Table 6.3), the MPL-EC model is also used here. However, its improvement on the classification is not significant, and its result is added into the result of MPL-C.

As we can see from the results, transferring data from the large hospital (CP-TAgg and BNPTL) or using domain expert judgments (MPL-C) both could improve the learning results. Moreover, ALL also achieves good performance based on the strong assumption of known node correspondence. Nevertheless, these learning algorithms are still outperformed by the MPL-TC (0.97), which uses all the available information.

6.5 Summary

The broad goal of this chapter was to introduce a new method (MPL-TC) that is the first attempt at BN parameter learning incorporating both transfer learning and qualitative constraints in a complementary way. Using the public BN repository, we showed that learning performance was greatly improved in MPL-TC across a range of networks. In particular, we demonstrated that the MPL-TC worked well in every data and constraint sparsity in the Cancer BN, and achieved the best performance in all BNs in the repository compared with other state-of-the-art algorithms. Most importantly, we showed that MPL-TC outperformed all other approaches in the real-world Trauma Care medical case study.

We discussed the limitations of all BN transfer learning approaches with respect to the fact that, in practice relatedness is hard to guarantee or estimate. Thus data-driven transfer (source selection) may be biased by inaccurate target data (resulting in bad choice of source and thus negative transfer) in extremely sparse settings. In the spirit of synergistically combining source data and constraints, available target constraints clearly provide an opportunity to guide and disambiguate transfer. In this chapter, we only used target parameter constraints to exclude individual incompatible source fragments. Richer models for guiding transfer with constraints including cross-node constraints, source-domain constraints, and cross-domain constraints should be investigated in future.

Chapter 7

Conclusions

In this final chapter, we discuss the main conclusions and contributions of the thesis (Section 7.1). This is followed by a discussion of future work (Section 7.2) and the final remarks (Section 7.3).

7.1 Contributions

The experimental results in this thesis has shown that, by incorporating small amounts of expert judgments and/or related data, the BNs parameter learning performance has been greatly improved over conventional and previous state-of-the-art parameter learning methods. In this section, we will discuss the overall conclusions and contributions in this thesis, which is organised according to the research hypothesis and objectives that were defined in Section 1.1. We first discuss how each of the research objectives I—V has been fulfilled.

Objective I: Review and analyse previously proposed algorithms for BN learning in general and parameter learning with both data and knowledge in particular.

In this thesis, we have undertaken a literature study in which previously proposed models and algorithms for BN learning in general (Section 2.2) and parameter learning in particular (Section 2.3) have been reviewed. A substantial portion of the review is based on recently published books (Koller and Friedman, 2009; Murphy, 2012; Fenton and Neil, 2012; Barber, 2012). In addition to constituting an updated overview of the research area, the literature study serves as a basis for the analysis of previous work, which includes the identification and discussion of key limitations of previous algorithms for parameter learning in general and parameter learning with scarce data and additional knowledge in particular (Section 2.5). To summarise, previous algorithms

typically suffer from one or more of the following limitations:

- The learning performance is more or less decided by setting the proper parameters, which makes implementing and tuning the algorithms more difficult.
- The current transfer learning algorithms are designed for network-level transfer, which may result in negative transfer in real-world applications.
- The fusion method in BN transfer learning is either a linear/logarithmic combination of the target and selected sources. However, the current method for getting the coefficients/weights is heuristic, which reduces the generalizability of the obtained results.

With the analysis of previous algorithms and their principal limitations, a number of key properties (Section 2.5) of new or updated algorithms for BN parameter learning with scarce data are identified. These properties constitute the research gap that is addressed in the thesis.

Objective II: Incorporate natural qualitative expert judgments or domain knowledge such that when combined with data, more accurate BN models can be built.

In this thesis, a novel auxiliary model for BN parameter learning with constraints was proposed and analysed. This model described in Chapter 3, MPL-C, fulfills most of the key properties identified in Section 3.3 for parameter learning with constraints (i.e., it supports multiple data scarceness and multiple types of constraints, and includes parameter priors). The data statistics and constraints converted from qualitative expert judgments are all encoded as nodes in the MPL-C model. Thus, the parameter learning process is achieved via the inference in the auxiliary model. In Section 3.2, commonly used types of qualitative expert judgments and their elicitation barriers are presented. The way to incorporate constraints converted from expert judgments are shown in Section 3.3. In Section 3.4, the inference method is discussed. The experiments on 6 standard BNs (Section 3.5) and a real software defects prediction BN (Section 3.6) demonstrate the benefits of using expert judgments in parameter learning.

Objective III: Investigate the way to reduce the burden of eliciting expert judgments, and propose new or updated models that are more appropriate for parameter learning with these judgments.

In Chapter 4, the burden of eliciting expert judgments was analysed and eliminated via introducing the monotonic causalities in BNs. A key property of a monotonic causality is that it enables the NPT values of the child node to monotonically increase or decrease as the parent state configuration varies. This introduces a set of exterior parameter constraints after knowing the labels of the monotonic causalities. We proposed a generative synergy model for monotonic causalities, which supports both homogeneous and heterogeneous synergies (Section 4.3). To learn with exterior constraint, the MPL-C is extended in this chapter. The new model is called as MPL-EC and presented in Section 4.4. The existence of monotonic causalities in real-world BNs is also investigated in Section 4.5.

Objective IV: Propose new algorithms to find the most relevant source Bayesian network or network fragments to transfer, and to fuse source and target knowledge in a robust way.

In Chapter 5, the parameter transfer learning algorithm is discussed which can be used to address the scarce data challenge by leveraging data from different but related problems. In Section 5.3, a two-step general-purpose BN parameter transfer learning framework called BNPTL is presented. This framework fulfills the key properties identified in Section 2.5 for parameter transfer learning. For example, it can robustly find the right source parameter and fuse it to the target. To verify this, we have also reproduced a previously published parameter transfer learning algorithm (CPTAgg). The experiments in Section 5.4 show that BNPTL consistently outperforms single task STL and former transfer learning algorithms.

Objective V: Propose extended algorithms for the generic parameter learning with both expert judgments and transferred knowledge, which leverages the benefits of both constraint-based and transfer-based parameter learning algorithms.

In Chapter 6, a generic framework has been proposed for parameter learning with both expert judgments and transferred knowledge. This framework is referred to as MPL-TC, which fulfills all of the key properties identified in Section 2.5. In Section 6.1, the details of using both expert judgments and transferred knowledge in the MPL-TC are discussed. Specifically, we create MPL-C models based on target data statistics and constraints, and transfer the source information as the target parameter priors. These priors are encoded in the target MPL-C models, and these models are updated

to produce the parameter posteriors. The experimental results on synthetic datasets demonstrate that the MPL-TC works well in every data and constraint scarceness.

7.2 Future Work

While the thesis has achieved its objectives of improved BN parameter learning with scarce data, there is scope for extensions and further improvements. In particular we identify the following:

- Current MPL-C, MPL-EC, and MPL-TC models actually learn the full posterior distribution for each parameter of interest. However, because the parameter corresponds to a probability value in a discrete NPT of the original BN, we currently simply take the mean of the posterior distribution as the NPT parameter value. Hence, we are 'throwing away' all the other information in the learnt distributions (such as the variance and percentiles). Future work is needed to avoid this loss of information.

- In the MPL-EC model, the generic synergy model support using the confidence weight of each introduced monotonic constraint. However, due to the limitations of current research resource, no real expert is involved to provide monotonic causality labels and associated confidence weights. To this end, an interactive tool should be developed to elicit such labels in real-world applications. And further investigation is needed to check whether the confidence of expert judgments has a significant impact on parameter learning performance.

- The current transfer fitness step only employs target constraints to exclude the incompatible source fragments. Ideally, these expert provided target constraints are correct and can be used to guide the transfer. Thus, we need to further investigate whether the target constraints can be used to help find more accurate sources.

- Our current work focuses on BN parameter learning where the BN structure is an input of our algorithms. However, in some empirical cases, the real structure of the network can not be accessed. Thus, a new method is needed to address both structure learning and parameter learning with constraints and transferred knowledge.

- Ideally, the proposed MPL-TC method in this thesis should be available as part of a standard BN tool. In other words, it should be possible within a single BN tool GUI to build a BN model, import both related datasets and a set of experts constraints relevant to the BN, and then have the NPTs in the BN automatically updated according to the MPL-TC method calculations. Additional work is needed to develop such tool.

7.3 Final Remarks

BNs are increasingly being used in critical risk assessment and decision support applications. There has been an over-reliance on the assumption that BNs can be learned from data alone. Hence, various parameter learning algorithms have previously been proposed to train the BNs. However, as discussed in the thesis, these algorithms typically suffer from scarce data limitations. In fact in most real-world applications the kind of data required for such algorithms to be accurate is simply not available. Hence we need to use domain knowledge like expert judgments and/or related datasets to boost the reliability of BN learning methods.

In this thesis, we have developed a set of novel BN parameter learning algorithms exploiting both data and domain knowledge. In particular, firstly, we have proposed two constrained parameter learning models: MPL-C and MPL-EC. These two models can address the parameter learning with interior and exterior parameter constraints respectively. These models are auxiliary BNs associated with each node whose NPT we wish to learn. Because the auxiliary BN is a hybrid model, we have employed a novel dynamic discretization junction tree algorithm for the inference. Secondly, we have proposed the parameter transfer learning algorithm (BNPTL) which leverages the source information at fragment-level. Finally, we have developed a first generic framework that combines both expert judgments and related knowledge to improve BN parameter learning. This framework is referred to as MPL-TC, since it based on both constraint-oriented (MPL-C and MPL-EC) and transfer-oriented (BNPTL) methods.

In our experiments on the publicly available BN repository and a real-world medical case study, MPL-TC achieves better performance in terms of average K-L divergence and AUC than conventional algorithms (MLE and MAP), state-of-the-art learning algorithms with constraints (MPL-C), and state-of-the-art parameter transfer learn-

ing algorithms (CPTAgg and BNPTL). We can summarize the comparison with other algorithms as follows: if the problem domain only contains expert judgments, MPL-C or MPL-EC is probably the best method to incorporate such judgments during the parameter learning; if the problem domain only contains source knowledge or datasets, the BNPTL algorithm will achieve good results even when the target data is limited; finally, if the problem domain contains both types of additional knowledge, the MPL-TC will use both of them in an applicable way, and improve the BN parameter learning results.

Appendix A

Java Code for Building MPL-C Model

```
package MPLC;
import java.util.*;
import uk.co.agena.minerva.model.*;
import uk.co.agena.minerva.model.extendedbn.*;
import uk.co.agena.minerva.model.scenario.Scenario;
import uk.co.agena.minerva.util.model.DataSet;
import uk.co.agena.minerva.util.nptgenerator.*;
/**
* @author Yun Zhou 20-10-2014
*/
public class MultinormialwithConstraints {

//This function can automatically create a MPL-C model
public ArrayList<Double> MultinormialwithConstraints(ArrayList<Integer> countList,
ArrayList<Constraint> constraintList, int numberiter) throws Exception {

ArrayList<Double> returnList = new ArrayList<Double>();
int               np         = countList.size();
double            trailsum   = 0.0;
for (int i = 0; i < np; i++) {
trailsum = trailsum + countList.get(i);
}
Model      model = Model.createEmptyModel();
ExtendedBN ebn   = model.getExtendedBNAtIndex(0);

//Step 1*****************************************************************
//Initial all the nodes
DataSet cidsp  = new DataSet();
cidsp.addIntervalDataPoint(0.0, 1.0);

int upinfinity = Integer.MAX_VALUE;
DataSet cidsn  = new DataSet();
cidsn.addIntervalDataPoint(0, upinfinity);

DataSet cidsc  = new DataSet();
cidsc.addLabelledDataPoint("True");
cidsc.addLabelledDataPoint("False");

ArrayList<ContinuousIntervalEN> pnodelist = new ArrayList<ContinuousIntervalEN>();
ArrayList<BooleanEN>            cnodelist = new ArrayList<BooleanEN>();
```

```
ArrayList<IntegerIntervalEN>    snodelist = new ArrayList<IntegerIntervalEN>();
ContinuousIntervalEN            psum      = new ContinuousIntervalEN();
ContinuousIntervalEN            nnode     = new ContinuousIntervalEN();

//Initialize all the nodes -- name and id
for (int i = 0; i < np; i++) {
String id   = "p"+i;
String name = "Probability"+i;
pnodelist.add(ebn.addContinuousIntervalNode(id, name));
pnodelist.get(i).createExtendedStates(cidsp);

String ids = "s"+i;
String names = "S"+i;
snodelist.add(ebn.addIntegerIntervalNode(ids, names));
snodelist.get(i).createExtendedStates(cidsn);
}

String id    = "psum";
String name  = "Sum";
psum = ebn.addContinuousIntervalNode(id, name);
psum.createExtendedStates(cidsp);

String idt   = "n";
String namet = "Number of Trails";
nnode = ebn.addContinuousIntervalNode(idt, namet);
nnode.createExtendedStates(cidsn);

for (int i = 0; i < constraintList.size(); i++) {
String idc   = "c"+i;
String namec = "Constraint"+i;
cnodelist.add(ebn.addBooleanNode(idc, namec));
cnodelist.get(i).createExtendedStates(cidsc);
}

//Step 2**************************************************************
//Define all the edges
for (int i = 0; i < np; i++) {
pnodelist.get(i).addChild(psum);
pnodelist.get(i).addChild(snodelist.get(i));
nnode.addChild(snodelist.get(i));
}

for (int i = 0; i < cnodelist.size(); i++) {
Constraint tmp = constraintList.get(i);
if (tmp.type) { //single contraint
pnodelist.get(tmp.nodeindex).addChild(cnodelist.get(i));
} else {
pnodelist.get(tmp.nodeindex).addChild(cnodelist.get(i));
int index2 = (int)tmp.value;
pnodelist.get(index2).addChild(cnodelist.get(i));
}
}

//Step 3**************************************************************
//Set simulation nodes
psum.setSimulationNode(true);
nnode.setSimulationNode(true);
```

```
for (int i = 0; i < np; i++) {
pnodelist.get(i).setSimulationNode(true);
snodelist.get(i).setSimulationNode(true);
}

//Step 4*****************************************************************
//Set the distributions and relationships
//Create the probability nodes p0 p1 p2 p3 px....
for (int i = 0; i < np; i++) {
List parameters = new ArrayList();
parameters.add("0");//Lowerbound
parameters.add("1");//Upperbound
ExtendedNodeFunction uniform = new ExtendedNodeFunction(Uniform.displayName,
parameters);
pnodelist.get(i).setExpression(uniform);
ebn.regenerateNPT(pnodelist.get(i));
}
//Create the auxiliary nodes Sum
String arithstring = new String();
for (int i = 0; i < np; i++) {
String p = "p"+i;
if (i<(np-1)){
arithstring = arithstring + p + "+";
} else {
arithstring = arithstring + p;
}
}
List parameters2 = new ArrayList();
parameters2.add(arithstring);//Arithmatic
ExtendedNodeFunction arith3 = new ExtendedNodeFunction(Arithmetic.displayName,
parameters2);
psum.setExpression(arith3);
ebn.regenerateNPT(psum);

//Create the node N
List parameters = new ArrayList();
parameters.add("0");//Mean
parameters.add("100000000");//Variance
ExtendedNodeFunction normal = new ExtendedNodeFunction(Normal.displayName,
parameters);
nnode.setExpression(normal);
ebn.regenerateNPT(nnode);

//Create the binomial nodes s0 s1 s2 s3 sx....
for (int i = 0; i < np; i++) {
List s_parameters = new ArrayList();
s_parameters.add("n");   //Number of Trails
s_parameters.add("p"+i); //Probability
ExtendedNodeFunction binormal = new ExtendedNodeFunction(Binomial.displayName,
s_parameters);
snodelist.get(i).setExpression(binormal);
ebn.regenerateNPT(snodelist.get(i));
}

//Create the constraint node expression
for (int i = 0; i < cnodelist.size(); i++) {
Constraint tmp = constraintList.get(i);
```

```
if (tmp.type) { //single contraint
String consexpress = new String();
String p           = "p"+Integer.toString(tmp.nodeindex);
String v           = Double.toString(tmp.value);
consexpress        = p + tmp.relation + v;
consexpress        = "if(" + consexpress + ",\"True\",\"False\")";

List parameter = new ArrayList();
parameter.add(consexpress);
ExtendedNodeFunction arith = new ExtendedNodeFunction(Comparative.
displayName, parameter);
cnodelist.get(i).setExpression(arith);
ebn.regenerateNPT(cnodelist.get(i));
} else {
String consexpress = new String();
String p           = "p"+Integer.toString(tmp.nodeindex);
String p2          = "p"+Integer.toString((int)tmp.value);
consexpress        = p + tmp.relation + p2;
consexpress        = "if(" + consexpress + ",\"True\",\"False\")";

List parameter = new ArrayList();
parameter.add(consexpress);
ExtendedNodeFunction arith = new ExtendedNodeFunction(Comparative.
displayName, parameter);
cnodelist.get(i).setExpression(arith);
ebn.regenerateNPT(cnodelist.get(i));
}
}

//Step 5**************************************************************************
//Set evidences
Scenario s = model.getScenarioAtIndex(0);
s.addRealObservation(ebn.getId(), psum.getId(), 1.0);
s.addRealObservation(ebn.getId(), nnode.getId(), trailsum);
for (int i = 0; i < np; i++) {
s.addRealObservation(ebn.getId(), snodelist.get(i).getId(), countList.get(i));
}
for (int i = 0; i < cnodelist.size(); i++) {
s.addHardEvidenceObservation(ebn.getId(), cnodelist.get(i).getId(), cnodelist.
get(i).getExtendedStateAtIndex(0).getId());
}

//Step 6**************************************************************************
//Inference and save the results
long startTime   = System.currentTimeMillis();
model.setSimulationNoOfIterations(numberiter);
model.calculate();
long endTime     = System.currentTimeMillis();
long elapsedTime = endTime - startTime;
System.out.println("time"+elapsedTime);

for (int i = 0; i < np; i++) {
MarginalDataItemList mdil  = model.getMarginalDataStore().
getMarginalDataItemListForNode(ebn, pnodelist.get(i));
MarginalDataItem     mdi   = mdil.getMarginalDataItemAtIndex(0);
double               meanv = mdi.getMeanValue();
returnList.add(meanv);
```

```
}
model.save("Multinormial.cmp");
return returnList;
}
}
```

Appendix B

First Pages of Published Work

Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Incorporating Expert Judgement into Bayesian Network Machine Learning

Yun Zhou[1,2], **Norman Fenton**[1], **Martin Neil**[1], **Cheng Zhu**[2]
[1]Queen Mary University of London, UK
[2]National University of Defense Technology, China
{yun.zhou, norman, martin}@eecs.qmul.ac.uk; chengzhu@nudt.edu.cn

Abstract

We review the challenges of Bayesian network learning, especially parameter learning, and specify the problem of learning with sparse data.We explain how it is possible to incorporate both qualitative knowledge and data with a multinomial parameter learning method to achieve more accurate predictions with sparse data.

1 Review of Bayesian Network Learning

Constructing a Bayesian network (BN) from data is widely accepted as a major challenge in decision-support systems. For many critical risk analysis problems, decisions must be made where there is sparse or no direct historical data to draw upon, or where relevant data is difficult to identify. The challenge is especially acute when the risks involve novel or rare systems and events [Fenton and Neil, 2012] (e.g. think of novel project planning, predicting events like accidents, terrorist attacks, and cataclysmic weather events).

There are two typical categories of problems in learning BNs: one is parameter learning given a fixed graphical structure of the BN; and the other is structure learning, where the BN structure is unknown. Ideally, with sufficient data, classical learning algorithms like BDeu+MLE, PC+MLE or hybrid+MLE [Campos, 2007] can learn BNs that fit the true model in distribution and structure. However, these learning algorithms do not work when there is sparse data. To mitigate this problem, expert judgements are needed to supplement learning.

In the absence of data, experts are usually required to provide strong information like causality between nodes in structure learning, and specific numerical probability values of Node Probability Tables (NPTs) or Dirichlet priors in parameter learning. Such strong judgments can easily cause bias. Studies show that experts are often overconfident in providing qualitative knowledge rather than quantitative estimations [Druzdzel and van der Gaag, 2000]. Typical qualitative knowledge are constraints that limit the number of parents of a node (in structure learning) and equality/inequality relations among parameters in parameter learning; such constraints cut the search space significantly, and help escape local maxima. Because of the potential benefits, there is an increasing research interest in incorporating constraints into structure/parameter learning. Recent reseach developments have focused on triggering the necessary automated calculations and inferences to get more accurate BNs under these constraints.

For parameter learning, some approaches formulate this problem as a general constrained maximization problem, and outline the details of the classification of parameter constraint types ([Niculescu *et al.*, 2006] and [Liao and Ji, 2009]). Unfortunately, these approaches can be extremely inefficient for BNs with a large number of parameters. Nor can they handle exterior constraints among parameters, as discussed in [Feelders and van der Gaag, 2006] and [Tong and Ji, 2008]. The work of [Tong and Ji, 2008] has limited forms of constraints, while and the work of [Feelders and van der Gaag, 2006] can only learn the parameter for binary variables. Hence, our research is focused on the development of an extended BN graphical notation, and associated algorithms, to integrate judgements provided by domain experts in a much richer and less constrained way than the current state-of-the-art of modelling and tools supports.

2 Proposed Solution

For parameter learning, data statistics can be regarded as a sequence of independent Bernoulli experiments on different parameters. Given the number of Bernoulli experiments on a parameter, and observed results, the success probability of each value can be estimated by Bayesian inference. For each parameter in the BN, an auxiliary BN model called multinomial parameter learning model (MPL) can be created [Zhou *et al.*, 2013]. Then constraints can be integrated as additional nodes connected with this MPL.

Before inference, constraints and data observations are transferred as evidence in the separate BN, which are used to update the posterior distributions of target values. Inference refers to the process of computing the discretised posterior marginals of constrained values after obtaining the observations of its constraint. Because the parameter values are continuous, our approach requires the dynamic discretization inference algorithm [Neil *et al.*, 2007] to compute posteriors in this hybrid BN model. This algorithm has been implemented in the Agenarisk toolset [Agenarisk, 2013]. Any continuous node is implemented as a 'simulation' node meaning that its discretization will be calculated dynamically by the algorithm. The work presented here requires such an ac-

International Journal of Approximate Reasoning 55 (2014) 1252–1268

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Bayesian network approach to multinomial parameter learning using data and expert judgments

Yun Zhou [a,b,*], Norman Fenton [a], Martin Neil [a]

[a] Risk and Information Management (RIM) Research Group, Queen Mary University of London, United Kingdom
[b] Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology, PR China

ARTICLE INFO

Article history:
Received 8 May 2013
Received in revised form 17 February 2014
Accepted 26 February 2014
Available online 5 March 2014

Keywords:
Bayesian networks
Multinomial parameter learning
Expert judgments

ABSTRACT

One of the hardest challenges in building a realistic Bayesian Network (BN) model is to construct the node probability tables (NPTs). Even with a fixed predefined model structure and very large amounts of relevant data, machine learning methods do not consistently achieve great accuracy compared to the ground truth when learning the NPT entries (parameters). Hence, it is widely believed that incorporating expert judgments can improve the learning process. We present a multinomial parameter learning method, which can easily incorporate both expert judgments and data during the parameter learning process. This method uses an auxiliary BN model to learn the parameters of a given BN. The auxiliary BN contains continuous variables and the parameter estimation amounts to updating these variables using an iterative discretization technique. The expert judgments are provided in the form of constraints on parameters divided into two categories: linear inequality constraints and approximate equality constraints. The method is evaluated with experiments based on a number of well-known sample BN models (such as *Asia*, *Alarm* and *Hailfinder*) as well as a real-world software defects prediction BN model. Empirically, the new method achieves much greater learning accuracy (compared to both state-of-the-art machine learning techniques and directly competing methods) with much less data. For example, in the software defects BN for a sample size of 20 (which would be considered difficult to collect in practice) when a small number of real expert constraints are provided, our method achieves a level of accuracy in parameter estimation that can only be matched by other methods with much larger sample sizes (320 samples required for the standard machine learning method, and 105 for the directly competing method with constraints).

1. Introduction

Bayesian Networks (BNs) [1,2] are the result of a marriage between graph theory and probability theory, which enable us to model probabilistic and causal relationships for many types of decision-support problems. A BN consists of a directed acyclic graph (DAG) that represents the dependencies among related nodes (variables), together with a set of local probability distributions attached to each node (called a node probability table – NPT – in this paper) that quantify the strengths of these dependencies. BNs have been successfully applied to many real-world problems [3]. However, building realistic and

* Corresponding author at: Risk and Information Management (RIM) Research Group, Queen Mary University of London, United Kingdom.
E-mail addresses: yun.zhou@qmul.ac.uk (Y. Zhou), n.fenton@qmul.ac.uk (N. Fenton), m.neil@qmul.ac.uk (M. Neil).

http://dx.doi.org/10.1016/j.ijar.2014.02.008
0888-613X/

An Extended MPL-C Model for Bayesian Network Parameter Learning with Exterior Constraints

Yun Zhou[1,2,⋆], Norman Fenton[1], and Martin Neil[1]

[1] Risk and Information Management (RIM) Research Group,
Queen Mary University of London, United Kingdom
[2] Science and Technology on Information Systems Engineering Laboratory,
National University of Defense Technology, PR China
{yun.zhou,n.fenton,m.neil}@qmul.ac.uk

Abstract. Lack of relevant data is a major challenge for learning Bayesian networks (BNs) in real-world applications. Knowledge engineering techniques attempt to address this by incorporating domain knowledge from experts. The paper focuses on learning node probability tables using both expert judgment and limited data. To reduce the massive burden of eliciting individual probability table entries (parameters) it is often easier to elicit *constraints* on the parameters from experts. Constraints can be interior (between entries of the same probability table column) or exterior (between entries of different columns). In this paper we introduce the first auxiliary BN method (called MPL-EC) to tackle parameter learning with exterior constraints. The MPL-EC itself is a BN, whose nodes encode the data observations, exterior constraints and parameters in the original BN. Also, MPL-EC addresses (i) how to estimate target parameters with both data and constraints, and (ii) how to fuse the weights from different causal relationships in a robust way. Experimental results demonstrate the superiority of MPL-EC at various sparsity levels compared to conventional parameter learning algorithms and other state-of-the-art parameter learning algorithms with constraints. Moreover, we demonstrate the successful application to learn a real-world software defects BN with sparse data.

Keywords: BN parameter learning, Monotonic causality, Exterior constraints, MPL-EC model.

1 Introduction

Bayesian networks have proven valuable in modeling uncertainty and supporting decision making in practice [1]. However, in many applications there is extremely

⋆ The authors would like to thank the three anonymous reviewers for their valuable comments and suggestions. This work was supported by European Research Council (grant no. ERC-2013-AdG339182-BAYES-KNOWLEDGE). The first author was supported by China Scholarship Council (CSC)/Queen Mary Joint PhD scholarships and National Natural Science Foundation of China (grant no. 61273322).

L.C. van der Gaag and A.J. Feelders (Eds.): PGM 2014, LNAI 8754, pp. 581–596, 2014.

Noname manuscript No.
(will be inserted by the editor)

When and Where to Transfer for Bayes Net Parameter Learning

Yun Zhou · Timothy M. Hospedales · Norman Fenton

Received: date / Accepted: date

Abstract Learning Bayesian networks from sparse data is a major challenge in real-world applications where data are hard to acquire. Transfer learning techniques attempt to address this by leveraging data from different but related problems. For example, it may be possible to exploit medical diagnosis data from a different country. A challenge with this approach is heterogeneous relatedness to the target, both within and across source networks. In this paper we introduce the first Bayesian network parameter transfer learning (BNPTL) algorithm to reason about both network and fragment relatedness. BNPTL addresses (i) how to find the most relevant source network and network fragments to transfer, and (ii) how to fuse source and target parameters in a robust way. In addition to improving target task performance, explicit reasoning allows us to diagnose network and sub-graph relatedness across BNs, even if latent variables are present, or if their state space is heterogeneous. This is important in some applications where relatedness itself is an output of interest. Experimental results demonstrate the superiority of BNPTL at various sparsity and source relevance levels compared to single task learning and other state-of-the-art parameter transfer methods. Moreover, we demonstrate successful application to real-world medical case studies.

Keywords Bayesian networks parameter learning · Transfer learning · Bayesian model comparison · Bayesian model averaging

F. Author
RIM Group, Electronic Engineering and Computer Science
Queen Mary University, London, E1 4NS
Science and Technology on Information Systems Engineering Laboratory
National University of Defense Technology, Changsha, 410072
Tel.: +44-07429481207
E-mail: yun.zhou@qmul.ac.uk

S. Author and T. Author
RIM Group, Electronic Engineering and Computer Science
Queen Mary University, London, E1 4NS
E-mail: t.hospedales, n.fenton@qmul.ac.uk

An Empirical Study of Bayesian Network Parameter Learning with Exterior Constraints

Yun Zhou[a,b,*], Norman Fenton[a]

[a] *Risk and Information Management (RIM) Research Group, Queen Mary University of London*
[b] *Science and Technology on Information Systems Engineering Laboratory, National University of Defense Technology*

Abstract

Learning accurate Bayesian networks (BNs) is a key challenge in real-world applications, especially when training data are hard to acquire. The conventional way to mitigate this challenge in parameter learning is to introduce domain knowledge/expert judgements. Recently, the idea of qualitative constraints has been introduced to improve the BN parameter learning accuracy. In this approach, the exterior parameter constraints (between CPT entries of different parent state configurations) are encoded in the edges/structures of BNs with ordinary variables. However, no previous work has investigated the extent to which such constraints exist in the standard BN repository. This paper examines such constraints in each edge of the BNs from the standard repository. Experimental results indicate such constraints fully or partially exist in all these BNs, and our slightly improved constrained optimization algorithm achieves excellent parameter learning performance, especially in large BNs. These results can be used for guiding when to employ exterior constraints in parameter estimation. This has the potential to benefit many real-world case studies in decision support and risk analysis.

Key words:

BN parameter learning; Monotonic causality; Exterior constraints; Empirical

*Corresponding author

Email addresses: yun.zhou@qmul.ac.uk (Yun Zhou), n.fenton@qmul.ac.uk (Norman Fenton)

Probabilistic Graphical Models Parameter Learning with Transferred Prior and Constraints

Anonymous Authors

Abstract

Learning accurate Bayesian networks (BNs) is a key challenge in real-world applications, especially when training data are hard to acquire. Two approaches have been used to address this challenge: 1) introducing expert judgements and 2) transferring knowledge from related domains. This is the first paper to present a generic framework that combines both approaches to improve BN parameter learning. This framework is built upon an extended multinomial parameter learning model, that itself is an auxiliary BN. It serves to integrate both knowledge transfer and expert constraints. Experimental results demonstrate improved accuracy of the new method on a variety of benchmark BNs, showing its potential to benefit many real-world problems.

1 Introduction

Directed probabilistic graphical models, also known as Bayesian networks (BNs), are a natural framework for modelling causal relationships among variables in many real-world problems, such as medical symptom diagnosis (Velikova et al., 2014) and software defect prediction (Fenton and Neil, 2014). However, in problem domains with limited or no relevant training data, there are major challenges in accurately learning BN parameters (Friedman et al., 1999).

There are several methods for handling parameter learning with limited or no relevant data, described in a rich literature of books, articles and software packages, which are briefly summarized in (Druzdel and Van Der Gaag, 2000; Neapolitan, 2004; O'Hagan et al., 2006). Without considering any domain knowledge, the simplest learning approaches usually fail to accurately estimate parameters in a sparse dataset. To mitigate this problem, it may be possible to elicit numerical assessments from expert judgements, but this process is inefficient and error-prone.

Researchers have shown that experts tend to feel more comfortable providing qualitative or semi-numerical judgments (Feelders and van der Gaag, 2006) with less cognitive effort. Such judgments expressed as constraints between parameters of interest (e.g. "the probability of people getting cancer is smaller than 1%") are more reliable than numerical assessments, and have drawn considerable attention recently. In the work of (Zhou et al., 2014a), these kinds of constraints are modelled as nodes in an auxiliary BN model called MPL-C (Multinomial Parameter Learning model with Constraints), which includes nodes modelling training data statistics. The MPL-C improves parameter estimation accuracy by constraining the estimation with the expert constraints.

An alternative approach to improving BN learning in sparse data situations is to transfer knowledge from different but related BNs that may have more training data available (Luis et al., 2010). For example, transferring knowledge from the same medical diagnosis network learned in a different country. This can be effective if data for one or more sufficiently related source domains is available. However, the practical limitation is that transfer is contingent on availability of suitable related sources, and the relatedness of each source to the target task may not be known in advance. Estimating relatedness is thus important but challenging in practice, particularly when there are multiple potential sources of possibly varying relatedness.

While incorporating either parameter constraints or transfer learning from related data in source domains can improve parameter estimation accuracy, there exists no generic learning framework to synergistically exploit the benefits of both approaches. Achieving this is non-trivial because typical approaches to transfer (Luis et al., 2010) and to constrained learning (Zhou et al., 2014a) use very different formalisations. In this

Bibliography

Abramson, B., Brown, J., Edwards, W., Murphy, A., Winkler, R.L., 1996. Hailfinder: A Bayesian system for forecasting severe weather. International Journal of Forecasting 12, 57–71.

Akaike, H., 1998. A Bayesian analysis of the minimum AIC procedure, in: Selected Papers of Hirotugu Akaike. Springer, pp. 275–280.

Al Nasseri, A., Tucker, A., de Cesare, S., 2014. Big data analysis of stocktwits to predict sentiments in the stock market, in: Proceedings of the 17th International Conference on Discovery Science, Springer. pp. 13–22.

Altendorf, E.E., Restificar, A.C., Dietterich, T.G., 2005. Learning from sparse data by exploiting monotonicity constraints, in: Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, pp. 18–26.

Ancel, E., Shih, A.T., Jones, S.M., Reveley, M.S., Luxhøj, J.T., Evans, J.K., 2014. Predictive safety analytics: inferring aviation accident shaping factors and causation. Journal of Risk Research , 1–24.

Ban, S.S., Pressey, R.L., Graham, N.A., 2014. Assessing interactions of multiple stressors when data are limited: A Bayesian belief network applied to coral reefs. Global Environmental Change 27, 64–72.

Barber, D., 2012. Bayesian Reasoning and Machine Learning. Cambridge University Press.

Bartlett, M., Cussens, J., 2015. Integer linear programming for the Bayesian network structure learning problem. Artificial Intelligence .

Beinlich, I.A., Suermondt, H.J., Chavez, R.M., Cooper, G.F., 1989. The ALARM monitoring system: A case study with two probabilistic inference techniques for Belief networks. Springer.

Binder, J., Koller, D., Russell, S., Kanazawa, K., 1997. Adaptive probabilistic networks with hidden variables. Machine Learning 29, 213–244.

Bouckaert, R.R., 1993. Probabilistic network construction using the minimum description length principle, in: Symbolic and Quantitative Approaches to Reasoning and Uncertainty. Springer, pp. 41–48.

Bradley, A.P., 1997. The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition 30, 1145–1159.

Brenner, E., Sontag, D., 2013. Sparsityboost: A new scoring function for learning Bayesian network structure, in: Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence, pp. 112–122.

Broeck, G.V.d., Mohan, K., Choi, A., Pearl, J., 2014. Efficient algorithms for Bayesian network parameter learning from incomplete data. arXiv preprint arXiv:1411.7014 .

Buntine, W.L., 1996. A guide to the literature on learning probabilistic networks from data. Knowledge and Data Engineering, IEEE Transactions on 8, 195–210.

de Campos, C.P., Cozman, F.G., 2005. Belief updating and learning in semi-qualitative probabilistic networks, in: Proceedings of the 21st Conference in Uncertainty in Artificial Intelligence, pp. 153–160.

de Campos, C.P., Ji, Q., 2008. Improving Bayesian network parameter learning using constraints, in: Proceedings of the 19th International Conference on Pattern Recognition, pp. 1–4.

de Campos, C.P., Ji, Q., 2010. Properties of Bayesian Dirichlet scores to learn Bayesian network structures, in: Proceedings of the 24th AAAI Conference on Artificial Intelligence, pp. 431–436.

de Campos, C.P., Ji, Q., 2011. Efficient structure learning of Bayesian networks using constraints. Journal of Machine Learning Research 12, 663–689.

de Campos, C.P., Tong, Y., Ji, Q., 2008. Constrained maximum likelihood learning of Bayesian networks for facial action recognition, in: Proceedings of the 10th European Conference on Computer Vision. Springer, pp. 168–181.

de Campos, C.P., Zeng, Z., Ji, Q., 2009a. Structure learning of Bayesian networks using constraints, in: Proceedings of the 26th International Conference on Machine Learning, ACM. pp. 113–120.

de Campos, C.P., Zhang, L., Tong, Y., Ji, Q., 2009b. Semi-qualitative probabilistic networks in computer vision problems. Journal of Statistical Theory and Practice 3, 197–210.

Campos, L.M., 2007. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. Journal of Machine Learning Research 7, 21–49.

Cano, A., Masegosa, A.R., Moral, S., 2011. A method for integrating expert knowledge when learning Bayesian networks from data. Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 41, 1382–1394.

Chang, R., Stetter, M., Brauer, W., 2008. Quantitative inference by qualitative semantic knowledge mining with Bayesian model averaging. Knowledge and Data Engineering, IEEE Transactions on 20, 1587–1600.

Chickering, D.M., 1996. Learning Bayesian networks is NP-complete, in: Learning from data. Springer, pp. 121–130.

Cooper, G.F., 1990. The computational complexity of probabilistic inference using Bayesian belief networks. Artificial Intelligence 42, 393–405.

Cooper, G.F., Herskovits, E., 1992. A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9, 309–347.

Cussens, J., 2011. Bayesian network learning with cutting planes, in: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, AUAI Press. pp. 153–160.

Dai, W., Xue, G., Yang, Q., Yu, Y., 2007. Transferring naive Bayes classifiers for text classification, in: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, pp. 540–545.

Daly, R., Shen, Q., Aitken, S., 2011. Learning Bayesian networks: approaches and issues. The Knowledge Engineering Review 26, 99–157.

Darwiche, A., 2009. Modeling and reasoning with Bayesian networks. Cambridge University Press.

Davis, J., Domingos, P., 2009. Deep transfer via second-order Markov logic, in: Proceedings of the 26th International Conference on Machine Learning, pp. 217–224.

Diez, F.J., 1993. Parameter adjustment in Bayes networks. the generalized noisy OR-gate, in: Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann. pp. 99–105.

Diez, F.J., Druzdzel, M.J., 2006. Canonical probabilistic models for knowledge engineering. Technical Report. Technical Report CISIAD-06-01, UNED, Madrid.

Druzdzel, M.J., 1989. Verbal uncertainty expressions: Literature review. Pittsburgh, PA: Carnegie Mellon University, Department of Engineering and Public Policy .

Druzdzel, M.J., van der Gaag, L.C., 1995. Elicitation of probabilities for Belief networks: Combining qualitative and quantitative information, in: Proceedings of the 11th Conference on Uncertainty in Artificial Intelligence, pp. 141–148.

Druzdzel, M.J., Henrion, M., 1993. Efficient reasoning in qualitative probabilistic networks, in: Proceedings of the 11th National Conference on Artificial Intelligence, Washington, pp. 548–553.

Duan, L., Tsang, I.W., Xu, D., Chua, T.S., 2009. Domain adaptation from multiple sources via auxiliary classifiers, in: Proceedings of the 26th International Conference on Machine Learning, pp. 289–296.

Duval, R., 1993. Bootstrapping: A nonparametric approach to statistical inference. Sage.

Eaton, E., desJardins, M., Lane, T., 2008. Modeling transfer relationships between learning tasks for improved inductive transfer, in: Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases-Part I, Springer-Verlag. pp. 317–332.

Ezell, B.C., Bennett, S.P., Von Winterfeldt, D., Sokolowski, J., Collins, A.J., 2010. Probabilistic risk analysis and terrorism risk. Risk Analysis 30, 575–589.

Feelders, A., van der Gaag, L., 2006. Learning Bayesian network parameters under order constraints. International Journal of Approximate Reasoning 42, 37–53.

Fenton, N., Neil, M., 2012. Risk Assessment and Decision Analysis with Bayesian Networks. CRC Press, New York.

Fenton, N., Neil, M., Caballero, J.G., 2007. Using ranked nodes to model qualitative judgments in Bayesian networks. Knowledge and Data Engineering, IEEE Transactions on 19, 1420–1432.

Fenton, N., Neil, M., Marsh, W., Hearty, P., Radliński, Ł., Krause, P., 2008. On the effectiveness of early life cycle defect prediction with Bayesian nets. Empirical Software Engineering 13, 499–537.

Fenton, N.E., Neil, M., 1999. A critique of software defect prediction models. Software Engineering, IEEE Transactions on 25, 675–689.

van der Gaag, L.C., Renooij, S., Geenen, P.L., 2006. Lattices for studying monotonicity of Bayesian networks, in: Proceedings of the 3rd European Workshop on Probabilistic Graphical Models, pp. 99–106.

van der Gaag, L.C., Tabachneck-Schijf, H.J., Geenen, P.L., 2009. Verifying monotonicity of Bayesian networks with domain experts. International Journal of Approximate Reasoning 50, 429 – 436.

Hand, D.J., Till, R.J., 2001. A simple generalisation of the area under the ROC curve for multiple class classification problems. Machine learning 45, 171–186.

Heckerman, D., Geiger, D., Chickering, D.M., 1995. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning 20, 197–243.

Henrion, M., 1988. Practical issues in constructing a Bayes belief network. International Journal of Approximate Reasoning 2, 337.

Huang, J., Smola, A.J., Gretton, A., Borgwardt, K.M., Scholkopf, B., 2007. Correcting sample selection bias by unlabeled data, in: Advances in Neural Information Processing Systems, pp. 601–608.

Karaolis, M., Moutiris, J., Hadjipanayi, D., Pattichis, C., 2010. Assessment of the risk factors of coronary heart events based on data mining with decision trees. Information Technology in Biomedicine, IEEE Transactions on 14, 559–566.

Khodakarami, V., Fenton, N., Neil, M., 2007. Project scheduling: Improved approach to incorporate uncertainty using Bayesian networks. Project Management Journal 38, 1–39.

Koller, D., Friedman, N., 2009. Probabilistic graphical models: principles and techniques.

Korb, K.B., Nicholson, A.E., 2010. Bayesian Artificial Intelligence. CRC Press, New York.

Kullback, S., Leibler, R.A., 1951. On information and sufficiency. The Annals of Mathematical Statistics , 79–86.

Langseth, H., Nielsen, T.D., Rumí, R., Salmerón, A., 2009. Inference in hybrid Bayesian networks. Reliability Engineering & System Safety 94, 1499–1509.

Larrañaga, P., Poza, M., Yurramendi, Y., Murga, R.H., Kuijpers, C.M.H., 1996. Structure learning of Bayesian networks by genetic algorithms: A performance analysis of control parameters. Pattern Analysis and Machine Intelligence, IEEE Transactions on 18, 912–926.

Lauritzen, S., Spiegelhalter, D., 1988. Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society. Series B (Methodological) , 157–224.

Lauritzen, S.L., Jensen, F., 2001. Stable local computation with conditional Gaussian distributions. Statistics and Computing 11, 191–203.

Liao, W., Ji, Q., 2009. Learning Bayesian network parameters under incomplete data with domain knowledge. Pattern Recognition 42, 3046–3056.

López-Cruz, P.L., Larrañaga, P., DeFelipe, J., Bielza, C., 2014. Bayesian network modeling of the consensus between experts: An application to neuron classification. International Journal of Approximate Reasoning 55, 3–22.

Luis, R., Sucar, L.E., Morales, E.F., 2010. Inductive transfer for learning Bayesian networks. Machine learning 79, 227–255.

Malone, B., Yuan, C., Hansen, E., Bridges, S., 2011. Improving the scalability of optimal Bayesian network learning with frontier breadth-first branch and bound search, in: Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence, pp. 479–488.

Mascaro, S., Nicholso, A.E., Korb, K.B., 2014. Anomaly detection in vessel tracks using Bayesian networks. International Journal of Approximate Reasoning 55, 84–98.

Masegosa, A.R., Moral, S., 2013. An interactive approach for Bayesian network learning using domain/expert knowledge. International Journal of Approximate Reasoning 54, 1168–1181.

Mihalkova, L., Huynh, T., Mooney, R.J., 2007. Mapping and revising Markov logic networks for transfer learning, in: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, pp. 608–614.

Mihalkova, L., Mooney, R.J., 2009. Transfer learning from minimal target data by mapping across relational domains, in: Proceedings of the 21st International Jont Conference on Artifical Intelligence, pp. 1163–1168.

Mohan, K., Pearl, J., Tian, J., 2013. Graphical models for inference with missing data, in: Advances in Neural Information Processing Systems, pp. 1277–1285.

Mosteller, F., Youtz, C., et al., 1990. Quantifying probabilistic expressions. Statistical Science 5, 2–12.

Murphy, K., 1998. Inference and learning in hybrid Bayesian networks. University of California, Berkeley, Computer Science Division.

Murphy, K.P., 2012. Machine Learning: A Probabilistic Perspective. The MIT Press, Cambridge.

Neil, M., Chen, X., Fenton, N., 2012. Optimizing the calculation of conditional probability tables in hybrid Bayesian networks using binary factorization. Knowledge and Data Engineering, IEEE Transactions on 24, 1306–1312.

Neil, M., Tailor, M., Marquez, D., 2007. Inference in hybrid Bayesian networks using dynamic discretization. Statistics and Computing 17, 219–233.

Niculescu, R.S., Mitchell, T., Rao, B., 2006. Bayesian network learning with parameter constraints. The Journal of Machine Learning Research 7, 1357–1383.

Niculescu-mizil, A., Caruana, R., 2007. Inductive transfer for Bayesian network structure learning, in: Proceedings of the 11th International Conference on Artificial Intelligence and Statistics, pp. 1–8.

Nie, S., Mauá, D.D., de Campos, C.P., Ji, Q., 2014. Advances in learning Bayesian networks of bounded treewidth, in: Advances in Neural Information Processing Systems, pp. 2285–2293.

Oates, C.J., Smith, J.Q., Mukherjee, S., Cussens, J., 2014. Exact estimation of multiple directed acyclic graphs. arXiv preprint arXiv:1404.1238 .

Ogunsanya, O.V., 2012. Decision support using Bayesian networks for clinical decision making. Ph.D. thesis. Queen Mary, University of London.

Overill, R.E., Zhang, E.P., Chow, K.P., 2012. Multi-parameter sensitivity analysis of a Bayesian network from a digital forensic investigation, in: Proceedings of the Conference on Digital Forensics, Security and Law, pp. 129–140.

Oyen, D., Lane, T., 2012. Leveraging domain knowledge in multitask Bayesian network structure learning, in: Proceedings of the 26th AAAI Conference on Artificial Intelligence, pp. 1091–1097.

Pan, S.J., Yang, Q., 2010. A survey on transfer learning. Knowledge and Data Engineering, IEEE Transactions on 22, 1345–1359.

Pearl, J., 1988. Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann.

Pearl, J., 2011. Bayesian networks. Department of Statistics, UCLA .

Renooij, S., van der Gaag, L.C., 1999. Enhancing QPNs for trade-off resolution, in: Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc.. pp. 559–566.

Renooij, S., van der Gaag, L.C., 2002. From qualitative to quantitative probabilistic networks, in: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann Publishers Inc.. pp. 422–429.

Renooij, S., Van der Gaag, L.C., 2008. Enhanced qualitative probabilistic networks for resolving trade-offs. Artificial Intelligence 172, 1470–1494.

Renooij, S., Van der Gaag, L.C., Parsons, S., 2002. Context-specific sign-propagation in qualitative probabilistic networks. Artificial Intelligence 140, 207–230.

Russell, S., Norvig, P., 2009. Artificial intelligence: A modern approach .

Sá, A.G., Pappa, G.L., Pereira, A., 2014. Generating personalized algorithms to learn Bayesian network classifiers for fraud detection in web transactions, in: Proceedings of the 20th Brazilian Symposium on Multimedia and the Web, ACM. pp. 179–186.

Scanagatta, M., de Campos, C.P., Zaffalon, M., 2014. Min-BDeu and Max-BDeu scores for learning Bayesian networks, in: Probabilistic Graphical Models. Springer, pp. 426–441.

Schwarz, G., et al., 1978. Estimating the dimension of a model. The Annals of Statistics 6, 461–464.

Seah, C.W., Ong, Y.S., Tsang, I., 2013a. Combating negative transfer from predictive distribution differences. Cybernetics, IEEE Transactions on 43, 1153–1165.

Seah, C.W., Tsang, I., Ong, Y.S., 2013b. Transfer ordinal label learning. Neural Networks and Learning Systems, IEEE Transactions on 24, 1863–1876.

Selen, U., Jaime, C., 2011. Feature selection for transfer learning, in: Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases-Volume Part III, Springer-Verlag. pp. 430–442.

Settles, B., 2010. Active learning literature survey. University of Wisconsin, Madison 52, 1–11.

Silander, T., Kontkanen, P., Myllymaki, P., 2012. On sensitivity of the MAP Bayesian network structure to the equivalent sample size parameter. arXiv preprint arXiv:1206.5293 .

Steck, H., Jaakkola, T.S., 2002. On the Dirichlet prior and Bayesian regularization, in: Advances in Neural Information Processing Systems, pp. 697–704.

Su, J., Zhang, H., Ling, C.X., Matwin, S., 2008. Discriminative parameter learning for Bayesian networks, in: Proceedings of the 25th International Conference on Machine Learning, ACM. pp. 1016–1023.

Tong, S., Koller, D., 2001a. Active learning for parameter estimation in Bayesian networks. Advances in Neural Information Processing Systems , 647–653.

Tong, S., Koller, D., 2001b. Active learning for structure in Bayesian networks, in: Proceedings of the 17th International Joint Conference on Artificial Intelligence, Morgan Kaufmann Publishers Inc.. pp. 863–869.

Torrey, L., Shavlik, J., 2009. Transfer learning. Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques 1, 1–22.

Tsamardinos, I., Brown, L.E., Aliferis, C.F., 2006. The max-min hill-climbing Bayesian network structure learning algorithm. Machine Learning 65, 31–78.

Velikova, M., van Scheltinga, J.T., Lucas, P.J., Spaanderman, M., 2014. Exploiting causal functional relationships in Bayesian network modelling for personalised healthcare. International Journal of Approximate Reasoning 55, 59–73.

Wallace, C., Korb, K.B., Dai, H., 1996. Causal discovery via MML, in: Proceedings of the 13th International Conference on Machine Learning, pp. 516–524.

Wellman, M.P., 1990. Fundamental concepts of qualitative probabilistic networks. Artificial Intelligence 44, 257–303.

Woudenberg, S., van der Gaag, L., 2015. Propagation effects of model-calculated probability values in Bayesian networks. International Journal of Approximate Reasoning 61, 1–15.

Xiang, Y., Jia, N., 2007. Modeling causal reinforcement and undermining for efficient CPT elicitation. Knowledge and Data Engineering, IEEE Transactions on 19, 1708–1718.

Xiang, Y., Truong, M., 2014. Acquisition of causal models for local distributions in Bayesian networks. Cybernetics, IEEE Transactions on 44, 1591–1604.

Yang, S., Natarajan, S., 2013. Knowledge intensive learning: Combining qualitative constraints with causal independence for parameter learning in probabilistic models, in: Machine Learning and Knowledge Discovery in Databases. Springer, pp. 580–595.

Yet, B., Perkins, Z., Fenton, N., Tai, N., Marsh, W., 2014. Not just data: A method for improving prediction with knowledge. Journal of Biomedical Informatics 48, 28 – 37.

Yuan, C., Malone, B., Wu, X., 2011. Learning optimal Bayesian networks using A* search, in: Proceedings of the 22nd International Joint Conference on Artificial Intelligence, pp. 2186–2191.

Zagorecki, A., Druzdzel, M.J., 2013. Knowledge engineering for Bayesian networks: How common are Noisy-MAX distributions in practice? Systems, Man, and Cybernetics: Systems, IEEE Transactions on 43, 186–195.

Zhou, Y., Fenton, N., 2015. An empirical study of Bayesian network parameter learning with exterior constraints. Submitted to International Journal of Approximate Reasoning .

Zhou, Y., Fenton, N., Neil, M., 2014a. Bayesian network approach to multinomial parameter learning using data and expert judgments. International Journal of Approximate Reasoning 55, 1252 – 1268.

Zhou, Y., Fenton, N., Neil, M., 2014b. An extended MPL-C model for Bayesian network parameter learning with exterior constraints, in: van der Gaag, L., Feelders, A. (Eds.), Probabilistic Graphical Models. Springer International Publishing. volume 8754 of *Lecture Notes in Computer Science*, pp. 581–596.

Zhou, Y., Fenton, N., Neil, M., Zhu, C., 2013. Incorporating expert judgement into Bayesian network machine learning, in: Proceedings of the 23rd International Joint Conference on Artificial Intelligence, pp. 3249–3250.