Exploiting Parameter Domain Knowledge for Learning in Bayesian Networks

Radu Stefan Niculescu

July 2005

CMU-CS-05-147

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

This research was sponsored by the SRI International under subcontract 03-0002111 for the Department of the Interior, the National Science Foundation under grant nos. CCR-0085982 and CCR-0122581, a Merck Corporation Graduate Fellowship and a generous gift from Siemens Medical Solutions. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

Keywords: Constraint Optimization, Domain Knowledge, Graphical Models

Abstract

The task of learning models for many real-world problems requires researchers to incorporate problem Domain Knowledge into the learning algorithms because there is rarely enough training data to enable accurate learning of the structures and underlying relationships in the problem. Domain Knowledge comes in many forms. Domain Knowledge about relevance of variables (Feature Selection) can help us ignore certain variables when building our model. Domain Knowledge specifying conditional independencies among variables can guide our search over possible model structures. This thesis presents a theoretical framework for incorporating a different kind of knowledge into learning algorithms for Bayesian Networks: Domain Knowledge about relationships among parameters.

We develop a unified framework for incorporating general Parameter Domain Knowledge constraints in learning procedures for Bayesian Networks by formulating this as a constrained optimization problem. We solve this problem using iterative algorithms based on Newton-Raphson method for approximating the solutions of a system of equations. We approach learning from both a frequentist and a Bayesian point of view, from both complete and incomplete data.

We also derive closed form solutions for our estimators for several types of Parameter Domain Knowledge: parameter sharing, as well as sharing properties of groups of parameters (sum sharing and ratio sharing). While models like Module Networks, Dynamic Bayes Nets and Context Specific Independence models share parameters at either conditional probability table or conditional distribution (within one table) level, our framework is more flexible, allowing sharing at parameter level, across conditional distributions of different lengths and across different conditional probability tables. Other results include several formal guarantees about our estimators and methods for automatically learning domain knowledge.

To validate our theory, we carry out experiments showing the benefits of taking advantage of domain knowledge for modelling the fMRI signal during a cognitive task. Additional experiments on synthetic data are also performed.

Acknowledgements

I would like to dedicate this thesis to my parents, Rodica and Stefan, for continuously encouraging and supporting my passion for mathematics and computer science starting at a very early age. I am also very grateful to Rich Caruana for helping me decide to pursue a PhD in Machine Learning.

I would like to thank my advisor, Tom Mitchell, for closely supervising my research and for giving me the freedom to explore different areas of this exciting field which is Graphical Models. Many thanks to the other members of my thesis committee: John Lafferty, Andrew Moore and Bharat Rao for their useful comments and suggestions during the development of this research. I would also like to thank Russ Greiner, Zoubin Ghahramani and William Cohen for sharing their expertise with me in the earlier stages of my thesis. I want to thank Daniel Sleator for helping me develop my algorithms background, which proved very beneficial in my research.

Special thanks to Siemens Medical Solutions for providing part of the funding for my research and to Bharat Rao for hiring me as an intern with Siemens during the last four summers. Other members of the Computer Aided Diagnosis group at Siemens, Sathyakama Sandilya and Balaji Krishnapuram, also provided very helpful remarks regarding my research.

Finally I would like to thank my CMU friends. My officemate Mihai provided a lot of advice about the PhD student's life, work at CMU and also the structure of this thesis. I am grateful to Vahe, one of my best friends, for introducing me to my favorite hobby, salsa dancing. I particularly enjoyed the company of Monica, Lucian and Kuki during our weekly movie nights. Many thanks to Emil for his advice during my coursework and for being a great roommate all these years.

Contents

Chapter 1

Introduction

1.1 Motivation

Probabilistic Models have become increasingly popular in the last decades because of the need to characterize the non-deterministic nature of relationships among variables describing many real world domains. Among these models, *Bayesian Networks* have received a tremendous amount of interest because of their ability to compactly encode uncertainty about random variables and to efficiently deal with missing data. Another major advantage of Bayesian Networks is that they are relatively easy to interpret by a non-expert, unlike Neural Networks or Support Vector Machines. Applications of Bayesian Networks include medical diagnosis, stock market prediction, fraud detection, intelligent troubleshooting and language modelling.

A *Bayesian Network* [Hec99, Pea88] is a model that compactly represents the probability distribution over a set of random variables. It consists of two components: a structure and a set of parameters. The structure is a Directed Acyclic Graph where one can think of the edges as cause-effect relationships. The parameters describe how each variable relates probabilistically to its parents. Intuitively, the parameters describe how probable each effect is given a combination of direct causes.

Figure 1.1 shows a simplified version of a Bayesian Network that can be used for disease diagnosis. Typically, a diagnosis is reached by looking at a combination of risk factors and symptoms. Risk factors like *Smoking* (whether or not the patient smokes), *FHxMI* (whether or not the patient has a family history positive for heart attack), *Pollution* (whether or not the area where the patient lives has high air pollution) can all determine the presence of a disease. Given a disease is present, the patient may or may not show certain symptoms: *Fever*, *Chest Pain*, *Vomiting*.

The task of learning models for many real-world problems requires researchers to incorporate problem *Domain Knowledge* into the learning algorithm because there is rarely enough training data to enable the learning of the structures and underlying relationships in the problem. Domain

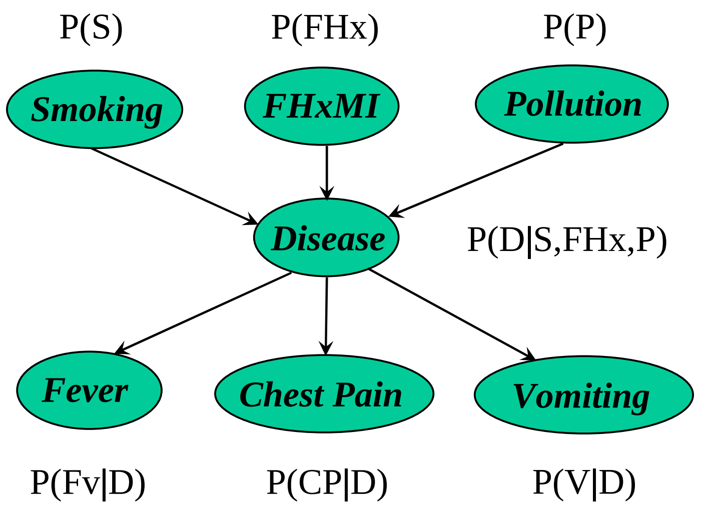

Figure 1.1: A simplified version of a Bayesian Network which models the interaction between risk factors, diseases and symptoms for the purpose of disease diagnosis in Emergency Room patients. The variables involved are: *Smoking(S)*, Family History of Heart Attack *FHxMI (FHx)*, *Pollution(P)*, *Disease(D)*, *Fever(Fv)*, *Chest Pain(CP)* and *Vomiting (V)*.

Knowledge comes in many forms. A domain expert can provide Domain Knowledge about relevance of certain variables, also called Feature Selection, that can help us ignore certain variables when building our model. Domain Knowledge specifying conditional independencies among variables can both guide our search over possible Bayesian Network structures and speed up inference. Both these forms of Domain Knowledge have been extensively studied.

This thesis presents a theoretical framework for incorporating a different kind of knowledge into *Parameter Learning* algorithms for Bayesian Networks: Domain Knowledge about relationships among parameters. *Parameter Learning* in a Bayesian Network is defined as the problem of estimating the parameters of that network from a dataset of training cases. These cases can be either fully or partially observable.

To see why one would need to take advantage of *Parameter Domain Knowledge*, consider the network in the above example. In a real world situation, we can have tens of potential risk factors and hundreds of potential symptoms. Also, the *Disease* variable can take values in very large set. With only 20 binary risk factors, the number of parameters of the diagnosis Bayesian Network easily runs in the millions. Unfortunately, clean and complete medical data is extremely hard to come by in a quantity sufficient to allow us to estimate these parameters accurately. However, medical Domain Knowledge is plentiful and it can come directly from physicians or can be extracted from written/online medical material. For example, a doctor may say: *All the other risk factors can be ignored (have little additional influence) when deciding on a diagnosis of Heart Attack given that the* *patient is a Smoker with a positive Family History of Heart Attack.* Knowledge coming from medical books may state: *A patient with Heart Attack exhibits chest pain and, very frequently, vomiting.* While in the first case, domain knowledge asserts equality of a large number of parameters, in the second case it asserts a deterministic relation between *Heart Attack* and *Chest Pain*.

First, *Parameter Domain Knowledge* can help by shrinking the space in which the parameters can take values. In the case of equality constraints, we achieve dimensionality reduction in this space (as we noticed in the above examples). Inequality constraints can significantly reduce the volume of the feasible region in the space of parameters in the case when this region is bounded. This is the case with Bayesian Networks that model only discrete variables, because each parameter is a probability between 0 and 1. Second, since Parameter Domain Knowledge has the effect of shrinking the space of allowed parameters and since the amount of training data does not change, intuitively one would expect that algorithms that know how to take advantage of Parameter Domain Knowledge will produce lower variance estimators, which would be a great plus when training data is scarce.

Currently, most popular ways of representing *Parameter Domain Knowledge* are: *Dirichlet Priors* and their variants and *Parameter Sharing* (HMMs, Module Networks, Context Specific Independence), each of them having serious limitations. With Dirichlet Priors, one can not represent even simple equality constraints between parameters. Generalizations would allow this simple kind of constraint by using priors on the parameters of the Dirichelet Prior but in this case the marginal likelihood can not be computed in closed anymore. Second, when the Bayesian Network has a very large number of parameters, it is often beyond the expert's ability to specify a full Dirichlet Prior. In models like Module Networks or HMMs, parameter sharing happens at the level of conditional probability table while Context Specific Independence can specify parameter sharing at the level of conditional probability distribution within the same table. No such model allows sharing at parameter level of granularity nor it is able to represent more complicated parameter constraints. We will discuss these forms of Parameter Domain Knowledge in more detail in Chapter 2.

In this thesis we will present a unifying framework for incorporating Parameter Domain Knowledge to perform automatic Parameter Learning in Bayesian Networks. While this framework uses an iterative procedure to approximate the parameters, we will also illustrate it with several Domain Knowledge types where we can compute the estimators in closed form. In particular, we define a Parameter Sharing Domain Knowledge type and show how current models that use Parameter Sharing assumptions can all be represented by this type. Examples on both real world and synthetic data will demonstrate the benefits of taking advantage of Parameter Domain Knowledge when compared to baseline models.

1.2 Research Approach

The derivation of our results relies heavily on optimization and approximation techniques. We will formulate maximum likelihood parameter learning from complete data as a constrained maximization problem. We will solve this optimization problem using Karush-Kuhn-Tucker theorem. This is a generalization of Lagrange Multipliers theorem, which looks for a set of inequality constraints that become equalities at the local optimum. As expected, the system of equations which results from Karush-Kuhn-Tucker theorem may be difficult to solve in closed form in the general case. However, this system has the same number of equations as variables and its solutions can be found using the Newton-Raphson iterative method. For this method to work, we require that our Parameter Domain Knowledge constraints be represented as twice differentiable functions with continuous second derivatives.

The dimensionality of the optimization problem can make the above approach prohibitive. Fortunately, in practice, most given constraints involve only a small fraction of the total number of parameters. In addition, the objective function (likelihood function in general) is nicely decomposable and therefore we will be able to split the initial maximization problem into a set of many independent maximization subproblems, each with its own set of constraints. These subproblems have much lower dimensionality and therefore can be solved more easily with the above mentioned method.

There are several approaches to parameter learning: from either a frequentist or a Bayesian point of view, from either complete or incomplete data. The above method performs learning from complete data from a frequentist point of view. In the case of incomplete data, we present several ways to perform Maximum Likelihood estimation based on methods similar to the ones for complete data. In particular, we notice that extending the Expectation Maximization algorithm for discrete Bayesian Networks in the presence of Parameter Domain Knowledge constraints is just a matter of applying in the M-Step the Maximum Likelihood estimators on the expected counts computed in the E-Step. From a Bayesian point of view, we define *Constrained Parameter Priors* that obey the Parameter Domain Knowledge and show how the normalization constant can be computed via a sampling algorithm. Based on these priors, we then discuss how one can perform Maximum Aposteriori estimation and Bayesian model averaging for both complete and incomplete data.

While the above methods work for general constraints, it would be preferable to be able to compute, in one step, closed form solutions for both the parameter estimators and the normalization constants of the *Constrained Parameter Priors*. Unfortunately, this task is not always possible, simply because of the fact that there is no known closed form solution for polynomial equations of degree higher than four. Three chapters of this thesis will be dedicated to the derivation of closed form estimators for several types of domain knowledge. We study Parameter Domain Knowledge constraints for both discrete and continuous variables, with a special emphasis on parameter sharing.

However, we also investigate constraints between sets of parameters (sum sharing, ratio sharing). In one of these three chapters, we compute closed form Maximum Likelihood estimators in the case when the domain knowledge comes as inequality constraints. These derivations will be performed by directly solving the system of equations that characterize the maximum point instead of resorting to the iterative method.

To validate our approach, we perform experiments on both synthetic and real world data. We compare our models with standard baseline models using the *KL divergence* in the case of synthetic data and the *Average Log Score* (which converges to the negative of cross-entropy on the long run) in the case of real world data.

1.3 Contributions

In this thesis we isolated the problem of incorporating Parameter Domain Knowledge in learning procedures for Bayesian Networks and developed mathematically sound methods to help solve this problem. We feel that we barely scratched the surface of this new area and that further research is needed to improve the methods described here. The main contributions of this research are the following:

- We developed a unified framework for incorporating general Parameter Domain Knowledge constraints in parameter learning procedures for Bayesian Networks by formulating this as a constrained optimization problem. We developed sound methods to solve this problem from both a frequentist and a Bayesian point of view and from both complete and incomplete data. Main contributions here include: computing Maximum Likelihood and Maximum Aposteriori estimators via a Newton-Raphson iterative algorithm, computing the normalization constant for *Constrained Parameter Priors* and presenting several algorithms to deal with incomplete data. All these methods work with arbitrary Parameter Domain Knowledge constraints that are twice differentiable and with continuous second derivatives.

- We derived closed form solutions for our estimators in several cases. Parameter Domain Knowledge types for which this is possible include different variants of parameter sharing, as well as sharing properties of groups of parameters: sum sharing or ratio sharing. We created a Parameter Sharing framework that can describe a broad class of models: Module Networks, Dynamic Bayes Nets, Context Specific Independence models (Bayesian Multinetworks and Bayesian Recursive Multinetworks). While in these models parameter sharing happens at either conditional probability table or conditional distribution (within one table) level, our framework is much more flexible, allowing sharing at parameter level, across conditional distributions of potentially different lengths and across different conditional probability tables.

We would like to point out the unexpected result that closed form estimators were also found even in the case of several inequality Parameter Domain Knowledge constraint types.

- We developed methods to automatically learn Parameter Domain Knowledge constraints based on two scores. The first score is similar to the marginal likelihood. This measure is feasible in practice only if a domain expert can specify restrictions on the set of possible domain knowledge assumptions. The second score for a set of Parameter Domain Knowledge constraints is computed as the cross-validation log-likelihood of the observed data based on Maximum Likelihood Estimators.

- As an application of our methods, we developed a generative model for the activity in the brain during a given cognitive task, as it is observed by an fMRI scanner. We employ the second score described above to find clusters of voxels which can be learnt together using Hidden Process Models with shared parameters. Our models taking advantage of parameter sharing far outperform the baseline model.

- Several formal guarantees are presented about our theoretical results. We show that with infinite amount of training data, our Maximum Likelihood Estimators converge to the *best distribution* (closest in terms of KL distance with the true distribution) that factorizes according to the given Bayesian Network structure and obeys the expert's parameter sharing assumptions, even in the case when incorrect knowledge is supplied. In the case when correct parameter sharing assumptions are provided, we prove that our models will yield lower variance estimates than standard learning methods that ignore this kind of domain knowledge.

1.4 Thesis Statement

Standard methods for performing parameter estimation in Bayesian Networks can be naturally extended to take advantage of domain knowledge that can be provided by a domain expert. These new methods can help lower the variance in parameter estimates by reducing the number of degrees of freedom in the space of allowed parameters. While with an infinite amount of training data one would expect standard parameter estimation methods to perform very well, we show that the impact of incorporating Domain Knowledge constraints is quite noticeable when training data is scare.

1.5 Thesis Outline

Chapter 2 describes work related to this research. There we investigate several types of domain knowledge and models that make certain domain knowledge assumptions. We discuss Dirichlet Priors (and their variants), Markov and Hidden Markov Models, Dynamic Bayesian Networks, Context

Specific Independence (and models that use it) and Probabilistic Rules. Also, that chapter provides a brief tutorial on parameter estimation in standard Bayesian Networks, with both discrete and continuous variables.

In Chapter 3 we formulate parameter learning in the presence of Parameter Domain Knowledge as a constraint optimization problem and show how to solve this problem using an iterative Newton-Raphson method. We study both a frequentist and a Bayesian approach, from both complete and incomplete data.

Chapters 4,5 and 6 present the main theoretical contributions of this research. In chapters 4 and 5 we derive methods that perform parameter estimation in Bayesian Networks involving discrete random variables. Chapter 4 shows how domain knowledge in the form of equality constraints can be incorporated in learning procedures for Bayesian Networks. Here we show how to compute close form estimators for several types of domain knowledge: known parameters, parameter sharing, parameter sum sharing and parameter ratio sharing. Both a frequentist and a Bayesian perspective are investigated, using both complete and incomplete data. Chapter 5 deals with domain knowledge given by inequality constraints involving groups of parameters. In particular, we show how to estimate parameters when the aggregate probability mass of a certain group of parameters is bounded from above by a constant or by the aggregate probability mass of another group of parameters (e.g. frequency of adjectives is less than frequency of nouns in a language). In chapter 6 we look at continuous random variables and equality constraints involving the parameters of these variables. We derive maximum likelihood estimators in the case when certain parameters are shared and in the case when certain parameters are proportional to given constants. There we also show an iterative algorithm to perform maximum likelihood estimation for Shared Hidden Process Models.

Chapter 7 provides some formal guarantees about our estimators. Here we present a theorem proving that our parameter estimators based on domain knowledge provided by an expert have a lower total variance than the ones computed with standard methods. This result assumes the domain knowledge is correct. We also derive a theorem stating how well our model can perform when the domain knowledge provided by the expert is not entirely correct.

To validate our models, in chapter 8 we present experimental results on both synthetic and real world data. There we model the fMRI signal using Hidden Process Models that are shared across clusters of neighboring voxels. This is a very high-dimensional problem, for which we only have several examples available. Since Domain Knowledge is not readily available, we automatically learn the clusters from our data using a cross-validation approach.

In chapter 9 we present a brief summary of this research and we conclude by listing several interesting ideas for future work.

Chapter 2

Related Work

In this chapter we present previous work which we will build on in this thesis, along with work on models that use certain types of domain knowledge. First we give a brief tutorial on parameter learning, from both a frequentist and a Bayesian point of view, from both complete and incomplete data and for both discrete and continuous variables. In the second part of this chapter we present several models that use parameter domain knowledge assumptions along with several of their shortcomings. We discuss *Dirichlet Priors* and related parameter priors, *Hidden Markov Models*, *Dynamic Bayesian Networks*, *Kalman Filters*, *Context Specific Independence*, *Bayesian Multinetworks* and *Recursive Multinetworks*, *Module Networks*, *Object Oriented Bayesian Networks*, *Probabilistic Relational Models*, *Bilinear Models* and *Probabilistic Rules*. Let us start by introducing several important theoretical results that we will rely on subsequently.

2.1 Some Useful Results

In parameter learning we maximize a measure which depends on the parameters of a Bayesian Network. This can be thought of as a constrained optimization problem. Therefore, we will first state two theorems that characterize the optimum point of a constraint optimization problem. We will start with *Lagrange Multipliers* theorem [Arf85, BNO03, Zwi03], which deals with equality constraints:

Theorem 2.1.1. (Lagrange Multipliers) *If the* ***regular*** *point x^* is a local maximizer of the function $f(x)$ of n variables with respect to constraints $g_i(x) = 0$ for $1 \leq i \leq m$, then there exists $\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)$ such that (x^*, λ^*) is the solution of the following system of $n + m$ equations with $n + m$ variables:*

$$\begin{cases} \nabla_x f(x^*) - \sum_i \lambda_i^* \cdot \nabla_x g_i(x^*) = 0 \\ g_i(x^*) = 0 \end{cases}$$

A similar characterization of the optimum points exists when certain constraints come in the form of inequalities. In this case, the optimum satisfies *Karush-Kuhn-Tucker* conditions [Kar39, KT51], which represent an extension of *Lagrange Multipliers* theorem. The main idea here is that any of the inequality constraints can be either *tight* (satisfied) at the optimum or *not tight*. The *Karush-Kuhn-Tucker* theorem basically looks for a set of constraints that are tight, describes the optimum point for those constraints in a fashion similar to *Lagrange Multipliers* theorem, then checks if this point satisfies the rest of the inequality constraints.

Theorem 2.1.2. (Karush-Kuhn-Tucker) *If the **regular** point x^* is a local maximizer of the function f of n variables with respect to constraints $g_i(x) = 0$ for $1 \leq i \leq m$ and $h_j(x) \leq 0$ for $1 \leq j \leq k$, then there exists $\lambda^* = (\lambda_1^*, \ldots, \lambda_m^*)$ and $\mu^* = (\mu_1^*, \ldots, \mu_k^*) \geq 0$ such that (x^*, λ^*, μ^*) is the solution of the following system of $n + m + k$ equations with $n + m + k$ variables:*

$$\begin{cases} \nabla_x f(x^*) - \sum_i \lambda_i^* \cdot \nabla_x g_i(x^*) - \sum_j \mu_j^* \cdot \nabla_x h_j(x^*) = 0 \\ g_i(x^*) = 0 \\ \mu_j^* \cdot h_j(x^*) = 0 \end{cases}$$

Both theorem 2.1.1 and theorem 2.1.2 work for *regular* points. A point x is a *regular* point if the gradients of the equality and *tight* inequality constraints at x are linearly independent. If we approximate the constraints with linear functions by using a Taylor expansion around x, the gradients of the constraints at x would provide the coefficients of these lines. Therefore, if the gradients are linearly dependent at x, one can intuitively think of these approximate linear constraints as either redundant or contradictory at x. In real world optimization problems, it very frequently happens that all points are regular. If non-regular points exist, they are only few and they almost never provide a maximum solution for the constrained optimization problem. When deriving closed form solutions for several types of domain knowledge in the subsequent chapters, we noticed that all feasible points are regular points.

It is important to mention that the above theorems describe only optimum points strictly inside the region on which the objective function f and the constraints are defined. Indeed, consider $f(x) = 2x$ on $[0, 1]$: f is maximized for $x^* = 1$, but the derivative of f does not cancel in $[0, 1]$. If the domain of f is a topologically open set (e.g. the interval $(0, 1)$), then this problem does not exist. Otherwise, one must be careful to consider potential maxima on the boundary of the domain of the objective function. When performing parameter estimation in a standard Bayesian Network using Lagrange Multipliers, one can deal with this problem either enforcing the fact that all observed counts are strictly positive (which might not be the case in real world situations) or by using Dirichlet Priors.

Note that the above theorems state conditions that the optimum point must satisfy. While these conditions are necessary, they are not sufficient. Next we state two propositions describing sufficiency conditions for both *Lagrange Multipliers* and *Karush-Kuhn-Tucker* theorems.

Proposition 2.1.1. (Sufficiency Criterion 1) *Let x^* be a partial solution of the system of equations in either theorem 2.1.1 or theorem 2.1.2. Then x^* is a global maximum provided that:*

- *The objective function f is concave.*
- *The equality constraints are linear functions and the inequality constraints are convex functions.*

This first sufficiency criterion [BV04] is a well known result in optimization theory. Its main drawback is that it makes very restrictive assumptions about the equality constraints. In our research we also deal with non-linear equality constraints. Below we present another set of sufficiency conditions, along with a quick proof:

Proposition 2.1.2. (Sufficiency Criterion 2) *Let x^* be a partial solution of the system of equations in either theorem 2.1.1 or theorem 2.1.2. Then x^* is a global maximum provided that:*

- *All partial solutions $\hat{x}$ of the system of equations satisfy $f(\hat{x}) = f(x^*)$.*
- *There exists a topologically closed region B that contains x^* such that $f(x) < f(x^*)\, \forall x \notin B$.*
- *The constraints define a compact set (the set is bounded and contains the limit of any sequence of points from it).*

In particular, if the above system has only one solution, then the last two conditions are sufficient for that unique solution to be a global maximum.

Proof. Let C be the compact set defined by the constraints and let $A = B \cap C$. Since B is a closed set and C is a compact set, it follows that A is a compact set and therefore the continuous function f would reach a global maximum on A in a point $x' \in A$. We have $f(x) < f(x^*) \leq f(x')\ \forall x \notin B)$ and $f(y) \leq f(x')\ \forall y \in A = B \cap C$. This implies x' is a global maximum of the constrained optimization problem. Therefore x' must be the partial solution of the system given by either theorem 2.1.1 or theorem 2.1.2. From the first sufficiency condition it follows that $f(x') = f(x^*)$ and consequently, x^* is a global maximum for the constrained optimization problem. □

We have seen that the optimum of a constrained optimization problem is characterized by a system which has the same number of equations and variables. For arbitrary constraints and objective function, such a system might be difficult to solve in closed form. Fortunately, several numeric techniques are available. Next we are going to present one of them, namely the Newton-Raphson method:

Algorithm 2.1.1. (Newton-Raphson) *Consider the following system of n equations with n variables:*

$$\begin{cases} f_1(x_1, \ldots, x_n) = 0 \\ \ldots \\ f_n(x_1, \ldots, x_n) = 0 \end{cases}$$

If $x^{(0)} = (x_1^{(0)}, \ldots, x_n^{(0)})^T$ is an initial guess, the Newton-Raphson algorithm looks for a root of the above system using the following recurrence until convergence is reached:

$$x^{(k+1)} = x^{(k)} - J(x^{(k)})^{-1} \cdot (f_1(x^{(k)}), \ldots, f_n(x^{(k)}))^T$$

In this method, $J(x)$ denotes the Jacobian of the system of equations evaluated at the point $x = (x_1, \ldots, x_n)$:

$$J(x_1, \ldots, x_n) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1}(x_1, \ldots, x_n) & \ldots & \frac{\partial f_1}{\partial x_n}(x_1, \ldots, x_n) \\ \ldots & \ldots & \ldots \\ \frac{\partial f_n}{\partial x_1}(x_1, \ldots, x_n) & \ldots & \frac{\partial f_n}{\partial x_n}(x_1, \ldots, x_n) \end{pmatrix}$$

For the interested reader, we recommend [BNO03, BV04] and [PTV93] to learn more details about the Newton-Raphson method as well as alternative optimization methods. As the authors point out in [PTV93], all these methods have limitations in the case when the constraints are arbitrary, non-linear functions.

This concludes our quick review of the optimization methods that we are going to employ in this thesis. In the next section we will see some of these methods at work on the task of developing parameter estimators for standard Bayesian Networks.

2.2 Parameter Estimation in Bayesian Networks

2.2.1 Bayesian Networks - The Basics

Bayesian Networks were introduced in [Pea88] as a means of representing probability distributions over a set of random variables in a compact fashion. A Bayesian Network consists of a structure, which is a Direct Acyclic Graph, and a set of parameters. The parameters are stored in *Conditional Probability Tables (CPTs)*, one for each variable. These tables describe how a variable in the network depends probabilistically on its parents (one can think of these parents as direct causes). In other words, for each possible value that the parents $PA(X)$ of a variable X can take, the table will contain a *Conditional Probability Distribution (CPD)* over the values of X.

In a Bayes Net, random variables will be denoted by capital letters while small letters will be used for the values that these variables can take. Let $X_1, \ldots, X_n$ be the variables in the network and let PA_i be the set of parents of X_i. Note that specifying the sets PA_i is equivalent to specifying the structure of the Bayes Net. Given this notation, the structure of a Bayesian Network encodes the following conditional independence assumption:

Property 2.2.1. *Local Markov Assumption: A variable X_i is conditionally independent of its non-descendants given its parents PA_i.*

Taking advantage of this property, one can derive all conditional independencies between groups of variables in the Bayesian Network by using a criterion called *d-separation* [Pea88] or its equivalent variant, the *Bayes Ball* algorithm [Sha98]. If we think of the example we described in the Introduction, the Bayes Net in figure 1.1 encodes the assumption that the symptoms are conditionally independent of risk factors given the disease of the patient.

Property 2.2.1 also allows us to obtain a factor representation of the joint probability distribution over $X_1, \ldots, X_n$:

$$P(X_1, \ldots, X_n) = \prod_i P(X_i|PA_i)$$

The problems of interest concerning Bayesian Networks are: *Structure Learning*, *Parameter Learning*, *Inference* and *Finding Hidden Variables*. *Structure Learning* is the task of automatically learning the structure of a Bayesian Network given a dataset of observed cases. This is a NP-Hard problem [CGH94]. Common approaches to perform Structure Learning make use of certain heuristics. The *IC Algorithm* [Pea00, PV91] and the *PC Algorithm* [SGS00] are both looking for conditional independencies in the observed data, then build a network structure consistent with these independencies. Other methods perform hill climbing on the structure space by using measures like the *Bayesian Dirichet(BD) Score* [CH92] or the *Bayesian Dirichlet Likelihood-Equivalent (BDE) Score* [HGC95]. The *Structural EM Algorithm* [Fri98] allows structure learning in the case of missing data.

Inference in Bayesian Networks is the task of estimating the posterior probability of a set of query variables in the network given the value of a vector of evidence variables. It can be shown [Coo87] that Inference is a NP-Hard problem. Methods to carry out inference include: *Variable Elimination* [Dec96], *Message Passing on Junction Trees* [Jen96, SS90], *Markov Chain Monte Carlo (MCMC)* sampling methods [GG84, GRS96, Jor99, Mac98, Nea93]. In [Zha98a, ZP96] the authors show methods to exploit conditional independencies in a Bayes Net to perform inference.

Learning Bayesian Networks in the presence of hidden variables was performed in [Fri97] via an EM style algorithm, very similar to Structural EM.

Parameter Learning is the task of estimating the parameters in the Conditional Probability Tables of a Bayesian Network given $D = \{d_1, d_2, \ldots\}$, a set of potentially partially observed examples assumed to be drawn independently at random from our Bayesian Network. We denote by $X_i^{(d)}$ the value of the variable X_i and by $PA_i^{(d)}$ the value of the parents of X_i for example $d \in D$. Once parameters are estimated, one can use the Bayesian Network to perform inference on future examples. In the following subsections we give a brief tutorial on parameter learning in Bayesian Networks, concentrating on the parts that we extended in this research. In particular, we assume the structure of the Bayes Net is provided. We focus on parameter learning for models that involve discrete variables, but we discuss the same task on particular types of models made of Gaussian random variables. Before we continue, we would like to suggest several additional references on Bayesian Networks for the interested reader: [CDL99, DHS01, Edw00, Hec99, Jen96, Jen01, Lau96, Mit97, Pea00, RN95, Whi90].

2.2.2 Frequentist Approach from Complete Data for Discrete Variables

A frequentist tries to estimate a ”best set” of parameters θ. In general, this translates into finding the set of parameters that maximize the *Data Likelihood*: $L(\theta) = P(D|\theta)$. Equivalently, one can maximize the *Data Log-Likelihood*: $l(\theta) = \log P(D|\theta)$. This measure has the following nice property:

Property 2.2.2. Decomposability of Log-Likelihood. *The Log-likelihood function can be written as the sum of n components, one for each variable in the Bayesian Network. The component corresponding to variable X can be decomposed further into a sum of subcomponents, one for each instantiation of the parents of X. If there are no constraints between parameters describing different subcomponents, then this decomposability will allow us to efficiently perform parameter estimation in Bayesian Networks by solving a collection or smaller optimization problems, one for each subcomponent.*

Below we show how to derive the Maximum Likelihood Estimators for the parameters in a Bayes Net since we are going to extend this framework in the subsequent chapters of this thesis. Before getting to the main result, we need to introduce additional notations that will be used subsequently.

For a discrete variable X, let $\{x_1, x_2, \ldots\}$ be its values. If the variable is indexed, then the index will be also kept. For example, the variable X_i will have values $\{x_{i1}, x_{i2}, \ldots\}$. To represent the parameters in the network we follow the notation in [Mit97]. Let $\theta_{ijk} = P(X_i = x_{ij}|PA_i = pa_{ik})$. Denote by θ_{i**} the set of parameters that appear in the Conditional Probability Table $P(X_i|PA_i)$. In this table, rows are associated with values of X_i and columns with values of PA_i. Let θ_{i*k} be the set of parameters in column k ($PA_i = pa_{ik}$) in this table.

Let us define the following indicator function over the training dataset D:

$$\delta_{ijk}(d) = \begin{cases} 1 & \text{if } X_i = x_{ij} \text{ and } PA_i = pa_{ik} \\ 0 & \text{otherwise} \end{cases}$$

In our training set, let $N_{ijk} = \sum_l \delta_{ijk}(l)$ be the number of examples which have $X_i = x_{ij}$ and $PA_i = pa_{ik}$. Further, let $N_{ik} = \sum_j N_{ijk}$ be the number of cases in the training set which have $PA_i = pa_{ik}$. One can think of N_{ijk} as the observed count corresponding to θ_{ijk} and of N_{ik} as the cumulative observed count associated with the parameters in column k of the Conditional Probability Table for X_i.

With these notations, the data likelihood can be written as: $P(D|\theta) = \prod_{i,j,k} \theta_{ijk}^{N_{ijk}}$. It is easy to see that, at the point $\hat{\theta}$ that maximizes $P(D|\theta)$ we can not have $\hat{\theta}_{ijk} = 0$ unless $N_{ijk} = 0$. If all the observed counts are positive, the maximum likelihood estimators are strictly positive and can be found by maximizing $\log P(D|\theta) = \sum_{i,j,k} N_{ijk} \cdot \log \theta_{ijk}$. We have the following theorem:

Theorem 2.2.1. *If all N_{ijk} are strictly positive, the Maximum Likelihood Estimators of the parameters in a Bayesian Network are given by:*

$$\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ik}}$$

Proof. Because of property 2.2.2, the problem of maximizing the data log-likelihood can be broken down into a set of independent optimization subproblems:

$$P_{ik}: \qquad argmax \;\; h_{ik}(\theta_{i*k}) = \sum_j N_{ijk} \cdot \log \theta_{ijk}$$
$$\left\{ \; g_{ik}(\theta_{i*k}) = (\textstyle\sum_j \theta_{ijk}) - 1 = 0 \right.$$

The domain of the functions h_{ik} and g_{ik} is given by $\theta_{i*k} \in \prod_j (0,1)$, which is a topologically open set. Therefore, if a maximum exists, it must lie inside the region determined by the domains of h_{ik} and g_{ik} and thus we try to find the solution of P_{ik} using Lagrange Multipliers theorem. Introduce Lagrange Multiplier λ_{ik} for the above constraint in P_{ik}. Let $LM(\theta_{i*k}, \lambda_{ik}) = h_{ik}(\theta_{i*k}) - \lambda_{ik} \cdot g_i k(\theta_{i*k})$. Then the point which maximizes P_{ik} is among the solutions of the system $\nabla LM(\theta_{i*k}, \lambda_{ik}) = 0$. Let $(\hat{\theta}_{i*k}, \hat{\lambda}_{ik})$ be a solution of this system. We have: $0 = \frac{\partial LM}{\partial \theta_{ijk}} = \frac{N_{ijk}}{\theta_{ijk}} - \hat{\lambda}_{ik}$ for all j. Therefore: $\hat{\theta}_{ijk} = \frac{N_{ijk}}{\hat{\lambda}_{ik}}$. Summing up for all values of j, we obtain:

$$0 = \frac{\partial LM}{\partial \lambda} = (\sum_j \hat{\theta}_{ijk}) - 1 = (\sum_j \frac{N_{ijk}}{\hat{\lambda}_{ik}}) - 1 = \frac{N_{ik}}{\hat{\lambda}_{ik}} - 1$$

From the last equation we compute the value of $\hat{\lambda}_{ik} = N_{ik}$. This gives us: $\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ik}}$. Because all the counts are positive, the point $\hat{\theta}$ is well defined.

Until now we only showed that, if a maximum point exists, it must be the $\hat{\theta}$ found above. Using the sufficiency criterion 2.1.1, it follows that $\hat{\theta}$ must be a global maximum. □

In the case when some, but not all, of the observed counts in subproblem P_{ik} are zero, the corresponding parameters don't even appear in the likelihood function. Therefore P_{ik} has an inequality constraint but eventually the Maximum Likelihood Estimators are given by the same formulas. If all of the observed counts in subproblem P_{ik} are zero, any probability distribution over the values of X_i will be a solution for P_{ik}.

2.2.3 Frequentist Approach from Incomplete Data for Discrete Variables

When training data is not fully observable, one can still perform approximate maximum likelihood estimation via the *Expectation Maximization (EM)* algorithm [DLR77, MK96]. Assume we have some incomplete data X which is described by a set of parameters θ. The EM algorithm is an iterative procedure that improves the data log-likelihood $\log\left(P(X|\theta)\right.$ at each step. If Z is the set of missing data, the parameters θ_{t+1} (at iteration $t+1$) are estimated from parameters θ_t (at iteration t) using the following formula:

$$\theta_{t+1} = \text{argmax}_\theta \; E_{P(Z|X,\theta_t)}\left[\log P(X,Z|\theta)\right] \tag{2.1}$$

This algorithm works in two steps. In the ***E-Step*** it computes $P(Z|X,\theta_t)$, the posterior probability of missing data given observed data and current parameter estimates. In the ***M-Step*** it re-estimates the parameters by maximizing $\log P(X,Z|\theta)$, the expected log-likelihood of complete data, assuming the missing data comes from the distribution computed in the *E-Step*. The EM is guaranteed to converge to a zero of the log-likelihood function's gradient. In [NH98] the authors present an alternative formulation of the EM algorithm where each step is maximizing over a subset of variables of a certain energy function. Based on that formulation, they derive a modified version of the EM algorithm which allows for partial E-Steps.

In the case of Bayesian Networks, when the data is incomplete, we cannot compute the counts N_{ijk} and N_{ik}. However, they can be treated as random variables. The equation 2.1 becomes:

$$\theta_{t+1} = \text{argmax}_\theta \sum_{i,j,k} E_{\theta_t}\left[N_{ijk}\right] \cdot \log \theta_{ijk} \tag{2.2}$$

The above equation yields the following iterative EM algorithm for computing the Maximum Likelihood Estimators for the parameters in a Bayes Net in the case when data is incomplete:

The EM Algorithm. *Repeat the following two steps until convergence:*

E-Step*: Use any sound inference algorithm to compute the expected counts $E[N_{ijk}]$ and $E[N_{ik}]$ under the current parameter estimates $\hat{\theta}$. If just starting, assign $\hat{\theta}$ randomly or according to some* *domain knowledge.*

M-Step*: Re-estimate the parameters by maximizing the data likelihood using Theorem 2.2.1, assuming that the observed counts are equal to the expected counts given by the E-Step:*

$$\hat{\theta}_{ijk} = \frac{E[N_{ijk}]}{E[N_{ik}]}$$

2.2.4 Bayesian Approach from Complete Data for Discrete Variables

From a Bayesian point of view, each choice of parameters is possible, but some choices have higher prior probability of occurring. Therefore, to do model averaging, we need to specify priors over the space of parameters. It can be proved [GH97], that under certain conditions, choosing Dirichlet Priors over the parameters of a Bayesian Network is inevitable. Below we define Dirichlet Priors and show how to perform Maximum Aposteriori and Bayesian estimation.

2.2.4.1 Dirichlet Distribution

Assume we have a distribution $P(X)$ over the finite set of values $\{x_1, x_2, \ldots, x_n\}$. Let $\theta_i = P(X = x_i)$ be the parameters of this discrete distribution. Before seeing any data from this distribution we may or may not know what the parameters are. In the case when we do not know them, for all purposes, the parameters of distribution P can be seen as a random vector $\theta = (\theta_1, \theta_2, \ldots, \theta_n)$. A Dirichlet Distribution is a way of representing the uncertainty one has about θ before seeing any data sampled from P. It has the following formula:

$$P(\theta) = \begin{cases} \frac{1}{Z}\prod_{i=1}^{n} \theta_i^{\alpha_i - 1} & \text{if } \sum_i \theta_i = 1 \text{ and } \theta_i \geq 0 \; \forall i \\ 0 & \text{otherwise} \end{cases}$$

Here are some properties of the Dirichlet Distribution:

- The normalization constant Z can be computed by enforcing the fact that this is a valid probability distribution. In other words, we must have $\int_{-\infty}^{\infty} P \, d\theta = 1$. This yields:

$$Z = \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\alpha)} \text{ where } \alpha = \sum_{i=1}^{n} \alpha_i$$

  Here $\Gamma(\cdot)$ represents the *Gamma Function*, also known as the *Extended Factorial*.

- $$E[\theta_i] = \frac{\alpha_i}{\alpha} \text{ and } Var[\theta_i] = \frac{E[\theta_i] \cdot (1 - E[\theta_i])}{\alpha + 1}$$

Intuitively one can think of a Dirichlet Distribution as an expert's guess of the parameters θ. Basically, the expert is telling us that he/she believes $\theta_i = \frac{\alpha_i}{\alpha}$, but is not completely sure, therefore allows these parameters to vary around the guess with the variance given above. Parameters α_i can be thought of as how many times the expert believes he/she will observe $X = x_i$ in a sample of α examples drawn independently at random from distribution P. The bigger α is, the lower the variance in each parameter and consequently, the larger the number of observed cases needed to overturn expert's belief.

- Assume $P(\theta)$ is a Dirichlet Distribution. Given D, a dataset of samples that are drawn independently at random from $P(X)$, it is easy to see that $P(\theta|D) \propto P(D|\theta) \cdot P(\theta)$ is also a Dirichlet Distribution. Because of this fact and because $P(D|\theta)$ is a Multinomial Distribution, Dirichlet Distribution is also referred to as *the conjugate* of Multinomial Distribution.

2.2.4.2 Dirichlet Priors in Bayesian Networks

We saw above how one can define a prior over the parameters of a discrete probability distribution. One can think of the parameters of a Bayesian Network as a set of conditional probability distributions. Each table contains a subset of conditional probability distributions, one for each column θ_{i*k}. One can specify a probability distribution over the space of parameters in a Bayesian Network by first specifying a Dirichlet Distribution for each conditional probability distribution in the Graphical Model, then describe the way these Dirichlet Distributions are dependent on each other. For the first part, let the Dirichlet Distribution associated to parameters θ_{i*k} be:

$$P(\theta_{i*k}) = \frac{1}{Z_{ik}} \prod_j \theta_{ijk}^{\alpha_{ijk}-1}$$

To relate these Dirichlet Distributions to each other, in [SL90] the following assumptions are made:

- *Global Independence* assumption: parameters corresponding to different variables in the Graphical Model are independent of each other. In other words, $\theta_{i**} \perp \theta_{j**}$ for all $i \neq j$.
- *Local Independence* assumption: parameters associated with different configurations of the parents of the same random variable are independent of each other. This is equivalent to $\theta_{i*k} \perp \theta_{i*l}$ for all $k \neq l$.

Based on the considerations above, we can define a Dirichlet Prior over the space of parameters in the Bayes Net:

$$P(\theta) = \prod_{i,k} P(\theta_{i*k}) = \frac{1}{Z} \prod_{i,j,k} \theta_{ijk}^{\alpha_{ijk}-1}$$

where the normalization constant is given by: $Z = \prod_{i,k} Z_{ik}$.

Property 2.2.3. *Assume $P(\theta)$ is a Dirichlet Prior. Given D a dataset of fully observable cases drawn independently at random from the probability distribution represented by a Bayes Net, one can easily see that $P(\theta|D) \propto P(D|\theta) \cdot P(\theta)$ is also a Dirichlet Prior.*

2.2.4.3 Maximum Aposteriori Estimators

In Maximum Aposteriori estimation we are looking for the parameters with the maximum posterior probability given a dataset of observed cases:

$$P(\theta|D) \propto P(D|\theta) \cdot P(\theta) \propto \prod_{i,j,k} \theta_{ijk}^{N_{ijk}+\alpha_{ijk}-1}$$

Note that, for the purpose of computing the maximum a posteriori estimates for parameters in our Bayes Network, $P(D)$ and the normalization constant Z do not matter. A theorem similar to the one describing Maximum Likelihood Estimators can be derived in this case:

Theorem 2.2.2. *The Maximum Aposteriori Estimators in a standard Bayes Net are given by:*

$$\hat{\theta}_{ijk} = \frac{N_{ijk} + \alpha_{ijk} - 1}{N_{ik} + \sum_{j'}(\alpha_{ij'k} - 1)}$$

Proof. Given the formula for $P(\theta|D)$ above, one can see that the Maximum Aposteriori Estimators are equal to the Maximum Likelihood Estimators of the same Bayes Net structure where the observed counts N_{ijk} are incremented with $\alpha_{ijk} - 1$. The result now follows from theorem 2.2.1. □

2.2.4.4 Bayesian Averaging

From a Bayesian point of view, we are interested in predicting the next data point given a set of previously observed data points $D = \{D_1, \ldots, D_n\}$. This can be written as follows:

$$\begin{aligned} P(d_{n+1}|D) &= \int P(d_{n+1}|\theta) \cdot P(\theta|D)\, d\theta \\ &= \int (\prod_{i,j,k} \theta_{ijk}^{\delta_{ijk}(d_{n+1})}) \cdot P(\theta|D)\, d\theta \\ &= \prod_{i,j,k} (E_{P(\theta|D)}[\theta_{ijk}])^{\delta_{ijk}(d_{n+1})}\, d\theta \end{aligned}$$

One can easily see that the above prediction formula using Bayesian averaging will yield the same results as predicting the next example using the Bayes Net model with parameters $\hat{\theta}_{ijk} = E_{P(\theta|D)}[\theta_{ijk}]$. We have the following theorem:

Theorem 2.2.3. *The Bayesian Averaging Estimators in a standard Bayes Net are given by:*

$$\hat{\theta}_{ijk} = E_{P(\theta|D)}[\theta_{ijk}] = \frac{N_{ijk} + \alpha_{ijk}}{N_{ik} + \alpha_{ik}} \ \textit{where} \ \alpha_{ik} = \sum_{j'} \alpha_{ij'k}$$

Proof. As stated in property 2.2.3, $P(\theta|D) \propto P(D|\theta) \cdot P(\theta)$ is also a Dirichlet Prior. Our result now follows from the basic properties stated above for Dirichlet Distributions. □

2.2.5 Bayesian Approach from Incomplete Data for Discrete Variables

When data is incomplete, the MAP Estimators can be computed by slightly modifying the EM Algorithm for computing the Maximum Likelihood Estimators. In order to do Bayesian Averaging we write:

$$P(d_{n+1}|d_1, \ldots, d_n) = \frac{\int P(d_{n+1}, \ldots, d_1|\theta) \cdot P(\theta) d\theta}{\int P(d_n, \ldots, d_1|\theta) \cdot P(\theta) d\theta} \tag{2.3}$$

Let us take a look at the denominator. Denote by U the set of missing values such that $D \cup U$ is a complete dataset. Then, $\int P(D|\theta) \cdot P(\theta) d\theta = \sum_U \int P(D, U|\theta) \cdot P(\theta) d\theta$. It is easy to see that $\int P(D, U|\theta) \cdot P(\theta) d\theta$ is the normalization constant for a Dirichlet Prior. Therefore, in the case of incomplete data, $\int P(D|\theta) \cdot P(\theta) d\theta$ is a sum of normalization constants for certain Dirichlet Priors (one for each completion of D). Since we know how to compute normalization constants for Dirichlet Priors, we therefore know how to compute the denominator of the the equation 2.3. In a similar way we can compute the nominator and thus we just showed how to perform Bayesian Averaging in the case when we are dealing with incomplete data.

The above procedure for performing Bayesian Estimation is computationally expensive because the number of terms in the summation grows extremely quickly. If only one binary value is missing in each of the n training examples, there will be 2^n terms in the summation. Instead, techniques that approximate $P(D|\theta)$ with a Dirichlet Prior [CDS96, TI76] are used when data is incomplete.

2.2.6 Learning with Continuous Variables

If a variable X_i is continuous, it can take any value in an open set of the canonical topological space defined over the real numbers. In this case we have to specify several parameters describing the continuous conditional probability distribution of X_i given each instantiation of the parents. A continuous random variable X_i is characterized by several parameters that describe the conditional probability distribution for each instantiation of the parents. Learning procedures differ depending on the type of continuous random variable. Because of this fact, here we are only going to present learning in the setting when the conditional probability distributions corresponding to X_i are Gaussians whose means are linear combinations of the values of the parents. In other words, we have $X_i|PA_i \sim N(PA_i \cdot \theta_i, \sigma_i^2)$ where θ_i is a column vector of parameters of length equal to the number of parents of variable X_i.

Let us denote by X_i^l the value of X_i and by PA_i^l the value of PA_i in training example d_l. Define A_i to be the *Parents Matrix* and b_i to be the *Variable Vector* for variable X_i such that each has a row corresponding to each example in the training set:

$$A_i = \begin{pmatrix} PA_i^1 \\ PA_i^2 \\ \cdots \\ PA_i^m \end{pmatrix} \text{ and } b_i = \begin{pmatrix} X_i^1 \\ X_i^2 \\ \cdots \\ X_i^m \end{pmatrix}$$

With these notations, we have the following theorem:

Theorem 2.2.4. *If $A_i^T \cdot A_i$ is non-singular, the Maximum Likelihood Estimators of the parameters for gaussian variable X_i are given by:*

$$\hat{\theta}_i = (A_i^T \cdot A_i)^{-1} \cdot A_i^T \cdot b_i$$
$$\hat{\sigma}_i^2 = \frac{||A_i \cdot \hat{\theta}_i - b_i||^2}{m}$$

Proof. We remind the reader that, because of property 2.2.2, the problem of maximizing the data log-likelihood can be broken down into a set of independent optimization problems, one for each conditional probability distribution. The component of the log-likelihood corresponding to variable X_i can be written as:

$$\begin{aligned} l_i(\theta_i, \sigma_i) &= -\frac{m}{2} \cdot \log(2\pi) - m \cdot \log(\sigma_i) - \frac{1}{2 \cdot \sigma_i^2} \cdot \sum_l (X_i^l - PA_i^l \cdot \theta_i)^2 \\ &= -\frac{m}{2} \cdot \log(2\pi) - m \cdot \log(\sigma_i) - \frac{1}{2 \cdot \sigma_i^2} \cdot ||A_i \cdot \theta_i - b_i||^2 \end{aligned}$$

It is easy to see that the value of θ_i that maximizes l_i for a given σ_i is the same for all values of σ_i. Therefore we can first maximize l_i with respect to θ_i, then maximize with respect to σ_i. Maximizing with respect to θ_i is equivalent to solving for the least squares solution of the system $A_i \cdot x = b_i$. The solution to this problem is given by $(A_i^T \cdot A_i)^{-1} \cdot A_i^T \cdot b_i$. Therefore we have:

$$\hat{\theta}_i = \text{argmax}_\theta \; ||A_i \cdot \hat{\theta}_i - b_i||^2 = (A_i^T \cdot A_i)^{-1} \cdot A_i^T \cdot b_i$$

Now that we computed the Maximum Likelihood estimator for θ_i, we can maximize with respect to σ_i by simply solving $\frac{\partial l_i}{\partial \sigma_i}(\hat{\theta}_i, \hat{\sigma}_i) = 0$. The solution of this equation is $\hat{\sigma}_i^2 = \frac{||A_i \cdot \hat{\theta}_i - b_i||^2}{m}$ and it is easy to see that this is a maximum. □

If $A_i^T \cdot A_i$ is singular, θ_i can still be estimated by minimizing $||A_i \cdot x - b_i||^2$ using an SVD approach. Once $\hat{\theta}_i$ is computed, the estimator $\hat{\sigma}_i^2$ can be found using the same formula as in the above theorem.

A Bayesian Network where all variables in the network are gaussian is called a *Gaussian Network*. In this case, the joint probability distribution is also a Gaussian. Learning in Gaussian Networks was studied in [GH94]. A different approach to deal with continuous variables is presented in [FG96a, FGL98] where the authors show how discretization of continuous random variables can help on classification tasks.

In order to perform Maximum Aposteriori estimation and Bayesian Averaging in a Gaussian Network, one has to define priors on the parameters θ_i and σ_i. Common choices are *ARD Priors* [LCT02] and *Normal-Wishart Priors* [Min00b]. When data is incomplete, parameter estimation becomes very difficult. In [GH94] it is suggested that missing values can be filled with their expectation under the current parameter estimates, an approach very similar to the EM algorithm. We are not going to provide more details here because we did not extend the results in these papers to take advantage of parameter related domain knowledge.

2.2.7 Estimating Performance in a Bayesian Network

In the previous subsections we showed how to perform parameter estimation in Bayesian Networks. However, we provided no way of assessing the quality of the learnt parameters. The purpose of this subsection is to describe ways to estimate the performance of a Bayesian Network.

A Bayesian Network is a compact way to represent a joint probability distribution Q over a set of random variables. This distribution may or may not reflect accurately the true probability distribution P that we are trying to estimate. A standard way to measure the distance between two distributions over the same values is to compute their *KL Divergence*:

$$KL(P, Q) = \sum_x P(x) \cdot \log \frac{P(x)}{Q(x)} = H(P, Q) - H(P)$$

Here $H(P)$ stands for the *Entropy* of a probability distribution P and $H(P, Q)$ stands for the *Cross-Entropy* between two probability distributions P and Q over the same values. We suggest [CT91] for a comprehensive review of Information Theory concepts. It is easy to see that $KL(P, P) = 0$ and that this measure is not symmetric. Using KL divergence to compute the performance of our learnt distribution assumes we have access to the true distribution of the data. This is possible only in a controlled environment. In the real world however, we rarely have access to the underlying distribution of the data. In this case it is common to use the *Average Log-Score (ALS)* on a test set D:

$$ALS = \frac{\sum_{i=1}^{m} \log\left(P_1(d_i)\right)}{m}$$

As a consequence of the Law of Large Numbers, it is easy to see that the Average Log-Score is a measure that approximates the negative of cross-entropy ($-H(P,Q)$) when the number of test examples goes towards infinity. This score is negative and, the better the model, the higher the score is. If the model is perfect, then the ALS score will be close to $-H(P)$ given enough testing examples.

Now let us see how the improvement in the Average Log-Score translates in terms of improvement in data likelihood. Assume we have two probabilistic models, one described by P_1 and one by P_2. Also, assume $D = \{d_1, \ldots, d_m\}$ is a test set with m examples drawn independently at random from the true distribution we are trying to estimate. The difference δ in ALS can be written as:

$$\delta = \frac{\sum_{i=1}^{m} \log\left(P_1(d_i)\right)}{m} - \frac{\sum_{i=1}^{m} \log\left(P_2(d_i)\right)}{m} = \frac{\sum_{i=1}^{m} \log \frac{P_1(d_i)}{P_2(d_i)}}{m}$$

This can be rephrased equivalently as: on the average, $\log \frac{P_1(d)}{P_2(d)}$ is equal to δ. This means on the average $\frac{P_1(d)}{P_2(d)}$ is equal to e^{δ}. To summarize, on the average (over the test set), P_1 outperforms P_2 by a factor of e^{δ} in terms of data likelihood.

This concludes our mini-tutorial on learning parameters in Bayesian Networks. We remind the reader that we focused only on the parts that we are going to extend in this research. A more comprehensive tutorial on learning Graphical Models can be found in [Mur02].

2.3 Parameter Related Domain Knowledge: Previous Research

The standard way of representing Parameter Domain Knowledge in Bayesian Models is by using Dirichlet Priors, which were presented in the previous section. In [GH97], it is shown that Dirichlet Priors are the only possible priors provided certain assumptions hold. As mentioned before, one can think of a Dirichlet Prior as an expert's guess for the parameters in a Bayes Net, allowing room for some variance around the guess. For example, when we want to create a language model for a specific topic, we can take a global language model and "guess" that the word probabilities in the topic specific model are the same are as the ones in the global model before seeing any documents on that specific topic. The total size of the documents in the global model will determine how confident we are in this guess.

A Dirichlet Prior assumes a guess on all parameters in the model. However, in many real world problems, a domain expert might not be able to specify a useful guess for all the parameters in the model. In this case, one common use of Dirichlet Priors is to make sure all the counts corresponding to parameters in the model become positive to overcome problems that may appear in Maximum

Likelihood estimation with zero observed counts. Moreover, for the task of structure learning, assigning Dirichlet Priors for all possible structures can be prohibitive. Uninformative Dirichlet Priors are used in [CH92] in order to derive the $K2$ score, a version of the BD score used in learning Bayes Net structures. An alternative is provided in [HGC95] where the authors show that, if several conditions are satisfied, one can build informative Dirichlet Priors starting from a prior Bayes Net. The problem of estimating the parameters of a Dirichlet Prior from a set of observed probabilities coming from that prior is also difficult: there is no known closed form solution for the maximum likelihood estimators, but [Min00a] presents an iterative method to estimate these parameters.

Even though an expert can provide very useful information, this information might not be representable as a Dirichlet Prior. Because a Dirichlet Prior consists of a collection of independent Dirichlet Distributions, one for each column in each Conditional Probability Table, it can represent neither constraints among parameters in different columns of a table nor constraints among parameters in different tables. Moreover, Dirichlet Priors can not even represent simple equality constraints between two parameters in the same conditional distribution. They can represent equality constraints like $\theta_{111} = \theta_{121} = 0.1$, but not the less restrictive $\theta_{111} = \theta_{121}$. In order to help enforce such constraints, one can place priors on the parameters of the Dirichlet Prior. To illustrate, we can consider that $\alpha_{111} = \alpha_{121} = \mu$ where $\mu \sim U(0, 1)$. The problem with this approach is that the marginal likelihood can not be computed in closed form anymore, even though approximate sampling and iterative methods can be conceived, depending on the type of constraints one wants to represent.

Dirichlet Tree Priors [Den91, Min99] are extensions of standard Dirichlet Priors. These priors work on a different parametrization of the Bayesian Network. Each parameter in a conditional probability distribution in a standard Bayes Net is seen as a leaf of a tree and its value is equal to the product of the probabilities assigned to the edges on the path from the root to the corresponding leaf. For each node in the tree, the Dirichlet Tree Prior basically assigns a Dirichlet Prior on the probabilities corresponding to edges coming out of that node. Even though Dirichlet Tree Priors allow a little more correlation (between the parameters within one conditional probability distribution) than standard Dirichlet Priors, they inherit the same problems with representing parameter constraints. In addition, a Dirichlet Tree Prior may require up to twice as many parameters to represent.

Dirichlet Priors can be considered to be part of a broader category of methods that employ parameter domain knowledge, called smoothing methods. A comparison of the common smoothing methods for language models can be found in [ZL01]. In all the methods presented there, the word probability model for each document is a combination of the document specific Maximum Likelihood estimators with a global word model.

In subsection 2.2.4 we introduced *Local Independence* and *Global Independence* assumptions that allowed us to define Dirichlet Priors on the parameter space of a Bayes Net. However, these are very restrictive assumptions which often do not hold in the real world. An approach to relax the local independence assumption for binary variables is investigated in [GMC99], but the reported results seem minimal on relatively simple networks. *Dependent Dirichlet Priors* [Hoo04] are a generalization of Dirichlet Priors that allow for certain dependencies among different conditional probability distributions in the same conditional probability table for a given variable X in the network. These priors can be written as functions of independent $Gamma$ random variables. The dependence among conditional probability distributions is due to the fact that, for each value of X, there are three types of such $Gamma$ distributions: one corresponding to the specific instantiation of the parents of X, one corresponding to the specific value of each parent and one corresponding to the variable itself. It can be shown the *Dependent Dirichlet Priors,* when restricted to a specific instantiation of the parents of X, are standard Dirichlet distributions. These *Dependent Dirichlet Priors* can cover the case when all the conditional probability distributions of X are independent and also the case when all are equal. In this case, Bayesian estimators cannot be computed in closed form, but in [Hoo04], the author presents a method to compute approximate estimators, which are linear rational fractions of the observed counts and Dirichlet parameters, by minimizing a certain mean square error measure.

Context Specific Independence (CSI) [BFG96] states conditional independencies that hold only in certain contexts i.e. of the form $P[X|Y, Z, C = c] = P[X|Z, C = c]$ and therefore can specify that some conditional probability distributions (the ones with $C = c$ where $C \subseteq PA(X)$) in the conditional probability table for a variable X are the same (they share all parameters in those distributions). Context Specific Independence can be exploited to efficiently encode and learn conditional probability tables using decision trees and decision graphs as described in [CHM97, FG96b]. The role of Context Specific Independence for the task of probabilistic inference in Bayes Nets is discussed in [Zha98b, ZP99].

The Dependent Dirichlet Priors defined in [Hoo04] can represent a very restrictive set of Context Specific Independence assumptions: for each context given by a subset $C \subseteq PA_i$, X_i and $PA_i \setminus C$ are independent in the context $C = c$ for all values of c. In other words, X_i is independent of $PA_i \setminus C$ given C. Other models that use Context Specific Independence assumptions are: Bayesian Multinets, Bayesian Recursive Multinets and Similarity Networks. *Bayesian Multinets* [GH96] describe probability distributions as mixture distributions, where each mixture component is represented using a Bayes Net. Each mixture determines a subpopulation. More precisely, a Bayesian Multinet consists of an subpopulation indicator variable S and a collection of Bayes Nets, one for each value of S. Using the indicator variable S as the class variable, Bayesian Multinets have been applied in [CG01, FGG97, GH96] for classification purposes. The conditional independence relations among variables may differ from one subpopulation to another i.e. they hold within the context given by the value of S. *Dynamic Bayes Multinets*, a temporal extension of Bayesian Multinets, have been presented in [Bil00]. *Similarity Networks* [Hec90] were the basis for developing Bayes Multinets. They also encode different independence assumptions for different values/set of values of the distinguished (class) variable. *Recursive Bayesian Multinets* [PLL02] are an extension of Bayes Multinets where the contexts are represented using a decision tree's edges and each leaf represents a Bayesian network over the variables not involved in the context.

A widely used form of parameter domain knowledge employed by Graphical Models is *Parameter Sharing*. Context Specific Independence (explained above) is used to represent assumptions that certain conditional probability distributions which are part of the same conditional probability table share all their parameters. Dynamical Models like *Hidden Markov Models (HMMs)* [Rab89], *Input-Output HMMs* [BF96], *Factorial HMMs* [GJ97] and *Coupled Hidden Markov Models* [Bra96] are all part of *Dynamic Bayes Nets* [Mur02], a broader category of models that make the parameter sharing assumption that the conditional probability tables corresponding to a given variable are the same at each point in time. *Kalman Filters* [Kal60, May79, WB95], a particular subclass of *Linear Gaussian Models* [RG99], express the current state of the system as a linear function of the previous state with zero mean gaussian noise while the current measurement/observation is modelled as a linear function of the current state with zero mean gaussian noise. The assumption behind Kalman Filters is these state-to-state and state-to-observation matrices are shared across all time instances.

Object Oriented Bayes Nets [KP97] make the assumption that objects belonging to a certain type class can be modelled using a single fragment Bayes Net. Each object can be influenced from the outside via its input variables and can influence other objects via its output variables. However, internal variables of an object are not accessible outside that object. A bigger Bayes Net may involve multiple instances of the same type class, and therefore their inner conditional probability tables can be learned jointly. The task of learning Object Oriented Bayes Nets was addressed in [BLN01, LB01]. Similar models are presented in [BW00, LM97] and an extension to *Dynamic Object Oriented Bayes Nets* is described in [FKP98]. *Probabilistic Relational Models* [FGK99, Pfe00] are a more general class of models, where objects of a certain type class also share the way they depend on related objects of other type classes. The assumption of independence of examples given the model does not hold anymore for Probabilistic Relational Models: all objects coexist and can influence each other. Learning a Probabilistic Relational Model is performed by first unfolding the model as a Bayes Net and then by estimating the parameters from a single ”huge” training example which contains all objects. This turns out to be a feasible approach because of the parameter sharing assumption stated above.

Module Networks [SPR03, SSR03], can be used in domains where many variables exhibit similar behavior. A Module Network is a Bayes Net where the variables have been partitioned in modules. The variables in a module all share the same set of parents, same set of values and they depend probabilistically on their parents in the same way. This fact allows their corresponding conditional probability tables to be learned together.

A different type of parameter sharing is found in *Bilinear Models* [TF00]. Here, each instance of the observed data vector Y of dimension n is assumed to have a ”style” and a ”content”. An observed example Y_{sc} in style s and content c can be written as $Y_{sc} \sim N(A_s \cdot b_c, \sigma^2 \cdot I_n)$ where all A_s are nXk matrices, all b_c are k dimensional vectors and σ^2 is the variance (in the error) which does not depend on s and c. As one can easily notice, all observations in a certain style s share the matrix A_s while all the observations with content c share the vector b_c. In addition, all observations share the variance in the error.

Another type of parameter related domain knowledge comes in the form of *Probabilistic Rules*. Using this kind of domain knowledge in the medical field was addressed in [RR03, RSN02, RSN03]. There, the authors present an approach for modelling disease state using probabilistic rules to specify the posterior probabilities of certain disease related outcomes given some phrases that appear in doctors’ dictations. Each such rule generates a probabilistic observation about a certain variable in a Bayes Net. The observations corresponding to a variable are then combined in a Naive Bayes fashion to allow to estimate the most likely disease state for a specific patient. Probabilistic rules can be used to assign values to certain parameters, but we are not aware of them being used beyond that purpose for estimating the parameters of a Bayesian Network.

To summarize, the main methods to represent parameter related domain knowledge fall into two categories: Dirichlet Priors and their variants (including smoothing techniques) and Parameter Sharing of several kinds. One of the main problems with Dirichlet Priors and related models is that it is impossible to represent even simple equality constraints between parameters without using priors on the parameters of the Dirichelet Prior, in which case the marginal likelihood can not be computed in closed form anymore and expensive approximate methods are required to perform parameter estimation. A second problem is that it is often beyond the expert’s ability to specify a full Dirichlet Prior on the parameters of a Bayes Net. Parameter Sharing methods can only represent equalities among parameters, but no other, more complicated, constraints. Current models use parameter sharing at either the level of conditional probability table (Module Networks, HMMs) or at the level of conditional probability distribution (Context Specific Independence) within the same table. No such model allows sharing at parameter level of granularity.

The main contribution of this thesis is an unified framework that allows us to incorporate any kind of domain knowledge constraints (that obey certain differentiability assumptions) in parameter learning procedures for Bayesian Networks. We present closed form solutions for several types of Parameter Domain Knowledge which the methods described in this chapter can not represent. We show how widely used models including Hidden Markov Models, Dynamic Bayesian Networks, Module Networks and Context Specific Independence are just particular cases of one of our Parameter Domain Knowledge types, namely the General Parameter Sharing Framework described in section 4.7. This framework is able to represent parameter sharing assumptions at parameter level of granularity, which previous models were not able to do. While the domain knowledge presented in this chapter can only accommodate simple equality constraints between parameters, we also derived closed form solutions for Parameter Domain Knowledge types that involve relationships between groups of parameters (sum sharing, ratio sharing). Moreover, we show how to compute closed form Maximum Likelihood estimators when the domain knowledge comes in the form of several types of inequality constraints. Along with our estimators come a series of formal guarantees that show the benefits of taking advantage of the available domain knowledge and also study the performance in the case when the domain knowledge might not be entirely accurate. Finally, we developed methods to automatically learn the domain knowledge, which we illustrate in Chapter 8 on a task of modelling the fMRI signal during a cognitive task.

Chapter 3

Approach

In this chapter we present a unified framework that allows us to take advantage of Parameter Domain Knowledge constraints in order to perform parameter learning in Bayesian Networks. There are two approaches to parameter learning. A frequentist tries to estimate one set of parameters that best explain the data, while a Bayesian assumes that different sets of parameters are possible, but some of them are more likely to occur. In the sections below we develop sound methods to incorporate Parameter Domain Knowledge in learning, from both a frequentist and Bayesian point of view, from both complete and incomplete data. The constraints that we deal with here do not need to have any specific form, but should satisfy certain differentiability assumptions. In the subsequent chapters we will develop more efficient methods tuned to specific types of Parameter Domain Knowledge.

3.1 The Problem

In this section we describe the problem and state several assumptions that we are making when deriving our estimators. These assumptions will also apply to most of the types of domain knowledge presented in the following chapter, unless otherwise specified.

The Problem. Our task is to perform parameter estimation in a Bayesian Network where the structure is known in advance. To accomplish this task, we assume a dataset of examples is available. In addition, a set of Parameter Domain Knowledge equality and/or inequality constraints is provided by a domain expert. Let $g_i(x) = 0$ for $1 \leq i \leq m$ be the equality constraints and let $h_j(x) \leq 0$ for $1 \leq j \leq k$ be the inequality constraints, where θ represents the set of parameters of the Bayesian Network.

In our parameter estimation methods we assume the domain knowledge provided by the expert is correct (in chapter 7, we investigate what happens if this knowledge is not entirely accurate).

Therefore, the space of feasible parameters is given by:

$$PDK = \{\theta \mid g_i(\theta) = 0 \;\; \forall\, i \in \{1, \ldots, m\}, \;\; h_j(\theta) \leq 0 \;\; \forall\, j \in \{1, \ldots, k\}\}$$

Next we will enumerate several assumptions that must be satisfied for our methods to work. These are similar to common assumptions made when learning parameters in standard Bayesian Networks. First of all, we consider that the examples in the training dataset are drawn independently at random from the underlying distribution. In other words, examples are conditionally independent given the parameters of the Graphical Model. Note that this assumption can be violated in the case of specific Graphical Models such as Probabilistic Relational Models, where objects are related to each other and the structure of the model varies depending on how many objects are in the dataset used for training.

Second, we assume that all the variables in the Bayesian Network can take at least two different values. This is a safe assumption since there is no uncertainty in a random variable with only one possible value. If there are such variables in our Bayesian Network, we can safely delete them, along with all the arcs that go in and out of the nodes corresponding to those variables.

Another important assumption that we will be using when computing parameter estimators in the discrete case later in this thesis is that all observed counts corresponding to parameters in the Bayesian Network are strictly positive. It is easy to see why we enforce this condition. Consider the case when all the observed counts associated to parameters in one conditional probability distribution are equal to zero. In this case we will have a *divide by zero* in the derivation of the maximum likelihood estimators in standard Bayesian Networks. It turns out that in this case any valid distribution would be a maximum likelihood estimator for the conditional probability distribution to estimate. This can be easily fixed by imposing the more relaxed constraint that the total observed count corresponding to that conditional probability distribution is strictly positive. This will ensure that there is a unique maximum likelihood estimator. However, if some of the observed counts (not the total observed count) are zero in that distribution, the maximum likelihood estimators will be zero for some parameters. Assume $N_{ijk} = 0$ for fixed values of i and j, but for all values of k. In other words, $X_i = x_{ij}$ was not observed when $PA(X_i) = pa_{ik}$ for all values of k. This means $\hat{\theta}_{ijk} = P(X_i = x_{ij}|PA(X_i) = pa_{ik}) = 0$ for all k and consequently $P(X_i = x_{ij}) = 0$. Therefore anything that is conditional on $X_i = x_{ij}$ is not defined. This impacts inference negatively. Even though these considerations are about parameter estimation in standard Bayes Nets, we are going to face the same kind of problems with the domain knowledge enhanced estimators. Therefore we decided to enforce the condition that all the observed counts are strictly positive when we are computing estimators in the discrete case. However, in the real world there will be observed counts which are zero. In this case we must use Dirichlet Priors and compute MAP estimators instead. Dirichlet Priors have the effect of adding a positive quantity to the observed count and essentially create strictly positive new counts.

Finally, in order for the results in this chapter to hold, the functions $g_1, \ldots, g_m$ and $h_1, \ldots, h_k$ must be twice differentiable, with continuous second derivatives. This will allow for certain Taylor series expansions that justify the Newton-Raphson method on the gradient of the Lagrangian function.

3.2 Frequentist Approach from Fully Observable Data

A frequentist tries to learn one single model that ”best” fits the data. When the structure of the model is known in advance, this often translates into finding the *Maximum Likelihood (ML)* estimators for the parameters in the model. Subsequently, we are also going to discuss *Maximum Aposteriori (MAP)* estimators.

In Chapter 2 we showed how one can derive Maximum Likelihood parameter estimators in standard Bayesian Networks by using the Lagrange Multipliers theorem. When training data is scarce, these estimators may have extremely high variance and therefore may violate the domain knowledge provided by the domain expert. In this section we will extend the result in Theorem 2.2.1 to an iterative procedure that computes Maximum Likelihood estimators that obey the domain knowledge specified by the expert. Without loss of generality, we will consider that parameter constraints that do not go into the log-likelihood function represent parameter domain knowledge. For example, in the case of standard Bayesian Networks, the equality constraints state that the parameters corresponding to a conditional probability distribution should sum up to one and the inequality constraints state that all parameters are positive.

Using the notations in the previous section, Maximum Likelihood estimators can be found by maximizing data log-likelihood $\log P(D|\theta)$ where $\theta \in PDK$. In the case of fully observable data, this is equivalent to solving the following Maximum Likelihood optimization problem:

$$\hat{\theta} = \quad \text{argmax} \;\; l(\theta) = \log P(D|\theta) = \sum_{d \in D} \sum_i \log P(X_i^{(d)} | PA_i^{(d)}, \theta) \tag{3.1}$$

$$\begin{cases} g_1(\theta) = 0 \\ \ldots \\ g_m(\theta) = 0 \\ h_1(\theta) \leq 0 \\ \ldots \\ h_k(\theta) \leq 0 \end{cases}$$

This maximization problem can be solved using Karush-Kuhn-Tucker theorem that we introduced in Chapter 2. The maximum point (if it exists), can be found by solving a system of equations. With potentially arbitrary complex parameter domain knowledge constraints, the solution of this system might not be computed in closed form. To see that, it is sufficient to note that there is no general closed form solution for a polynomial equation of degree higher than four. Fortunately, the Karush-Kuhn-Tucker theorem yields a system of equations with the same number of equations as variables, to which one can apply the Newton-Raphson iterative method (also described in Chapter 2) to compute its solutions. Putting the pieces together, the algorithm for computing Maximum Likelihood parameter estimators in the presence of Parameter Domain Knowledge is as follows:

Algorithm 3.2.1. (Maximum Likelihood estimation with Parameter Domain Knowledge)

STEP 1. *Build the following system of equations based on Karush-Kuhn-Tucker theorem:*

$$\begin{cases} \nabla_\theta l(\theta) - \sum_i \lambda_i \cdot \nabla_\theta g_i(\theta) - \sum_j \mu_j \cdot \nabla_\theta h_j(\theta) = 0 \\ g_i(\theta) = 0 \\ \mu_j \cdot h_j(\theta) = 0 \end{cases}$$

STEP 2. *Compute the solutions $(\hat{\theta}, \hat{\lambda}, \hat{\mu})$ of the above system using the Newton-Raphson method (potentially using random restarts to deal with local optima or possible divergence of this method when the initial guesstimate is far from a solution).*

STEP 3. *For the solutions with $\hat{\mu} \geq 0$, check if they are maximizers for the Maximum Likelihood optimization problem. The sufficiency criteria in propositions 2.1.1 and 2.1.2 might prove useful here. If the domain on which the log-likelihood and the constraints are defined is not a topologically open set, also check for potential maxima on the boundary. If multiple local maxima are found, keep the one that achieves the highest $l(\hat{\theta})$ value.*

With a large number of parameters in the Bayesian Network, the procedure described above can be extremely expensive because it involves potentially multiple runs of the Newton-Raphson method and each such run involves several expensive matrix inversions. Other methods for finding the solutions of a system of equations can be employed here, but, as noted in in [PTV93], all these methods have limitations in the case when the constraints are arbitrary, non-linear functions. The worst case happens when there exists a constraint that explicitly uses all parameters in the Bayesian Network. Fortunately, in practice, domain knowledge constraints may often only involve a small fraction of the total number of parameters. Also, the data log-likelihood can be nicely decomposed over examples, variables and values of the parents of each variable (in the case of discrete variables). Therefore, the Maximum Likelihood optimization problem can be split into a set of many independent, more manageable, optimization subproblems, on which we can apply the algorithm we just described. For example, in Theorem 2.2.1, each such subproblem was defined over one single conditional probability distribution. In general, in the discrete case, each optimization subproblem will span its own set of conditional probability distributions, as we will see later, in Chapter 4. The set of Maximum Likelihood parameters will be the union of the solutions of these subproblems.

3.3 Frequentist Approach from Incomplete Data

When data is only partially observable, we cannot write data log-likelihood as a double sum (equation 3.1) as we did in the previous section. However, one can represent the log-likelihood as the logarithm of a combination of sums and integrals over the missing values, depending on whether these values correspond to either discrete or continuous variables. Using this new objective function, we can still perform Maximum Likelihood estimation as described by Algorithm 3.2.1.

Because of the increase in complexity of the expression for data log-likelihood, this procedure can be prohibitively expensive. To see why this happens, let us take a look at the case when all variables are discrete and there is no additional domain knowledge (except for standard constraints that parameters in a conditional probability distribution should be positive numbers that sum up to one). If only one value is missing in every example, the complexity of performing Maximum Likelihood estimation using the above algorithm becomes slower by a factor exponential in the number of examples in the training dataset because the objective function will contain an exponential number of summands under the logarithm. Moreover, in this case, the system provided by Algorithm 3.2.1 doesn't have a closed form solution anymore, as it did in when the data was fully observable (see Theorem 2.2.1).

If continuous variables have missing values, additional complexity is incurred when evaluating the integrals over missing data, in the formula of the log-likelihood function. A common approach is to use the trapezoid method [Cor96], which approximates the function under the integral with a piecewise linear function.

An alternative to the approach described in the first paragraph of this section is to move the combination of sums and integrals over missing values outside the logarithm by taking the expectation of the log-likelihood of completed dataset over the missing values. If we restrict our search space to parameters that obey the Parameter Domain Knowledge constraints (i.e. $\theta \in PDK$), we obtain the following Expectation Maximization algorithm:

Algorithm 3.3.1. (Expectation Maximization with Parameter Domain Knowledge constraints) *Iterate using Algorithm 3.2.1 until convergence is reached:*

$$\hat{\theta}_{t+1} = \quad \textit{argmax}_\theta \; E_{P(Z|D,\hat{\theta}_t)} [\log P(D, Z|\theta)]$$
$$\begin{cases} g_1(\theta) = 0 \\ \quad \ldots \\ g_m(\theta) = 0 \\ h_1(\theta) \leq 0 \\ \quad \ldots \\ h_k(\theta) \leq 0 \end{cases}$$

This EM algorithm has the advantage that the data log-likelihood of the completed dataset is cheap to compute. However, we still have a very expensive algorithm because of the expectation taken over the missing data Z. To overcome the complexity issue, in [GH94], the authors suggest that missing values in a Gaussian Bayesian Network can be filled with their expectation given their parents under the current parameter estimates. Adapting this idea for parameter estimation in the presence of Parameter Domain Knowledge constraints yields the following algorithm that approximates (but is not guaranteed to find) the Maximum Likelihood estimators:

Algorithm 3.3.2. (Maximum Likelihood estimation by filling missing values) *Repeat the following two steps until convergence is reached:*

STEP 1. *Fill in the missing values with their expectations given their parents and current parameter estimates $\hat{\theta}_t$. Start with the variables that do not have any parents and end with the leaves of the Bayesian Network. By the end of this step, each example $d \in D$ is a completed example.*

STEP 2. *Use Algorithm 3.2.1 to re-estimate the Maximum Likelihood solution given the completed dataset from the previous step:*

$$\hat{\theta}_{t+1} = \quad \textit{argmax}_\theta \; \log \prod_{d \in D} \prod_i P(X_i^{(d)} | PA_i^{(d)}, \theta)$$
$$\begin{cases} g_1(\theta) = 0 \\ \quad \ldots \\ g_m(\theta) = 0 \\ h_1(\theta) \leq 0 \\ \quad \ldots \\ h_k(\theta) \leq 0 \end{cases}$$

Unlike the previous two algorithms, in the above procedure each iteration is as complex as the fully observable case. Unfortunately, this procedure is a greedy approach that is not guaranteed to converge to a local maximum of the likelihood function because at each iteration the missing data is instantiated to one value, while all other possible values are ignored. An even cheaper method would be to directly fill the missing values with their observed unconditional expectations, then directly resort to the full data case described in previous section. Even though the last approach is also not guaranteed to find the Maximum Likelihood estimators, it is frequently used in practice not only in models like Bayesian Networks, but also in Decision Trees and Neural Networks to fill in missing data.

In the discrete case, the complete data log-likelihood can be written as $l(\theta) = \sum_{i,j,k} N_{ijk} \cdot \log \theta_{ijk}$ while in the incomplete data case, the objective function of the EM algorithm can be expressed as $E_{P(Z|D,\hat{\theta}_t)} [\log P(D, Z|\theta)] = \sum_{i,j,k} E[N_{ijk}|\hat{\theta}_t] \cdot \log \theta_{ijk}$. It is easy to see that both complete data log-likelihood and the objective of the EM algorithm have the same structure and therefore, in the discrete case, each iteration of the EM algorithm will be same as expensive as Algorithm 3.2.1, provided that the expected counts are known. The EM algorithm in the discrete case becomes:

Algorithm 3.3.3. (Expectation Maximization for discrete Bayesian Networks) *Repeat the following two steps until convergence is reached:*

E-Step*: Use any inference algorithm to compute expected counts* $E[N_{ijk}|\hat{\theta}_t]$ *and* $E[N_{ik}|\hat{\theta}_t]$ *under the current parameter estimates* $\hat{\theta}_t$.

M-Step*: Re-estimate the parameters* $\hat{\theta}_{t+1}$ *using Algorithm 3.2.1, assuming that the observed counts are equal to the expected counts given by the E-Step.*

We have seen that, because of its simplicity, our extended version of the EM algorithm can be favored in the case of discrete Bayesian Networks. However, for Bayesian Networks that involve both continuous and discrete variables, one should carefully consider picking one of the four approaches mentioned in this section based on the complexity versus potential estimation inaccuracy trade-off.

3.4 Bayesian Approach from Fully Observable Data

To perform parameter learning from a Bayesian point of view, we need to define priors over the parameters of the Bayesian Network. In Chapter 2 we mentioned that these priors are Dirichlet priors in the case of discrete Bayesian Networks and Normal-Wishart priors for Gaussian Bayesian

Networks. When additional parameter constraints present, the region where parameters can vary is restricted further and therefore the above standard priors no longer reflect the Domain Knowledge. In this section we see how one can define parameter priors that are both consistent with the domain knowledge provided by the expert and allow developing procedures for Bayesian parameter learning. We will first describe these priors, then we will argue how Maximum Aposteriori estimators can be computed and finally we show how Bayesian model averaging can be performed for the purpose of predicting a new data point.

3.4.1 Constrained Parameter Priors

In the previous chapter we have seen that the Dirichlet prior is well suited for performing Maximum Aposteriori estimation of the parameters in a Bayesian Network because it is the conjugate of the probability distribution $P(D|\theta)$. In other words, $P(D|\theta)$ and $P(\theta|D) \propto P(D|\theta) \cdot P(\theta)$ are the same type of functions of θ for all complete datasets D, but with different hyperparameters. In the discrete case, for $P(D|\theta)$, the hyperparameters are the observed counts corresponding to each parameter while for $P(\theta|D)$ the hyperparameters are the observed counts incremented with the Dirichlet exponents. Following the same idea, we define *Constrained Parameter Priors* when Parameter Domain Knowledge constraints are provided.

Definition 3.4.1. *A Constrained Parameter Prior for a Bayesian Network with a set of Parameter Domain Knowledge constraints is a probability distribution such that:*

$$P(\theta) = \begin{cases} \frac{1}{Z_{PDK}} f(\theta) & \textit{if } \theta \in PDK \\ 0 & \textit{otherwise} \end{cases}$$

and $f(\theta)$ is chosen such that $P(\theta)$ is the conjugate of $P(D|\theta)$ for all possible complete datasets D. Consequently, $P(\theta|D)$ is also a Constrained Parameter Prior.

In general, choosing the function $f(\theta)$ is very intuitive, as he have seen in the previous chapter. For example, one can define $f(\theta) = P(D'|\theta)$ where D' is a fixed complete dataset of examples. Under the assumption that examples are drawn independently at random from the distribution described by the Bayesian Network, we obtain $P(\theta|D) \propto P(D \cup D'|\theta)$ and therefore $P(\theta)$ defined by f above is a valid *Constrained Parameter Prior*.

Computing the normalization constant Z_{PDK} is the most difficult part about dealing with Constrained Parameter Priors. We have $Z_{PDK} = \int_{\theta \in PDK} f(\theta)\, d\theta$. One may be tricked into thinking that this is a very simple task since there are many methods to either compute or approximate an integral. There are indeed many ways to accomplish this, but they rely on the fact that the region to integrate on is explicitly given (for example a one-dimensional interval, a rectangle or a sphere). In our case, the integration region is the intersection of regions described by potentially arbitrary equality and inequality constraints. In the general case it is impossible to parameterize the region PDK and that makes the task of computing the normalization constant Z_{PDK} very difficult. Another difficulty arises because the region PDK may not have the same dimension as the space of parameters and therefore the integral is a surface integral instead of being a volume integral. In this case, if we try to use a variant of trapezoid method by splitting the space in hypercubes, we cannot assume that the surface integral is additive over these hypercubes. The worst case happens when the surface has a part of positive area included in one of the faces of such a cube.

Here we present a method that will allow us to compute Z_{PDK} under the assumption that all Parameter Domain Knowlewdge constraints can be extended to well defined constraints over the whole multidimensional space of real numbers. This assumption allows us to consider, without loss of generality, that the domain of the Parameter Domain Knowledge constraints is the whole multidimensional space of real numbers because we can add any relationship that defines the domain of Parameter Domain Knowledge constraints as a separate constraint. With these considerations, one can approximate the constant Z_{PDK} using a sampling approach. The idea is to reduce the surface integral to a volume integral which can be computed by sampling. To perform this reduction, the constraints are approximated with linear functions on small hypercubes.

First, let us see how one can express a surface S parameterized by non-strict linear constraints as a non-zero volume. We will eliminate a variable if it can be expressed as a linear function of the other variables. After repeating this procedure for as long as possible, we are left with a region in a lower dimensional space that has positive volume. The dimension of the initial surface is defined as the dimension of this space. One can transform the surface integral into a volume integral of a new function by making the corresponding substitutions in the formula for f. The procedure looks is follows:

Algorithm 3.4.1. Computing the dimension of a surface S given by a set of linear constraints.

STEP 1. *Write each equality constraint as two inequality constraints such that the surface S can be described by $S = \{\theta \mid a_i^T \cdot \theta - b_i \leq 0 \,\forall\, 1 \leq i \leq m\}$.*

STEP 2. *Use the simplex method [Dan63] to solve:*

$$\alpha_i = \quad \mathit{min}\ a_i^T \cdot \theta - b_i \\ \begin{cases} a_1^T \cdot \theta - b_1 \leq 0 \\ \quad \ldots \\ a_m^T \cdot \theta - b_m \leq 0 \end{cases}$$

STEP 3. *If any of the above problems returns no feasible point, then the constraints are contradictory and we STOP. If $\alpha_i < 0 \,\forall\, 1 \leq i \leq m$, then STOP and declare $dim(S) = m$ because there* *exists a point strictly in the interior of the region defined by the remaining constraints. Otherwise, if there exists i such that $\alpha_i = 0$, this means the constraint i is tight and we can express one of the variables as a linear combination of the others. We therefore obtain a system of linear inequalities with one less variable and one less constraint. Also, it might happen that the coefficients of some of these new inequalities are all zeros (except for a negative free term), in which case we discard those constraints as redundant. We adjust a_i, b_i and m accordingly and then GO TO STEP 2.*

Next we will see how one can compute the value of an integral over a bounded surface defined by non-strict linear constraints. We assume that the bound is given by a hypercube H. The idea is to reduce the surface integral to a volume integral which can be computed using a sampling technique based on the Law of Large Numbers. We use the notation $S|H$ to denote the piece of the surface S that lies in H. In terms of content, this is the same as $S \cap H$, but should not be confused with the surface of potentially lower dimensionality that is obtained by the intersection of the constraints that describe S and H. In other words, $S|H$ should be seen as a piece of surface S as opposed to a new surface to integrate on.

Algorithm 3.4.2. Computing $\int_{\theta \in S|H} f(\theta)\, d\theta$ when S is a surface given by a set of linear constraints and H is a hypercube.

STEP 1. *Use Algorithm 3.4.1 to compute $m = dim(S)$ and $m' = dim(S \cap H)$. If $m' < m$ then STOP and declare $\int_{\theta \in S|H} f(\theta)\, d\theta = 0$. This happens because the area of $S \cap H$ is zero within surface S. If $m' = m$, it means that $S \cap H$ is a positive area surface within S and, using the elimination method described in Algorithm 3.4.1, we can characterize $S \cap H$ as a region of positive volume in a space described by a subset $\{\theta_1, \ldots, \theta_m\} \subseteq \theta$. We now reduced the initial surface integral $\int_{\theta \in S|H} f(\theta)\, d\theta$ to a volume integral $\int_{(\theta_1,\cdots,\theta_m) \in V} \bar{f}(\theta_1, \ldots, \theta_m)\, d\theta_1 \ldots d\theta_m$ where positive volume V is described by a series of constraints of the form $c^T \cdot (\theta_1, \ldots, \theta_m)^T - d \leq 0$. It is obvious that $V \subseteq H_{1..m}$, the m-dimensional hypercube which is the projection of H on dimensions $1, \ldots, m$. The second integral can now be computed using the sampling method described in STEPS 2 and 3.*

STEP 2.(Computing V) *Draw N uniform samples from $(\theta_1, \ldots, \theta_m) \in H_{1..m}$. Let N_V be the number of times a sample falls in V. Approximate $V = \frac{N_V}{N} \cdot V(H_{1..m})$.*

STEP 3.(Computing the integral) *It is easy to see that*

$$E[\bar{f}(Uniform(V))] = \int_{(\theta_1,\cdots,\theta_m) \in V} \frac{1}{V} \cdot \bar{f}(\theta_1, \ldots, \theta_m)\, d\theta_1 \ldots d\theta_m$$

We can compute $E[f'(Uniform(V))]$ using The Law of Large Numbers based on a sufficiently *large number of samples and let*

$$\int_{\theta \in S|H} f(\theta)\, d\theta = \int_{(\theta_1,\ldots,\theta_m)\in V} \bar{f}(\theta_1,\ldots,\theta_m)\, d\theta_1 \ldots d\theta_m = V \cdot E[\bar{f}(Uniform(V))]$$

Note that we could not use directly the sampling method on a surface which had volume zero and therefore we needed to reduce the surface integral to a volume integral defined on a region of positive volume. Also, the method breaks if the volume is infinite and that is why we required that the integral be computed within the limits of a hypercube. However, if the surface is unbounded, we can still use the above algorithm by taking the limit when the size of the bounding hypercube increases towards infinity. We are now ready to present a method to compute the normalization constant Z_{PDK} for arbitrary, non-linear constraints that obey the two assumptions discussed earlier in this section.

Algorithm 3.4.3. Computing the Normalization Constant Z_{PDK}.

STEP 1. *Run Steps 2 and 3 to compute $I_l = \int_{\theta \in PDK|H(l)} f(\theta)\, d\theta$ for $l \in \{1, 2, \ldots\}$. Here $H(l)$ represents the hypercube of size l centered at the origin of the axes. We have $Z_{PDK} = \lim_{l\to\infty} I_l$. In practice, we will stop once very little progress is made.*

STEP 2. *For a fixed small ϵ, split $H(l)$ in hypercubes of size ϵ: $H_1, \ldots, H_{s(\epsilon)}$. Run Step 3 to compute $I_{l,\epsilon} = \sum_{1\le t\le s(\epsilon)} \int_{\theta\in PDK|H_t} f(\theta)\, d\theta$. We have $I_l = \lim_{\epsilon\to 0} I_{l,\epsilon}$. In practice, we will use $\epsilon \in \{\frac{1}{2}, \frac{1}{3}, \ldots\}$ and we will stop once the progress made is below a certain threshold.*

STEP 3. *Within the hypercube H_t, approximate the constraints with linear constraints obtained via a first order Taylor expansion around any point in H_t. Apply Algorithm 3.4.2 to compute $\int_{\theta\in PDK|H_t} f(\theta)\, d\theta$.*

In the above algorithm, Step 1 is only needed in the case when the surface PDK is not bounded. In Step 2 the Parameter Domain Knowledge constraints are approximated by linear constraints within each small hypercube. This approximation allows us to transform the surface integral into a volume integral within each small hypercube so that we can apply Algorithm 3.4.2. We choose not to approximate the constraints with the same linear functions for all $H(l)$ because a first order Taylor expansion can potentially be very inaccurate on large regions of the space.

Figure 3.1 illustrates the above algorithm on the task of estimating the normalization constant for the unnormalized prior $f(\mu, \sigma)$ (for example, f can be an unnormalized Normal-Wishart prior) on parameters μ and σ of a Gaussian. The domain knowledge constraints are given by: $g(\mu, \sigma) = 4\sigma^2 + (2\mu - 1)^2 - 1 = 0$, $h_1(\mu, \sigma) = -\mu \le 0$ and $h_2(\mu, \sigma) = -\sigma \le 0$. The figure shows what happens when $l = 1$ and $\epsilon = \frac{1}{2}$. In Step 2 of algorithm 3.4.3, $H(1)$ is split in 16 smaller squares of

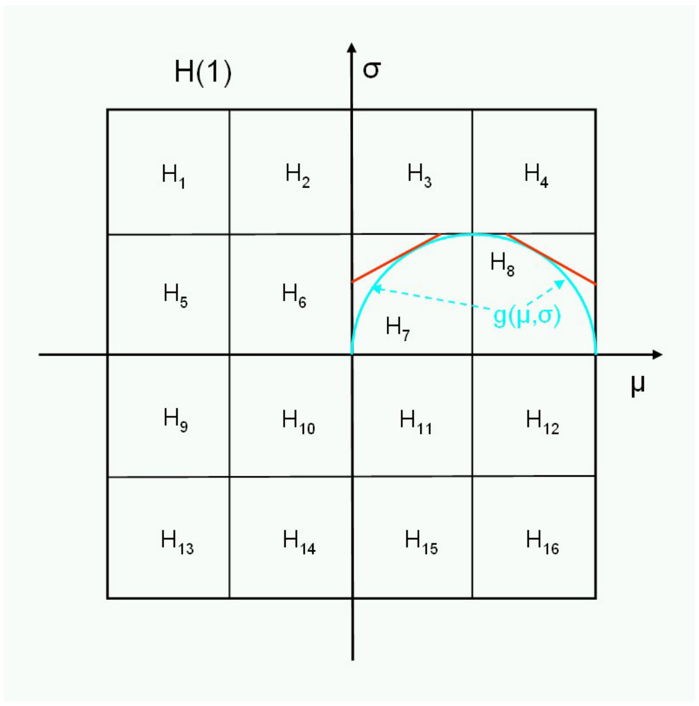

Figure 3.1: Algorithm 3.4.3 at work. The goal is to estimate the normalization constant for an unnormalized prior $f(\mu, \sigma)$ on parameters μ and σ of a Gaussian, given the constraints $g(\mu, \sigma) = 4\sigma^2 + (2\mu - 1)^2 - 1 = 0$, $h_1(\mu, \sigma) = -\mu \leq 0$ and $h_2(\mu, \sigma) = -\sigma \leq 0$.

size ϵ and then, in Step 3, the non-linear constraint g is approximated with a line segment in each of these squares. The only two squares where the approximation is non-empty are H_7 and H_8. In H_7, using a Taylor expansion around $(\frac{1}{4}, \frac{\sqrt{3}}{4})$, we have $g(\mu, \sigma) \sim -2(\mu - \frac{1}{4}) + \frac{8\sqrt{3}}{4}(\sigma - \frac{\sqrt{3}}{4})$. In H_8, using a Taylor expansion around $(\frac{3}{4}, \frac{\sqrt{3}}{4})$, we have $g(\mu, \sigma) \sim 2(\mu - \frac{3}{4}) + \frac{8\sqrt{3}}{4}(\sigma - \frac{\sqrt{3}}{4})$. The algorithm then proceeds by substituting σ in f with a linear function of μ (see algorithm 3.4.1) to obtain $\bar{f}(\mu)$. Within either of the squares H_7 or H_8, the integral of f is therefore simplified to the integral of $\bar{f}(\mu)$ where μ varies within an interval. To compute these two integrals of $\bar{f}(\mu)$, the sampling technique described in algorithm 3.4.2 is then used.

Note that, in order to have additivity in Step 2 of Algorithm 3.4.3, we assumed that the intersection of surface PDK with the boundaries of the hypercubes has area zero. However, for particular constraints and for particular values of ϵ, it may happen that PDK has some positive area subsurface that lies on a boundary of some of the hypercubes. Note however, that in Step 2 we are taking a limit when $\epsilon \to 0$ and therefore our algorithm still works if there exists a sequence $\epsilon_j \to 0$ that obeys our assumption. Alternatively, when positive areas of PDK are found on the boundaries of the hypercubes, one may subtract the value of the integrals on these areas from the sum in Step 2 so that they are counted only once.

In practice, the likelihood can be decomposed nicely and therefore the conjugate *Constrained Parameter Priors* will inherit the same property. Also, as mentioned before, each constraint will only relate a small fraction of the total number of parameters and therefore the integral $Z_{PDK} = \int_{\theta \in PDK} f(\theta)\ d\theta$ can be written as a product of several, easier to compute, independent integrals. As we will see in the following chapter, in many cases we will be able to compute the normalization constant in closed form and therefore we do not need to go through the expensive procedure described above.

3.4.2 Maximum Aposteriori Estimators

In chapter 2, we have seen that the problem of computing Maximum Aposteriori estimators can be reduced to the problem of computing Maximum Likelihood estimators in the case of fully observable data for standard Bayesian Networks. Let us see what happens in the presence of arbitrary Parameter Domain Knowledge constraints. To compute Maximum Aposteriori estimators, we need to maximize the posterior $P(\theta|D) \propto P(D|\theta) \cdot P(\theta)$ subject to Parameter Domain Knowledge constraints. If we pick *Constrained Parameter Priors* as described in the previous subsection, then $P(D|\theta)$ and $P(\theta|D)$ are the same type of functions of θ for all complete datasets D. Therefore, we can carry out Maximum Aposteriori estimation exactly in the same fashion with Maximum Likelihood estimation described earlier in this chapter.

3.4.3 Bayesian Model Averaging

From a Bayesian point of view, we are interested in predicting the next data point given previous data points by averaging over all possible models. This task can be written as follows:

$$P(d_{n+1}|d_n, \ldots, d_1) = \frac{\int_{\theta \in PDK} P(d_{n+1}, \ldots, d_1|\theta) \cdot P(\theta) d\theta}{\int_{\theta \in PDK} P(d_n, \ldots, d_1|\theta) \cdot P(\theta) d\theta}$$

If $P(\theta)$ is a *Constrained Parameter Prior*, then it follows from the definition that both the integrals above are normalization constants for different *Constrained Parameter Priors*. We have seen previously that these constants can be computed using sampling Algorithm 3.4.3 and therefore we have a method to perform Bayesian model averaging.

3.5 Bayesian Approach from Incomplete Data

When data is incomplete, one can easily adapt the methods described in section 3.3 to perform Maximum Aposterior estimation by multiplying the objective function by the *Constrained Param-*

eter Prior $P(\theta)$. For example, the iteration performed by a modified version of the Expectation Maximization algorithm will be :

$$\hat{\theta}_{t+1} = \quad \text{argmax}_{\theta \in PDK} \, E_{P(Z|D,\hat{\theta}_t)} \left[\log\left(P(D,Z|\theta) \cdot P(\theta)\right)\right]$$

From the definition of *Constrained Parameter Priors*, it follows that $P(D,Z|\theta) \cdot P(\theta)$ are the same type of functions of θ and therefore there is no additional complexity incurred when running the modified version of the EM algorithm to perform Maximum Aposteriori estimation subject to Parameter Domain Knowledge constraints.

The problem of performing Bayesian model averaging in the presence of Parameter Domain Knowledge constraints becomes more difficult when data is incomplete. In this case both integrals $\int_{\theta \in PDK} P(d_{n+1},\ldots,d_1|\theta) \cdot P(\theta)\, d\theta$ and $\int_{\theta \in PDK} P(d_n,\ldots,d_1|\theta) \cdot P(\theta)\, d\theta$ are no longer normalization constants for *Constrained Parameter Priors*, but they are a combination of sums and integrals (which can be evaluated using the trapezoid method) over the missing data of such normalization constants, depending on whether the missing values correspond to discrete or continuous variables. The amount of missing data determines the complexity of computing the above two integrals. If every example is missing at least one value, then the computation of the two integrals may require evaluating the normalization constants for an exponential number of *Constrained Parameter Priors*. This approach is again very expensive. One idea to lower its exponential complexity is to incrementally approximate the posterior of parameters with a *Constrained Parameter Prior* every time we observe a new example. However, investigating this approach is beyond the scope of this research.

3.6 Comparing Different Parameter Domain Knowledge Schemes

Define a *Parameter Domain Knowledge Scheme PDKS* over a Bayesian Network as a set of Parameter Domain Knowledge constraints for that network. As mentioned before, we denote by PDK the set of feasible parameters for such a scheme. We remind the reader that in all our work, the structure of the model is given and does not change. Thus, all *Parameter Domain Knowledge Schemes* will be defined over the same structure and parameter sets. Learning structure in the presence of parameter sharing is a subject for future work.

Later in this thesis we will see that taking advantage of a *Parameter Domain Knowledge Scheme* helps reduce the variance in parameter estimates. We will also show experimentally that parameter estimators based on Parameter Domain Knowledge yield distributions closer in KL distance to the true underlying distribution than the ones estimated without taking advantage of such knowledge.

Therefore it is important to recover as much Domain Knowledge as possible, even when the expert can specify only a limited part of it. Also, if two experts provide two different, potentially incorrect or inconsistent with each other, *Parameter Domain Knowledge Schemes*, we would like to be able to decide in a principled fashion which one (or which combination of the two) to use. To be able to do this, we need to come up with a method to score $PDKS$, a given *Parameter Domain Knowledge Scheme*. We propose a metric similar to the one used for structure search i.e. we try to find the $PDKS$ (among our possible choices) that maximizes $P(D|PDKS)$. Averaging over all sets of feasible parameters $\theta \in PDK$ for $PDKS$, we obtain:

$$P(D|PDKS) = \int_{\theta \in PDK} P(D|\theta,\ PDKS) \cdot P(\theta|PDKS)\, d\theta$$

Assuming that the prior $P(\theta|PDKS)$ is a *Constrained Parameter Prior*, it is easy to notice that the above integral is the normalization constant for a different *Constrained Parameter Prior* and therefore it can be computed using Algorithm 3.4.3 under the assumption that the data D is complete. In the case of incomplete data, the score is a combination of sums and integrals (over missing values) of normalization constants for *Constrained Parameter Priors*, which can be computed as mentioned in the previous section.

One drawback of this score is that it tries to find the *Parameter Domain Knowledge Scheme* that best fits the training data. An alternative is to compare *Parameter Domain Knowledge Schemes* based on their cross-validation log-likelihood. Briefly, the idea is to split the dataset in k folds, then train a model form a frequentist point of view using $k-1$ folds and the *Parameter Domain Knowledge Scheme,* then use that model to compute the log-likelihood of the remaining fold. Repeating this procedure for each fold yields the log-likelihood for all dataset D, computed in such a way that test data was not used in training. Because of this fact and because it is much faster to perform frequentist parameter learning, we suggest this measure is to be preferred instead of $P(D|PDKS)$ when comparing different *Parameter Domain Knowledge Schemes*. This cross-validation score was employed in our experiments on fMRI data in Chapter 8 to derive the optimal clustering of brain voxels during a cognitive task.

The scores defined above can be used to choose between several different available *Parameter Domain Knowledge Schemes*. One can also imagine an automated approach to learning an optimal *Parameter Domain Knowledge Scheme* from training data via some hill climbing techniques. We can derive the current scheme candidate from previous one by using small modifications, in a fashion similar to the one employed by structure search. The search space however is prohibitively large, so restricting the set of potential candidates can be extremely useful. For instance, in module networks one can restrict the variables that can belong together in a module. In addition, an expert who knows a subset of Parameter Domain Knowledge constraints can restrict further the search space.

Chapter 4

Equality Constraints for Discrete Variables

In the previous chapter we developed general methods to perform parameter learning in Bayesian Networks when a domain expert specifies in advance a set of Parameter Domain Knowledge constraints. While these methods can deal with arbitrary parameter constraints that obey several smoothness assumptions, they can potentially be very slow since they involve expensive iterative and sampling procedures. However, for certain particular types of Parameter Domain Knowledge, we do not need to go into the Newton-Raphson iterative procedure because the system of equations in Algorithm 3.2.1 can be solved in closed form. Also, in some of these cases we might be able to find a closed form formula for the normalization constant of the corresponding Constrained Parameter Prior. This and the following two chapters address the problem of finding such closed form solutions for several types of Parameter Domain Knowledge.

In the next sections we look at incorporating parameter equality constraints in learning algorithms for discrete Bayesian Networks. We have already shown in Chapter 3 that, in the discrete case, the Expectation Maximization algorithm to deal with missing data as well as the Bayesian approach for learning from both complete and incomplete data can be derived in a straightforward fashion from the Maximum Likelihood estimators procedure and based on the normalization constant of the Constrained Parameter Prior. Therefore, for all types of domain knowledge that we introduce in this chapter, we only need to come up with closed form solutions for the Maximum Likelihood estimators from complete data and for the normalization constant for the corresponding Constrained Parameter Priors. The conjugate of the multinomial distribution is the Dirichlet distribution and therefore our Constrained Parameter Priors will be *Constrained Dirichlet Priors*.

The results in this chapter build on work previously presented in [NMR05]. We start by analyzing several types of Parameter Domain Knowledge equality constraints that span only inside conditional probability distributions and then we extend these results to domain knowledge types that involve parameters across multiple such distributions. A set of conditional probability distributions will form the scope of a Parameter Domain Knowledge constraint. Different constraint types can be defined over disjoint sets of conditional probability distributions, but only one constraint type can be applied to any such set. Because of the decomposability of data log-likelihood, one can learn independently the parameters involved in each type of domain knowledge constraint. Also, the normalization constant for the Constrained Dirichlet Prior can be computed over the scope of a certain constraint and then all such constants are multiplied to obtain the normalization constant for the prior over the whole set of parameters of the Bayesian Network.

4.1 A Note on Normalization Constants and Dirichlet Integrals

When computing the normalization constant of a Dirichlet distribution over $\theta = (\theta_1, \ldots, \theta_n)$, the idea is to write $\theta_n = 1 - \sum_{i=1}^{n-1} \theta_i$ and compute the normalization constant Z_n for:

$$p(\theta_1, \ldots, \theta_{n-1}) = \frac{1}{Z_n} \cdot (1 - \sum_{i=1}^{n-1} \theta_i)^{\alpha_n - 1} \cdot \prod_{i=1}^{n-1} \theta_i^{\alpha_i - 1}$$

If we eliminated θ_1 instead of θ_n, we would have estimated the normalization constant for a distribution over $(\theta_2, \ldots, \theta_n)$ instead. Luckily, all these constants are equal no matter which variable we eliminate. However, with Constrained Dirichlet Priors, this will not be the case anymore. For example, consider learning with the constraint $\sum_i b_i \cdot \theta_i = 1$, which may appear in the case when certain parameters are shared within the same distribution. Here parameter θ_i appears in b_i different known places in the distribution. We are looking for *Generalized Dirichlet Priors* of the form:

$$P(\theta) = \begin{cases} \frac{1}{Z} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1} & \text{if } \theta \geq 0, \sum b_i \cdot \theta_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

If we eliminate θ_n, we have to estimate the *Generalized Dirichlet Integral*:

$$Z_n = \int_{\theta \geq 0, \sum b_i \cdot \theta_i = 1} (\frac{1 - \sum_{i=1}^{n-1} b_i \cdot \theta_i}{b_n})^{\alpha_n - 1} \cdot \prod_{i=1}^{n-1} \theta_i^{\alpha_i - 1} \, d\theta_1 \, d\theta_2 \ldots \, d\theta_{n-1} \tag{4.1}$$

This integral is reduced to a standard Dirichlet Integral by making the substitution $x_i = b_i \cdot \theta_i$ for $1 \leq i \leq n - 1$. We obtain:

$$Z_n = \frac{b_n}{\prod_{i=1}^{n} b_i^{\alpha_i}} \cdot \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)} \tag{4.2}$$

If we eliminated θ_1 first, we would have obtained a potentially different normalization constant Z_1 over the remaining $n-1$ parameters. In the case of standard Dirichlet Distribution, all b_i were equal to 1. Even though the n possible distributions over $n-1$ have different normalization constants, they are essentially the same distribution as they can be obtained from one another via a variable substitution.

In general, when coming up with Constrained Parameter Priors, we eliminate several parameters and we actually compute a distribution over a smaller subset of parameters that determine the eliminated ones in a deterministic fashion. Depending on the elimination order, we may obtain different normalization constants. Note however, that the elimination order does not matter for any of our learning procedures. Therefore it will be enough to show that we can compute the normalization constant for any elimination order.

4.2 Known Parameters

The simplest type of Parameter Domain Knowledge consists of *Known Parameters*. The domain expert directly specifies the values of certain parameters in the Bayesian Network. The corresponding constraints have the form $\theta_{ijk} = c$ where c is a known value. For example, an expert can state: *"If a patient has a heart attack (Disease = "Heart Attack"), then there is a 90% probability that the patient will experience chest pain."*

4.2.1 Maximum Likelihood Estimation from Complete Data

The above constraints do not span across multiple conditional probability distributions and therefore, because of the decomposability of log-likelihood, we can break the bigger Maximum Likelihood optimization problem into a set independent subproblems, one for each conditional probability distribution. If θ_i are the unknown parameters of a conditional probability distribution, N_i their corresponding counts, $N = \sum_i N_i$ the total observed counts and S the sum of known parameters of this distribution, we have the following theorem:

Theorem 4.2.1. *The Maximum Likelihood Estimators for parameters θ in our distribution are given by:*

$$\hat{\theta}_i = (1-S) \cdot \frac{N_i}{N}$$

Proof. Our optimization problem becomes:

$$P : argmax\ \{h(\theta) \mid g(\theta) = 0\}$$

where $h(\theta) = \sum_i N_i \log \theta_i$ and $g(\theta) = (\sum_i \theta_i) - (1-S) = 0$

When all counts are positive, it can be easily proved that P has a global maximum which is achieved in the interior of the region determined by the constraints. In this case the solution of P can be found using Lagrange Multipliers. Introduce Lagrange Multiplier λ for the constraint in P. Let $LM(\theta, \lambda) = h(\theta) - \lambda \cdot g(\theta)$. Then the point which maximizes P is among the solutions of the system $\nabla LM(\theta, \lambda) = 0$. Let $(\hat{\theta}, \lambda)$ be a solution of this system. We have:

$0 = \frac{\partial LM}{\partial \theta_i} = \frac{N_i}{\theta_i} - \lambda$ for all i. Therefore: $\hat{\theta}_i = \frac{N_i}{\lambda}$. Summing up for all values of i, we obtain: $0 = \frac{\partial LM}{\partial \lambda} = (\sum_i \hat{\theta}_i) - (1 - S) = (\sum_i \frac{N_i}{\lambda}) - (1 - S) = \frac{N_i}{\lambda} - (1 - S)$

From the last equation we compute the value of $\lambda = N_i$. This gives us: $\hat{\theta}_i = \frac{N_i}{N}$. From Sufficiency Criterion 2.1.1, it follows that $\hat{\theta}$ is the set of Maximum Likelihood estimators. □

4.2.2 Constrained Dirichlet Priors

For this type of domain knowledge, the Constrained Dirichlet priors have the following form:

$$P(\theta) = \begin{cases} \frac{1}{Z}\prod_{i=1}^{n} \theta_i^{\alpha_i - 1} & \text{if } \theta \geq 0, \sum \theta_i = 1 - S \\ 0 & \text{otherwise} \end{cases}$$

This a Generalized Dirichlet Prior with $b_i = \frac{1}{1-S}$ and therefore the normalization constant is given by:

$$Z_n = (1 - S)^{\sum_{i=1}^{n} \alpha_i - 1} \cdot \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)}$$

4.3 Parameter Sharing within One Distribution

Here we allow certain parameters to be shared within the same conditional probability distribution. This corresponds to statements like: *"Given this combination of causes, several effects are equally likely"*. Since the scope of these additional constraints does not go beyond the conditional probability distribution level, the problem of maximizing the data likelihood can again be split in a set of independent optimization subproblems, one for each such conditional probability distribution. Let's look at one of these subproblems (for a variable X and a specific value $PA(X) = pa$ of the parents). Assume that a domain expert is stating that certain parameters are equal: parameter θ_i appears in k_i different positions in our distribution. Denote by N_i the cumulative observed count corresponding to θ_i. The cumulative observed count is the sum of all the observed counts corresponding to the k_i positions where θ_i appears in the distribution. Let $N = \sum_i N_i$ be the sum of all observed counts in this conditional probability distribution i.e. the total number of observed cases with $PA(X) = pa$.

One may believe that Maximum Likelihood estimation can be performed using standard methods by introducing new variables that describe what group of shared parameters a given parameter belongs to. To see that this is not the case, consider the following example. Assume a variable X with values $\{1,2,3,4\}$ depends on Y. Moreover, assume the expert is stating that $P(X=1|Y=0) = P(X=2|Y=0)$ and $P(X=3|Y=0) = P(X=4|Y=0)$. Then one can introduce variable X_{12} which is 1 if $X \in \{1,2\}$ and 0 otherwise. This variable is assumed dependent on Y and added as a parent of X. It is easy to see that $P(X|X_{12}=0, Y=0)$ must be equal to the distribution on $\{1,2,3,4\}$ that assigns half probability to each of 3 and 4. Therefore, if Y takes only one value, the task of finding Maximum Likelihood estimators with Parameter Sharing is reduced to the one of finding standard Maximum Likelihood estimators for $X_{12}|Y=0$. However, if Y takes only one value, then we can safely remove it as a parent of X. When Y can take two values, 0 and 1, assume the expert states the additional assumption that $P(X=1|Y=1) = P(X=3|Y=1) = P(X=4|Y=1)$. Now we need to introduce a new variable X_{134} that depends on Y and add it as a parent of X. It is straightforward to see that in this case, the conditional $P(X|X_{12}=0, X_{134}=1, Y=1)$ is not a constant distribution anymore and therefore the above approach of reducing our parameter sharing problem to a Maximum Likelihood optimization problem in standard Bayesian Networks fails. Also, the structural assumption that X_{12} and X_{134} are conditionally independent given Y is not true. Same argument holds for all types of Parameter Domain Knowledge presented in this chapter.

4.3.1 Maximum Likelihood Estimation from Complete Data

Theorem 4.3.1. *The Maximum Likelihood Estimators for the parameters in the above conditional probability distribution are given by:*

$$\hat{\theta}_i = \frac{N_i}{k_i \cdot N}$$

Proof. Our optimization subproblem can be restated as:

$$P : argmax\ \{h(\theta) \mid g(\theta) = 0\}$$

where $h(\theta) = \sum_i N_i \log \theta_i$ and $g(\theta) = (\sum_i k_i \cdot \theta_i) - 1 = 0$

When all counts are positive, it can be easily proved that P has a global maximum which is achieved in the interior of the region determined by the constraints. In this case the solution of P can be found using Lagrange Multipliers. Introduce Lagrange Multiplier λ for the constraint in P. Let $LM(\theta,\lambda) = h(\theta) - \lambda \cdot g(\theta)$. Then the point which maximizes P is among the solutions of the system $\nabla LM(\theta,\lambda) = 0$. Let $(\hat{\theta},\lambda)$ be a solution of this system. We have: $0 = \frac{\partial LM}{\partial \theta_i} = \frac{N_i}{\theta_i} - \lambda \cdot k_i$ for all i. Therefore, $k_i \cdot \hat{\theta}_i = \frac{N_i}{\lambda}$. Summing up for all values of i, we obtain:

$$0 = \frac{\partial LM}{\partial \lambda} = (\sum_i k_i \cdot \hat{\theta}_i) - 1 = (\sum_i \frac{N_i}{\lambda}) - 1 = \frac{N}{\lambda} - 1$$

From the last equation we compute the value of $\lambda = N$. This gives us: $\hat{\theta}_i = \frac{N_i}{k_i \cdot N}$. The fact that $\hat{\theta}$ is the set of Maximum Likelihood estimators follows again from Sufficiency Criterion 2.1.1. $\square$

4.3.2 Constrained Dirichlet Priors

For this type of domain knowledge, the Constrained Dirichlet priors have the following form:

$$P(\theta) = \begin{cases} \frac{1}{Z} \prod_{i=1}^{n} \theta_i^{\alpha_i - 1} & \text{if } \theta \geq 0, \sum k_i \cdot \theta_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

As we have seen in section 4.1, the normalization constant depends on the elimination order. If θ_n is eliminated first, we get:

$$Z_n = \frac{k_n}{\prod_{i=1}^{n} k_i^{\alpha_i}} \cdot \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)}$$

4.4 Proportionality Constants within One Distribution

This is a slight generalization of Parameter Sharing. We partition θ, the set of parameters of a conditional probability distribution in subsets $S_i = \{\theta_{i1}, \theta_{i2}, \ldots\}$ such that the parameters in S_i are proportional to given constants $c_{i1}, c_{i2}, \ldots$. This corresponds to statements like: *"Given a combination of health risk factors, disease A is twice as likely to occur than disease B is."*. A set of shared parameters are proportional with a set of ones. An unconstrained parameter can be thought of as a parameter that is shared in just one place. Let $c_i = \sum_j c_{ij}$ be the sum of the constants corresponding to S_i and $N_i = \sum_j N_{ij}$ the sum of the observed counts of parameters in S_i. Also, let $N = \sum_{i,j} N_{ij}$ be the total number of observations we have about our conditional probability distribution. With these notations, we are ready to present the main result of this section.

4.4.1 Maximum Likelihood Estimation from Complete Data

Theorem 4.4.1. *The Maximum Likelihood Estimators for the parameters in the above distribution are given by:*

$$\hat{\theta}_{ij} = \frac{c_{ij}}{c_i} \cdot \frac{N_i}{N}$$

Proof. Let γ_i be the proportionality factor for set S_i. We are going to make a change of variable from θ to γ. We have $\theta_{ij} = c_{ij} \cdot \gamma_i$ for all i, j. Therefore maximizing $\sum_{i,j} N_{ij} \cdot \log \theta_{ij}$ is equivalent to maximizing $\sum_{i,j} N_{ij} \cdot \log \gamma_i = \sum_i N_i \cdot \log \gamma_i$. The constraint $\sum_{i,j} \theta_{ij} = 1$ is equivalent to $\sum_{i,j} c_{ij} \cdot \gamma_i = \sum_i c_i \cdot \gamma_i = 1$.

Our optimization subproblem can be restated as:

$$P : argmax \; \{h(\gamma) \mid g(\gamma) = 0\}$$

$$\text{where } h(\gamma) = \sum_i N_i \cdot \log \gamma_i \text{ and } g(\gamma) = (\sum_i c_i \cdot \gamma_i) - 1 = 0$$

In this case the solution of P can be found using Lagrange Multipliers. Introduce Lagrange Multiplier λ for the constraint in P. Let $LM(\gamma, \lambda) = h(\gamma) - \lambda \cdot g(\gamma)$. Then the point which maximizes P is among the solutions of the system $\nabla LM(\gamma, \lambda) = 0$. Let (γ, λ) be a solution of this system. We have:

$0 = \frac{\partial LM}{\partial \gamma_i} = \frac{N_i}{\gamma_i} - \lambda \cdot c_i$ for all i. Therefore: $c_i \cdot \gamma_i = \frac{N_i}{\lambda}$. Summing up for all values of i, we obtain: $0 = \frac{\partial LM}{\partial \lambda} = (\sum_i c_i \cdot \gamma_i) - 1 = (\sum_i \frac{N_i}{\lambda}) - 1 = \frac{N}{\lambda} - 1$

From the last equation we compute the value of $\lambda = N$. This gives us: $\gamma_i = \frac{N_i}{c_i \cdot N}$. Therefore the estimators for the parameters in our model can be computed as $\hat{\theta}_{ij} = c_{ij} \cdot \gamma_i = \frac{c_{ij}}{c_i} \cdot \frac{N_i}{N}$. From Sufficiency Criterion 2.1.1, it follows that $\hat{\theta}$ is the set of Maximum Likelihood estimators. $\square$

4.4.2 Constrained Dirichlet Priors

For this type of domain knowledge, we define priors on the hyperparameters γ:

$$P(\theta) = \begin{cases} \frac{1}{Z} \prod_{i=1}^{n} \gamma_i^{\alpha_i - 1} & \text{if } \gamma \geq 0, \sum c_i \cdot \gamma_i = 1 \\ 0 & \text{otherwise} \end{cases}$$

This is again a *Generalized Dirichlet Distribution* and therefore the normalization constant given that we first eliminate γ_n is:

$$Z_n = \frac{c_n}{\prod_{i=1}^{n} c_i^{\alpha_i}} \cdot \frac{\prod_{i=1}^{n} \Gamma(\alpha_i)}{\Gamma(\sum_{i=1}^{n} \alpha_i)}$$

4.5 Sum Sharing within One Distribution

Sum Sharing is similar to Parameter Sharing, but here several sets of parameters within one distribution have the same aggregate probability mass. If two sets of parameters A and B have this property we will write $A \doteq B$. This corresponds to statements like: *"A patient who is a smoker has the same chance of having a Heart Disease (Heart Attack or Congestive Heart Failure) as having a Pulmonary Disease (Lung Cancer or Chronic Obstructive Pulmonary Disease).*". Formally, suppose a domain expert tells us that within one of the distributions in the graphical model, $S_{11} \doteq S_{12} \doteq \ldots \doteq S_{1k_1}$, …, $S_{l1} \doteq S_{l2} \doteq \ldots \doteq S_{lk_l}$ where the sets of parameters S_{ij} are mutually disjoint. Equivalently, the expert can state that the sets in $S_{i*} = \{S_{i1}, S_{i2}, \ldots, S_{ik_i}\}$ have the same sum and that holds for all i between 1 and l. We can consider that each parameter in the given distribution belongs to one of S_{ij}. Otherwise we can create a new $S_{(l+1)1}$ which contains all these parameters. Denote by θ_{ij}^{k} the k^{th} parameter in S_{ij} and by N_{ij}^{k} its corresponding observed count. Again, let $N = \sum_{i,j,k} N_{ij}^{k}$ be the total number of samples from this conditional probability distribution.

4.5.1 Maximum Likelihood Estimation from Complete Data

Theorem 4.5.1. *The Maximum Likelihood Estimators for the parameters θ_{ij}^{h} are given by:*

$$\hat{\theta}_{ij}^{k} = \frac{N_{ij}^{k}}{\sum_{k'} N_{ij}^{k'}} \cdot \frac{\sum_{j,k} N_{ij}^{k}}{k_i \cdot N}$$

Proof. Denote by s_i the sum of parameters in any of the sets S_{ij}. The constraints of our problem can be rewritten as $\sum_k \theta_{ij}^{k} = s_i$ for all i, j and $\sum_i k_i \cdot s_i = 1$. Our optimization subproblem can be restated as:

$$P : argmax \; \{h(\theta, s) \mid g(\theta, s) = 0, \; g_{ij}(\theta, s) = 0\}$$

where $h(\theta, s) = \sum_{i,j,k} N_{ij}^{k} \cdot \log \theta_{ij}^{k})$, $g(\theta, s) = (\sum_i k_1 \cdot s_i) - 1 = 0$ and
$g_{ij}(\theta, s) = (\sum_k \theta_{ij}^{k} - s_i) = 0$

In this case the solution of P can be found using Lagrange Multipliers. Introduce Lagrange Multiplier λ for the constraint g and lagrange Multipliers τ_{ij} for constraint g_{ij}. Let $LM(\theta, s, \lambda, \tau) = h(\theta, s) - \lambda \cdot g(\theta, s) - \sum_{i,j} \tau_{ij} \cdot g_{ij}(\theta, s)$. Then the point which maximizes P is among the solutions of the system $\nabla LM(\theta, s, \lambda, \tau) = 0$. Let $(\hat{\theta}, s, \lambda, \tau)$ be a solution of this system. We have:

$$0 = \frac{\partial LM}{\partial \theta_{ij}^{k}} = \frac{N_{ij}^{k}}{\theta_{ij}^{k}} - \tau_{ij} \forall i, j \tag{4.3}$$

$$0 = \frac{\partial LM}{\partial s_i} = -k_i \cdot \lambda + \sum_j \tau_{ij} \forall i \tag{4.4}$$

Therefore: $\hat{\theta}_{ij}^{k} = \frac{N_{ij}^{k}}{\tau_{ij}}$. Summing up for all values of k, we obtain:

$$0 = \frac{\partial LM}{\partial \tau_{ij}} = -(\sum_k \hat{\theta}_{ij}^{k} - s_i) = s_i - \frac{\sum_k N_{ij}^{k}}{\tau_{ij}}$$

Therefore:

$$\tau_{ij} = \frac{\sum_k N_{ij}^{k}}{s_i}$$

Summing this up for all values of j we get: $0 = \frac{\partial LM}{\partial s_i} = -\lambda \cdot k_i + \sum_j \tau_{ij} = -\lambda \cdot k_i + \frac{\sum_{j,k} N_{ij}^k}{s_i}$
Therefore

$$k_i \cdot s_i = \frac{\sum_{j,k} N_{ij}^k}{\lambda}$$

. Summing this for all values of i yields:
$0 = \frac{\partial LM}{\partial \lambda} = -(\sum_k k_i \cdot s_i - 1) = 1 - \frac{\sum_{i,j,k} N_{ij}^k}{\lambda} = 1 - \frac{N}{\lambda}$
From the last equation we compute the value of $\lambda = N$. This gives us:

$$s_i = \frac{\sum_{j,k} N_{ij}^k}{k_i \cdot N}$$

$$\tau_{ij} = (\sum_k N_{ij}^k) \cdot \frac{k_i \cdot N}{\sum_{j,k} N_{ij}^k}$$

$$\hat{\theta}_{ij}^k = \frac{N_{ij}^k}{\sum_{k'} N_{ij}^{k'}} \cdot \frac{\sum_{j,k} N_{ij}^k}{k_i \cdot N}$$

The fact that $\hat{\theta}$ is the set of Maximum Likelihood estimators is again a consequence of sufficiency conditions 2.1.1. $\square$

4.6 Ratio Sharing within One Distribution

Ratio Sharing is similar to the Proportionality type of Domain Knowledge defined in section 4.4, but here several equally sized sets of parameters are proportional to each other. In other words, these sets of parameters can be obtained from one another by multiplying with a constant. If two sets of parameters A and B have this property we will write $A \sim B$. This type of domain knowledge corresponds to statements like: *”In a bilingual corpus, the relative frequencies of certain groups of words are the same, even though the aggregate frequencies of these groups may be different.*” Such groups of words can be: ”words about computers” (”computer”, ”mouse”, ”monitor”, ”keyboard” in both languages) or ”words about business”, etc. In some countries computer use is more extensive than in others and one would expect the aggregate probability of ”words about computers” to be different. However, it would be natural to assume that the relative proportions of the ”words about computers” are the same within the different languages.

Formally, suppose a domain expert tells us that within one of the distributions in the Bayesian Network, $T_{11} \sim T_{12} \sim \ldots \sim T_{1k_1}$, …, $T_{l1} \sim T_{l2} \sim \ldots \sim T_{lk_l}$ where the sets of parameters T_{ij} are mutually disjoint. Equivalently, the expert can state that the sets in $T_{i*} = \{T_{i1}, T_{i2}, \ldots, T_{ik_i}\}$ are proportional to each other and that holds for all i between 1 and l. We can consider that each parameter in the given distribution belongs to one of T_{ij}. Otherwise we can create a new $T_{(l+1)1}$ which contains all these parameters. Denote by θ_{ij}^k parameter in position k in T_{ij}. Because of our ratio sharing assumption, the position of a parameter within a set T_{ij} matters. As before, let $N = \sum_{i,j,k} N_{ij}^k$ be the total number of observed counts corresponding to our conditional probability distribution.

4.6.1 Maximum Likelihood Estimation from Complete Data

Theorem 4.6.1. *The Maximum Likelihood Estimators for the parameters in the above distribution are given by:*

$$\hat{\theta}_{ij}^k = \frac{\sum_j N_{ij}^k \cdot \sum_k N_{ij}^k}{N \cdot \sum_{j,k} N_{ij}^k}$$

Proof. Because $T_{i1} \sim T_{i2} \sim \ldots$, there must exist a vector $p_i = (p_{i1}, p_{i2}, p_{i3}, \ldots)$ and multiplication factors $c_{i1}, c_{i2}, \ldots$ such that $T_{ij} = c_{ij} \cdot p_i$ or, equivalently, $\theta_{ij}^k = c_{ij} \cdot p_{ik}$ for all i, j, k. Therefore maximizing $\sum_{i,j,k} N_{ij}^k \cdot \log \theta_{ij}^k$ is equivalent to maximizing $\sum_{i,j,k} N_{ij}^k \cdot \log{(c_{ij} \cdot p_{ik})}$. The constraint $\sum_{i,j,k} \theta_{ij}^k = 1$ is equivalent to $\sum_{i,j,k} c_{ij} \cdot p_{ik} = 1$.

Our optimization subproblem can be restated as:

$$P : argmax \; \{h(c,p) \mid g(c,p) = 0\}$$

where $h(c,p) = \sum_{i,j,k} N_{ij}^k \cdot \log{(c_{ij} \cdot p_{ik})}$ and $g(c,p) = (\sum_{i,j,k} c_{ij} \cdot p_{ik}) - 1 = 0$

In this case the solution of P can be found using Lagrange Multipliers. Introduce Lagrange Multiplier λ for the constraint in P. Let $LM(c,p,\lambda) = h(c,p) - \lambda \cdot g(c,p)$. Then the point which maximizes P is among the solutions of the system $\nabla LM(c,p,\lambda) = 0$. Let (c,p,λ) be a solution of this system. We have:

$$0 = \frac{\partial LM}{\partial c_{ij}} = \frac{\sum_k N_{ij}^k}{c_{ij}} - \lambda \cdot \sum_k p_{ik} \forall i,j \tag{4.5}$$

$$0 = \frac{\partial LM}{\partial p_{ik}} = \frac{\sum_j N_{ij}^k}{p_{ik}} - \lambda \cdot \sum_j c_{ij} \forall i,k \tag{4.6}$$

Therefore: $c_{ij} \cdot \sum_k p_{ik} = \frac{\sum_k N_{ij}^k}{\lambda}$. Summing up 4.5 for all values of i and j, we obtain: $0 = \frac{\partial LM}{\partial \lambda} = (\sum_i c_{ij} \cdot p_{ik}) - 1 = (\sum_i \frac{N_{ij}^k}{\lambda}) - 1 = \frac{N}{\lambda} - 1$

From the last equation we compute the value of $\lambda = N$. This gives us: $c_{ij} = \frac{N_{ij}^k}{N \cdot \sum_k p_{ik}}$. Consequently, $\sum_j c_{ij} = \frac{\sum_{j,k} N_{ij}^k}{N \cdot \sum_k p_{ik}}$. Using 4.6 we get $p_{ik} = \frac{\sum_j N_{ij}^k}{\sum_{j,k} N_{ij}^k} \cdot \sum_{k'} p_{ik'}$. For each value of

$\sum_{k'} p_{ik'}$ we will obtain a stationary point of LM. However, for all these stationary points we get $\hat{\theta}_{ij}^k = c_{ij} \cdot p_{ik} = \frac{\sum_j N_{ij}^k \cdot \sum_k N_{ij}^k}{N \cdot \sum_{j,k} N_{ij}^k}$. Let $(\hat{p}, \hat{c})$ be one such maximizer for h and let:

$$B = \{(p, c) \mid c_{ij} \cdot p_{ik} \geq e^{2 \cdot h(\hat{p}, \hat{c})} \; \forall \, i, j, k\}$$

It is obvious that $h(b) < h(\hat{p}, \hat{c})$ for all $b \notin B$. Now, from Sufficiency Criterion 2.1.2, it follows that $\hat{\theta}$ is the set of Maximum Likelihood estimators. □

4.7 A General Parameter Sharing Framework

Here we present a General Parameter Sharing Framework that describes learning in a broad category of graphical models. A *General Parameter Sharing* assumption is specified by an expert over a set C of conditional probability distributions. Multiple such assumptions can be provided for disjoint sets of distributions, potentially covering all the set of conditional probability distributions in the Bayesian Network. Every assumption states that some parameters (denote their set by G_k) are shared (appear exactly once in each of the different distributions) within the set C, but not shared within the same distribution or outside C. For example, if we consider the Bayesian Network in Figure 1.1, a parameter sharing assumption may state: *”The probability that a person will have a heart attack given that he/she is a smoker with a family history of heart attack is the same no matter whether or not the patient lives in a polluted area”*. Let $L = C \setminus G$ be the set of local (not shared) parameters. Let N_g represent the cumulative count corresponding to shared parameter θ_g and N_{lc} be the count corresponding to local parameter θ_{lc} (in distribution $c \in C$).

We now discuss several graphical models which all fit within our framework and satisfy *General Parameter Sharing* assumptions. For instance, in HMMs and Dynamic Bayesian Networks, the same variable has the same conditional probability table at different time instants. Therefore, in this case, C is made out of the distributions in the tables which correspond to a given variable X and the same instantiation of the parents $PA(X) = pa$ across all time instants. In Module Networks, all variables in the same module share the same set of parents and have the same conditional probability tables. Consequently, C consists of the set of distributions corresponding to all variables in a module for a given instantiation of the parents of those variables. Context Specific Independence is used to specify conditional independencies that hold in certain contexts and therefore is useful to specify which distributions should be equal in a conditional probability table for a fixed random variable. In this case, C contains those distributions which are assumed to be equal in a specific table. However, note that our framework allows for much more flexibility in parameter sharing. We can share at the level of each parameter, not just at the whole distribution or table level. Also, the distributions in C do not need to be the same size. Moreover, a shared parameter does not need to be in the same position in different distributions within C.

4.7.1 Maximum Likelihood Estimation from Complete Data

Theorem 4.7.1. *The Maximum Likelihood Estimators for the parameters in C are given by:*

$$\hat{\theta}_g = \frac{N_g}{\sum_{\theta_{g'} \in G} N_{g'} + \sum_{\theta_{lc} \in L} N_{lc}}$$

$$\hat{\theta}_{lc} = \frac{\sum_{\theta_{l'c'} \in L} N_{l'c'}}{\sum_{\theta_g \in G} N_g + \sum_{\theta_{l'c'} \in L} N_{l'c'}} \cdot \frac{N_{lc}}{\sum_{\theta_{l'c} \in L} N_{l'c}}$$

Proof. Our optimization problem can be stated as:

$$P : argmax\ _\theta \{h(\theta) \mid g_c(\theta) = 0, \forall c \in C\}$$

where $g_c(\theta) = (\sum_{\theta_g \in G} \theta_g) + (\sum_{\theta_{lc} \in L} \theta_{lc}) - 1 = 0$

We are searching for the solution of P using Lagrange Multipliers. Introduce Lagrange Multipliers $\lambda = (\lambda_c)_{c \in C}$ for each distribution in C. Let $LM(\theta, \lambda) = h(\theta) - \sum_{c \in C} \lambda_c \cdot g_c(\theta)$. Then the point which maximizes P is among the solutions of the system $\nabla LM(\theta, \lambda) = 0$. It turns out that this system has a unique solution which is in fact the one in the statement of the theorem. According to Sufficiency Criterion 2.1.1, it follows that this solution provides the Maximum Likelihood Estimators for the parameters in the distributions in C. $\square$

Note that the Maximum Likelihood estimators for shared parameters look similar to the ones in the case of standard Bayesian Networks. However, the estimators for local parameters are a product of two factors. First factor represents the probability mass that remains after subtracting the shared parameters. The second factor basically says that this remaining ”local” probability mass in a distribution in C is split into values proportional to the observed counts corresponding to the local parameters in that distribution.

4.7.2 Constrained Dirichlet Priors

We consider Constrained Dirichlet Priors defined by:

$$P(C) = \begin{cases} \frac{1}{Z_C} \cdot \prod_{\theta_g \in G} \theta_g^{\alpha_g - 1} \cdot \prod_{\theta_{lc} \in L, c \in C} \theta_{lc}^{\alpha_{lc} - 1} \\ \quad \text{if} \sum_{\theta_g \in G} \theta_g + \sum_{\theta_{lc} \in L} \theta_{lc} = 1 \ \ \forall c \in C \\ 0 \quad \text{otherwise} \end{cases}$$

It is easy to evaluate the normalization constant. We have:

$$\begin{aligned} Z_C &= \int_{\sum \theta_g + \sum \theta_{lc} = 1 \ \ \forall c \in C} \prod \theta_g^{\alpha_g - 1} \cdot \prod \theta_{lc}^{\alpha_{lc} - 1} \ d\,\theta \\ &= \int_{\sum \theta_g \leq 1} \prod \theta_g^{\alpha_g - 1} \cdot \prod_{c \in C} (\int_{\sum \theta_{lc} = 1 - \sum \theta_g} \prod \theta_{lc}^{\alpha_{lc} - 1} \ d\,\theta_{Lc}) \ d\,\theta_G \end{aligned}$$

where θ_{Lc} represents the set of local parameters in distribution $c \in C$ and θ_G denotes the set of shared (global) parameters. First, we note that the portion of the integral over the local parameters in each distribution $c \in C$ is a *Generalized Dirichlet Integral*, which can be computed using the result in 4.1. Then, it is easy to see that the remaining part of the integral over the shared parameters is a standard Dirichlet Integral. We obtain:

$$Z_C = \prod_g \Gamma(\alpha_g) \cdot \prod_c \frac{\prod \Gamma(\alpha_{lc})}{\Gamma(\sum_{lc} \alpha_{lc})} \cdot \frac{\Gamma(\sum_{lc} \alpha_{lc} - |C| + 1)}{\Gamma(\sum_{lc} \alpha_{lc} + \sum_g \alpha_g - |C| + 1)}$$

There are several interesting properties of this Constrained Dirichlet Prior. First, the joint probability distribution over the shared parameters is a standard Dirichlet. Second, with no parameter sharing, this distribution is a product of independent standard Dirichlet distributions, one for each distribution in C. However, if there are both shared and local parameters, then the joint probability (obtained by marginalization) over a distribution $c \in C$ is not a standard Dirichlet.

4.8 Hierarchical Parameter Sharing Framework

Here we present a hierarchical extension of the framework in the previous section. This will address some of the limitations of the constraints that could be incorporated in the parameter sharing framework described before. In hierarchical parameter sharing, several parameters are shared across a set of distributions, then this set is partitioned and shared parameters are further specified for each subset of distributions. For example, the frequency of ”international words” (for instance ”computer”) may be shared across both Latin languages (Spanish, Italian) and Slavic languages (Russian, Bulgarian). Other Latin words will have the same frequency only across Latin languages and the same holds for Slavic Languages. Finally, other words will be language specific (for example names of country specific objects) and their frequencies will not be shared with any other language.

In order to derive our main results, we need to make some simplifying notations. We will denote by $\theta_1, \ldots, \theta_n$ the distinct parameters involved in the conditional probability distributions on which Hierarchical Parameter Sharing is specified. Let N_{θ_i} represent the cumulative observed count for parameter θ_i (which may appear in multiple places in the Bayesian Network).

We are now ready to describe our Hierarchical Parameter Sharing learning framework. First of all, we present Parameter Sharing Trees as a way to encode hierarchical parameter sharing assumptions and second we show how one can take advantage of such a Parameter Sharing Tree in order to alleviate learning.

4.8.1 Parameter Sharing Trees

Let C represent a set of conditional probability distributions in our Bayesian Network. Each such distribution introduces a constraint on the possible values that the parameters θ can take. A *Parameter Sharing Tree (PST)* is a tree with the following properties:

- Each node v of the tree consists of a pair $(Scope(v),\ Shared(v))$, where $Scope(v)$ is a subset of distributions from C and $Shared(v)$ represents a non-empty set of parameters that are shared across these distributions. A parameter from $Shared(v)$ is a parameter that is known to appear exactly once in each of the distributions in $Scope(v)$, but it is not shared multiple times within one distribution, nor with distributions outside the $Scope$.

- By convention, $Scope(Root) = C$ (this amounts to the fact that we would like to allow for the situation when a parameter is shared by all distributions in C).

- The $Scopes$ of the direct descendants of a node v form a partition of $Scope(v)$. Therefore, the Parameter Sharing Tree will describe a recursive way of partitioning C, with the leaf level being the finest grain of such partition.

- A parameter cannot be shared in multiple places (different nodes of the tree). Because of the recursive partitioning of C, this amounts to the fact that $Shared(v)$ is disjoint with all $Shares$ on the path from v to the root of the tree.

- Each parameter θ in a distribution in C is shared exactly once i.e. there exists exactly one node v such that $\theta \in Shared(v)$. One may argue that there are parameters which are not shared at all, but for all nodes v that have distributions in their $Scope$ such that there remain unshared parameters, one can partition those nodes further in leaves that have only one distribution in their $Scope$, for which previously unshared parameters become shared at the level of that single distribution.

Denote by $Ancestors(v)$ the set of nodes on the path from v to the root of the tree and $Shared(Ancestors(v)) = \bigcup_{v' \in Ancestors(v)} Shared(v')$. Let $Desc(v)$ be the set of descendants of node v (included) in and let $Shared(Desc(v)) = \bigcup_{v' \in Desc(v)} Shared(v')$.

4.8.2 Maximum Likelihood Estimation from Complete Data

Assume we are given a graphical model with known structure that satisfies a set of hierarchical parameter sharing assumptions given by a Parameter Sharing Tree T. In this section we present a theorem that will justify an algorithm for finding the Maximum Likelihood Estimators for the parameters in such a graphical model.

Theorem 4.8.1. *Let v be a node of T and $\theta_i \in Shared(v)$. The following equality holds:*

$$\hat{\theta}_i = \quad (1 - \sum_{\theta_j \in Shared(Ancestors(v))} \hat{\theta}_j) \cdot \frac{N_{\theta_i}}{\sum_{\theta_k \in Shared(Desc(v))} N_{\theta_k}}$$

Proof. By definition,

$$\hat{\theta} = argmax_{\theta} \; \{\sum_{\theta_i} N_{\theta_i} \cdot \log \theta_i \mid \theta \; satisfies \; T\}$$

Each conditional probability distribution $c \in C$ represents a constraint on the space of parameters: $g_c(\theta) = (\sum_{\theta_j \in c} \theta_j) - 1 = 0$. Because of parameter sharing, these constraints can involve some common variables. It is easy to show that, if all the cumulative counts are positive, the likelihood function has a global maximum inside the region determined by the constraints in T. Since the maximum is reached in the interior of the domain, one can apply Lagrange Multipliers to optimize for P_{ik}. Therefore, let us introduce new variables λ_c for each constraint (CPD) $c \in C$. The new function to optimize will be:

$$LM(\theta, \lambda) = \sum_{\theta_i} N_{\theta_i} \cdot \log \theta_i - \sum_{c} \lambda_c \cdot g_c(\theta)$$

According to Lagrange Multipliers theory, any point that is a local maximum or minimum for the initial optimization problem and it is NOT on the border of the region defined by the constraints will be obtained as a (partial) solution of the system of equations:

$$\nabla LM(\theta, \lambda) = 0$$

Therefore $\hat{\theta}$ verifies the above system for some values of λ. Because $\frac{\partial L(\theta|D)}{\partial \theta_i} = \frac{N_{\theta_i}}{\theta_i}$ and $\frac{\partial g_c}{\partial \theta_i} = \lambda_c$ if distribution c contains θ_i (otherwise the partial derivative is zero) , we get:

$$\hat{\theta}_i = \frac{N_{\theta_i}}{\sum_{c \in Scope(v)} \lambda_c} \; \forall \theta_i \in Shared(v) \tag{4.7}$$

Let $S(v) = \sum_{\theta_j \in Shared(Desc(v))} \hat{\theta}_j$. We will prove by induction the following stronger claim:

$$S(v) = \frac{\sum_{\theta_j \in Shared(Desc(v))} N_{\theta_j}}{\sum_{c \in Scope(v)} \lambda_c}$$

and

$$\hat{\theta}_i = \quad (1 - \sum_{\theta_j \in Shared(Ancestors(v))} \hat{\theta}_j) \cdot \frac{N_{\theta_i}}{\sum_{\theta_k \in Shared(Desc(v))} N_{\theta_k}}$$

Base case: If v is a leaf, then the distributions in $Scope(v)$ are equal. The first part of the claim is verified directly from 4.7 and the second part follows because of the fact that the probabilities that add up to $S(v)$ are proportional to their corresponding counts.

Induction Step: Assume v is not a leaf and has direct descendants $d_1, \ldots, d_k$ for which the claim holds. It is obvious that $S(d_1) = \ldots = S(d_k)$. Now, using the induction hypothesis, we obtain $S(d_1) = \ldots = S(d_k) = \frac{\sum_{l=1}^{k} \sum_{\theta_j \in Shared(Desc(d_l))} N_{\theta_j}}{\sum_{c \in Scope(v)} \lambda_c}$. This combined with 4.7 gives us the first part of the claim. The second part of the claim now follows from 4.7 and the fact that

$$\frac{1}{\sum_{c \in Scope(v)} \lambda_c} = \frac{S(v)}{\sum_{\theta_j \in Shared(Desc(v))} N_{\theta_j}}$$

□

The above theorem yields an obvious recursive top-down, breadth-first algorithm to compute the ML Estimates of the parameters. The correctness of the algorithm is justified by theorem 4.8.1 and the fact that a node v is processed sometime after all the nodes on the path from v to the root are processed. The algorithm uses a queue Q to perform breadth-first traversal of the tree.

Algorithm 4.8.1. (Maximum Likelihood Estimators with Hierarchical Parameter Sharing)

STEP 1. *Enqueue the root of the tree in Q.*
STEP 2. *If $Q = \emptyset$, STOP. Else, $v \leftarrow Dequeue(Q)$.*
STEP 3. *Compute $\hat{\theta}_i$ for all $\theta_i \in Shared(v)$.*
STEP 4. *Enqueue all children of v. GO TO STEP 2.*

4.8.3 Constrained Dirichlet Priors

We again choose our priors of the following form:

$$P(\theta) = \frac{1}{Z(T)} \prod \theta_i^{\alpha_i - 1}$$

Note that these priors are defined over the whole space of parameters and that a parameter θ_i can appear in multiple places in the graphical model (according to the given *Parameter Sharing Tree*). In addition, the normalization constant depends heavily on the structure of the parameter sharing tree since T describes the constraints among parameters (the sum of parameters shared on the path from the root to any leaf should sum up to 1).

$Z(T) = \int_{\theta\ obeys\ T} \prod \theta_i^{\alpha_i - 1} d\theta$ can be recursively computed as follows. First, note that this integral can be evaluated starting with the parameters from the leaf level. For each leaf v, the integral over the parameters involved in $Shared(v)$ is a *Generalized Dirichlet Integral*. The effect of computing this integral is to get the constant given by the Standard Dirichlet and propagate upwards a single parameter $S(v)$ with cumulative Dirichlet parameter $(\sum_{\theta_i \in Shared(v)} \alpha_i) - 1$. Now, it is easy to see that this parameter is the same for all leaves that belong to the same parent p. This will make the integral over the parameters in $Shared(p)$ and the new parameter to be also a *Generalized Dirichlet Integral* and the procedure continues as described above until we end the computation at the root level. This concludes or sketch of showing how one can recursively compute $Z(T)$ using *Generalized Dirichlet Integrals*.

4.9 Probability Mass Sharing

Here we show how to perform Maximum Likelihood learning in the case when the aggregate probability mass of a certain parameter type is the same across all distributions in a given set C. For example, we would like to show how to take advantage of constraints like: ”The frequency of nouns in Italian is the same as the frequency of nouns in Spanish”, when modelling the word probability in each of the two languages. In these case, types would be: nouns, verbs, etc.

Before stating the main result of this subsection, let us introduce a few notations. Assume the parameters in C may have the following types: $T_1, \ldots, T_s$. Denote by θ_i^j the i^{th} parameter in j^{th} distribution in C. Let N_i^j represent the observed count for parameter θ_i^j. Each parameter has exactly one type. For example, in the above example, $P(Computer|Italian)$ has type Noun, while $P(Blue|Italian)$ has type Adjective. We would like to stress the fact that in our framework, C is an arbitrary subset of conditional probability distributions in our Bayesian Network. These distributions can have different numbers of parameters and they can belong to different conditional probability tables. Formally, the *Probability Mass Sharing Assumption* states that for all types T_l

and for any j_1^{th} and j_2^{th} distributions in C, we have:

$$\sum_{\theta_i^{j_1} \in T_l} \theta_i^{j_1} = \sum_{\theta_i^{j_2} \in T_l} \theta_i^{j_2}$$

Back to our example, this translates into: ”The aggregate probability of Nouns is the same in all modelled languages and the same holds for other grammatical categories/types.” It might seem a little restrictive to have each parameter belong to one type because, for instance, one may argue that maybe only the probability of Nouns is being shared across languages. However, even if one specifies the Probability Mass Sharing Assumption only for Nouns, the rest of the parameters (non-nouns) must obey the same constraint and therefore that is equivalent to introducing a new dummy type that contains every other parameter in C.

With these considerations we are now ready to compute Maximum Likelihood estimators for the parameters in our model.

4.9.1 Maximum Likelihood Estimation from Complete Data

Theorem 4.9.1. *The maximum likelihood estimator $\hat{\theta}_i^j$ for a parameter θ_i^j in C that has type T_l is given by:*

$$\hat{\theta}_i^j = \frac{N_i^j}{\sum_{\theta_{i'}^j \in T_l} N_{i'}^j} \cdot \frac{\sum_{\theta_{i'}^{j'} \in T_l} N_{i'}^{j'}}{\sum_{\theta_{i'}^{j'}} N_{i'}^{j'}}$$

Proof. We introduce new variables $A = (A_l)_{1 \ldots s}$ that represent the probability mass associated with type T_l in any of the distributions in C. With the newly introduced variables, our optimization problem can be restated as maximizing $f(\theta, A) = \sum N_i^j \cdot \log \theta_i^j$ subject to the constraints that $\sum_{\theta_i^j \in T_l} \theta_i^j = A_l$ for all types T_l and for any j^{th} distribution in C. In addition to these constraints, we also have $\sum_i \theta_i^j = 1$. Similarly to the previous theorems, it is easy to show that, if all counts are positive, then the function f reaches a maximum inside the region on which the constraints and f are defined. In this case, we also apply Lagrange Multipliers theory, introducing a lagrange multiplier for each constraint: λ_l^j for the first type of constraints (probability mass equalities) and λ^j for the second type (distributions should sum up to 1). Therefore, differentiating with respect to θ and A, the point that maximizes f inside the region given by the constraints should also verify:

$$N_i^j = \hat{\theta}_i^j \cdot (\lambda^j + \lambda_l^j) \, \forall \theta_i^j \in T_l \tag{4.8}$$

$$\sum_j \lambda_l^j = 0 \, \forall \, l \tag{4.9}$$

For a fixed j and l, summing up 4.8 for all i such that $\theta_i^j \in T_l$ we get:

$$\sum_{\theta_i^j \in T_l} N_i^j = A_l \cdot (\lambda^j + \lambda_l^j) \tag{4.10}$$

For a fixed l, summing up 4.10 for all j and using 4.9 we obtain:

$$\sum_{\theta_i^j \in T_l} N_i^j = A_l \cdot \sum_j \lambda^j \tag{4.11}$$

Further, summing 4.11 over all values of l and using the fact that the distributions sum up to 1, we can compute:

$$\sum_{i,j} N_i^j = \sum_j \lambda^j \tag{4.12}$$

Now we can use 4.12 in 4.11 to get A_l which is further used in 4.10 to obtain $\lambda^j + \lambda_l^j$ which, substituted in 4.8 will yield the formulas in the statement of the theorem. From Sufficiency Criterion 2.1.1, it follows that $\hat{\theta}$ represents the set of Maximum Likelihood estimators. $\square$

4.10 Probability Ratio Sharing

In the previous section, we showed how to compute parameter estimators when certain parameter types share their aggregate probability mass across different distributions. Now assume we want instead to enforce the constraint that the relative proportions of parameters in a certain type are the same for every distribution within C. This corresponds to statements like: *"In two different countries (A and B), the relative frequency of Heart Attack to Angina Pectoris as the main diagnosis is the same, even though the the aggregate probability of Heart Disease (Heart Attack and Angina Pectoris) may be different because of differences in lifestyle in these countries.*" In this example, the types are *Disease Types*: Heart Disease (Heart Attack, Angina Pectoris), Pulmonary Disease (Pneumonia, Chronic Obstructive Pulmonary Disease, Lung Cancer), etc.

We keep the same notations as in the previous section. However, there are two major differences from the setting presented in the previous section. First, in this case, we must have the same number of parameters of type T_l in each of the distributions in C. For example, we must have the same number of "Heart Diseases" in both countries. This allows us to permute the parameters in the distributions in C such that corresponding parameters in T_l are located on the same position in each of the distributions. For example, we can assume $P(HeartAttack|A)$ and $P(HeartAttack|B)$ are both on the first position in the two diagnosis distributions. This allows us to write that a specific position $i \in T_l$. Second, now there may be parameters that do not belong to any type T_l. For example, the expert may specify that only the ”Heart Diseases” preserve their relative probability ratios across the two countries.

Formally, the *Probability Ratio Sharing Assumption* states that for any fixed type T_l and for fixed $i_1, i_2 \in T_l$, the following holds:

$$\frac{\theta^j_{i_1}}{\theta^j_{i_2}} = constant \,\forall\, j$$

Next we derive Maximum Likelihood estimators that take advantage of the Ratio Assumptions provided by the domain expert.

4.10.1 Maximum Likelihood Estimation from Complete Data

Theorem 4.10.1. *The maximum likelihood estimator $\hat{\theta}^j_i$ for a parameter θ^j_i in C is given by:*

a) if $i \in T_l$: $\hat{\theta}^j_i = \frac{\sum_{j'} N^{j'}_i}{\sum_{i' \in T_l, j'} N^{j'}_{i'}} \cdot \frac{\sum_{i' \in T_l} N^j_{i'}}{\sum_{i'} N^j_{i'}}$

b) if θ^j_i does not have a type: $\hat{\theta}^j_i = \frac{N^j_i}{\sum_{i'} N^j_{i'}}$

Proof. Again, we use Lagrange Multipliers theory to derive our estimators. Parameters in each distribution should sum up to 1 and that translates in the constraint $(\sum_i \theta^j_i) - 1 = 0$. Let the corresponding lagrange multiplier be λ^j. The *Probability Ratio Sharing Assumption* implies that there exist A^j_l and τ_i such that $\theta^j_i - A^j_l \cdot \tau_i = 0$ for all $i \in T_l$. A^j_l represent proportionality constants for distribution j for parameters of type T_l and τ_i are reference constants that, when multiplied with the proportionality constants yield the parameters on position i in each distribution. Let λ^j_i be the lagrange multipliers corresponding to the last type of constraints. Our new objective function becomes $f(\theta, A, \tau) = \sum N^j_i \cdot \log \theta^j_i$. When applying Lagrange Multipliers theory to our optimization problem, differentiating with respect to θ and the newly introduced A^j_l and τ_i, we obtain:

$$N^j_i = \hat{\theta}^j_i \cdot (\lambda^j + \lambda^j_i) \,\forall i \in T_l \tag{4.13}$$

$$N^j_i = \hat{\theta}^j_i \cdot \lambda^j \,\forall i \notin \cup T_l \tag{4.14}$$

$$\sum_j \lambda^j_i \cdot A^j_l = 0 \;or\; \sum_j \lambda^j_i \cdot \hat{\theta}^j_i = 0 \,\forall i \in T_l \tag{4.15}$$

$$\sum_j \lambda_i^j \cdot \tau_i = 0 \; or \sum_{i \in T_l} \lambda_i^j \cdot \hat{\theta}_i^j = 0 \; \forall \, j, l \tag{4.16}$$

For fixed j and l, summing up 4.13 for all $i \in T_l$, then using 4.16 we get:

$$\sum_{i \in T_l} N_i^j = \lambda^j \sum_{i \in T_l} \hat{\theta}_i^j \tag{4.17}$$

If we further sum over all l and use 4.14 we obtain:

$$\lambda^j = \sum_i N_i^j \tag{4.18}$$

Because $\frac{\hat{\theta}_i^{j_1}}{\hat{\theta}_i^{j_2}} = \frac{A_l^{j_1}}{A_l^{j_2}}$ for all j_1,j_2, l and $i \in T_l$, we can write:

$$\frac{A_l^{j_1}}{A_l^{j_2}} = \frac{\sum_{i \in T_l} \hat{\theta}_i^{j_1}}{\sum_{i \in T_l} \hat{\theta}_i^{j_1}} = \frac{\lambda^{j_2}}{\lambda^{j_1}} \cdot \frac{\sum_{i \in T_l} N_i^{j_1}}{\sum_{i \in T_l} N_i^{j_2}} \tag{4.19}$$

For a fixed $i \in T_l$ summing up 4.13 over all j and using 4.15 we have:

$$\sum_j N_i^j = \hat{\theta}_i^j \cdot (\lambda^j + \sum_{j' \neq j} \lambda^{j'} \cdot \frac{\hat{\theta}_i^{j'}}{\hat{\theta}_i^j}) \tag{4.20}$$

Using 4.18 and 4.19 in 4.20 proves part a) of the theorem. Part b) follows from 4.14 and 4.18. The fact that $\hat{\theta}$ are the Maximum Likelihood Estimators follows from Sufficiency Criterion 2.1.2, with an argument similar to the one in the proof of Theorem 4.6.1. $\square$

Chapter 5

Inequality Constraints for Discrete Variables

While in Chapter 4 we have seen how to derive closed form Maximum Likelihood estimators for the parameters in a discrete Bayesian Network when the domain knowledge constraints come in the form of equalities, here we investigate how to perform the same task when Parameter Domain Knowledge is provided as inequality constraints. Unlike in the case of equality constraints, we were not able to compute the normalization constant for the corresponding Constrained Dirichlet Priors in closed form and therefore we will limit our presentation to the derivation of the Maximum Likelihood estimators.

5.1 Inequalities between Sums of Parameters

Briefly, this type of Parameter Domain Knowledge states that the sum of several parameters within one conditional probability distribution is bounded by the sum of other parameters in the same distribution of the Bayesian Network. Intuitively, one can think of this constraint in terms of the parts of speech of a language. Usually, an adverb comes along with a verb and therefore it is reasonable to assume that a language expert can specify that the aggregate probability mass of adverbs is no greater than the aggregate probability mass of the verbs in a given language. Formally, in this type of domain knowledge, the parameters of a conditional probability distribution, denoted by $\theta_1, \ldots, \theta_n$, can be partitioned into $\theta = \cup_{k=1}^{s} A_k \cup_{k=1}^{s} B_k \cup C$ such that $\sum_{\theta_i \in A_k} \theta_i \leq \sum_{\theta_i \in B_k} \theta_i$ for all $1 \leq k \leq s$. Let us denote by N_{A_k} the sum of the observed counts corresponding to parameters in A_k. Similar definitions hold for N_{B_k} and N_C. Let N be the sum of all observed counts N_i corresponding to parameters θ. We have the following theorem:

Theorem 5.1.1. *If all N_i are strictly positive, the Maximum Likelihood Estimators of parameters θ are given by:*

a) $\hat{\theta}_i = \frac{N_i}{N} \cdot \frac{N_{A_k}+N_{B_k}}{2 \cdot N_{A_k}}$ *if* $\theta_i \in A_k$ *and* $N_{A_k} \geq N_{B_k}$
b) $\hat{\theta}_i = \frac{N_i}{N} \cdot \frac{N_{A_k}+N_{B_k}}{2 \cdot N_{B_k}}$ *if* $\theta_i \in B_k$ *and* $N_{A_k} \geq N_{B_k}$
c) $\hat{\theta}_i = \frac{N_i}{N}$ *if* $\theta_i \in A_k \cup B_k$ *and* $N_{A_k} < N_{B_k}$
d) $\hat{\theta}_i = \frac{N_i}{N}$ *if* $\theta_i \in C$

Proof. Finding Maximum Likelihood estimators is equivalent to maximizing $l(\theta) = \sum_i N_i \cdot \log \theta_i$ subject to the domain knowledge constraints, including the constraint that $g(\theta) = \sum_i \theta_i - 1 = 0$. Since this problem contains inequality constraints, we can attempt to solve it using Karush-Kuhn-Tucker theorem. We introduce the Lagrange Multiplier λ for g and μ_k for inequality constraint $h_k(\theta) = \sum_{\theta_i \in A_k} \theta_i - \sum_{\theta_i \in B_k} \theta_i \leq 0$. According to Theorem 2.1.2, we are looking for the optimum $\hat{\theta}$ among the solutions of the system:

$$\begin{cases} \nabla_\theta l(\hat{\theta}) - \lambda \cdot \nabla_\theta g(\hat{\theta}) - \sum_k \mu_k \cdot \nabla_\theta h_k(\hat{\theta}) = 0 \\ g(\hat{\theta}) = 0 \\ \mu_k \cdot h_k(\hat{\theta}) = 0 \\ h_k(\hat{\theta}) \leq 0 \\ \mu_k \geq 0 \end{cases}$$

From the first equation we obtain:

$$\hat{\theta}_i = \begin{cases} \frac{N_i}{\lambda+\mu_k} \text{ if } \theta_i \in A_k \\ \frac{N_i}{\lambda-\mu_k} \text{ if } \theta_i \in B_k \\ \frac{N_i}{\lambda} \text{ if } \theta_i \in C \end{cases}$$

Therefore, $\sum_{\theta_i \in A_k} \hat{\theta}_i = \frac{N_{A_k}}{\lambda+\mu_k}$ and $\sum_{\theta_i \in B_k} \hat{\theta}_i = \frac{N_{B_k}}{\lambda-\mu_k}$. Based on whether constraint k is tight or not we have:

- If $h_k(\hat{\theta}) = 0$, then $\frac{N_{A_k}}{\lambda+\mu_k} = \frac{N_{B_k}}{\lambda-\mu_k}$. This implies $\frac{N_{A_k}}{\lambda+\mu_k} = \frac{N_{B_k}}{\lambda-\mu_k} = \frac{N_{A_k}+N_{B_k}}{2\lambda}$ and therefore $\sum_{\theta_i \in A_k \cup B_k} \hat{\theta}_i = \frac{N_{A_k}+N_{B_k}}{\lambda}$. In this case, we also have $\lambda \cdot (N_{A_k} - N_{B_k}) = \mu_k \cdot (N_{A_k} + N_{B_k})$. Since $\mu_k \geq 0$, we also must have $N_{A_k} \geq N_{B_k}$ in order for constraint k to be tight.

- If $h_k(\hat{\theta}) < 0$, then $\mu_k = 0$ and therefore we again have $\sum_{\theta_i \in A_k \cup B_k} \hat{\theta}_i = \frac{N_{A_k}+N_{B_k}}{\lambda}$. In this case we also have $h_k(\hat{\theta}) = \frac{N_{A_k}-N_{B_k}}{\lambda}$ and since $h_k(\hat{\theta}) < 0$, we must also have $N_{A_k} < N_{B_k}$.

The above observations allow us to conclude that a constraint is tight if and only if $N_{A_k} \geq N_{B_k}$. Now, summing up over all parameters in the conditional probability distribution we get:

$$1 = \sum_i \hat{\theta}_i = \frac{N_C + \sum_k (N_{A_k} + N_{B_k})}{\lambda} = \frac{N}{\lambda}$$

This gives us: $\lambda = N$ and therefore:

$$\hat{\theta}_i = \begin{cases} \frac{N_i}{N+\mu_k} \text{ if } \theta_i \in A_k \\ \frac{N_i}{N-\mu_k} \text{ if } \theta_i \in B_k \\ \frac{N_i}{N} \text{ if } \theta_i \in C \end{cases}$$

Assume now that $N_{A_k} \geq N_{B_k}$. According to the observations above, it means constraint k is tight and we have: $\frac{N_{A_k}}{N+\mu_k} = \frac{N_{B_k}}{N-\mu_k} = \frac{N_{A_k}+N_{B_k}}{2 \cdot N}$. From this we immediately derive: $\hat{\theta}_i = \frac{N_i}{N} \cdot \frac{N_{A_k}+N_{B_k}}{2 \cdot N_{A_k}}$ if $\theta_i \in A_k$ and $N_{A_k} \geq N_{B_k}$ and $\hat{\theta}_i = \frac{N_i}{N} \cdot \frac{N_{A_k}+N_{B_k}}{2 \cdot N_{B_k}}$ if $\theta_i \in B_k$ and $N_{A_k} \geq N_{B_k}$.

If $N_{A_k} < N_{B_k}$, then, as discussed above, μ_k must be 0 and therefore $\hat{\theta}_i = \frac{N_i}{N}$ if $\theta_i \in A_k \cup B_k$ and $N_{A_k} < N_{B_k}$. From Sufficiency Criterion 2.1.1, it follows that $\hat{\theta}$ is the set of Maximum Likelihood estimators. This concludes the proof of our theorem.

$\square$

5.2 Upper Bounds on Sums of Parameters

Here the domain expert provides upper bounds on the sum of several parameters within one conditional probability distribution in the Bayesian Network. Consider the same language example described in the introduction of the previous section. Here the expert may state that the aggregate probability of nouns is no greater than 0.4, the aggregate probability of verbs is no greater than 0.4 and the aggregate probability of adjectives is no greater than 0.3. Even though the combined probability mass of all words equals one, the sum of the upper bounds provided by the expert can be greater than one. Formally, in this type of domain knowledge, the parameters of a conditional probability distribution, denoted by $\theta_1, \ldots, \theta_n$, can be partitioned in $\theta = \cup_{k=1}^{s} A_k$ such that $\sum_{\theta_i \in A_k} \theta_i \leq \alpha_k$ for all $1 \leq k \leq s$, where α_k is a given positive constant. Again, denote by N_{A_k} the sum of the observed counts corresponding to parameters in A_k and by N be the sum of all observed counts N_i corresponding to parameters θ. If there are parameters not involved in any of these constraints, then we can consider they belong to their own set A_k with $\alpha_k = 1$.

In the previous section we found an easy way to decide whether a constraint is tight or not at the optimum point. For the type of constraints that we deal with in this section, we are not able to derive such a simple criterion. However, we show a simple, linear algorithm that computes the set of tight constraints at the optimum point. This algorithm starts with an empty set and at each step adds one of the final tight constraints. Let us start with the theorem describing the Maximum Likelihood estimators:

Theorem 5.2.1. *Assume all observed counts N_i are strictly positive and also assume we know the set $K = \{k_1, \ldots, k_l\}$ of constraints that are tight at the point given by the Maximum Likelihood estimators $\hat{\theta}$. Then, we have:*

a) $\hat{\theta}_i = \alpha_k \cdot \frac{N_i}{N_{A_k}}$ *if* $\theta_i \in A_k$ *and* $k \in K$
b) $\hat{\theta}_i = (1 - \sum_{j \in K} \alpha_j) \cdot \frac{N_i}{\sum_{m \notin K} N_{A_m}}$ *if* $\theta_i \in A_k$ *and* $k \notin K$

Proof. We can approach the problem of finding the Maximum Likelihood estimators in a similar fashion as in Theorem 5.1.1. The data log-likelihood is given by $l(\theta) = \sum_i N_i \cdot \log \theta_i$ which we have to maximize with respect to the domain knowledge constraints, including the constraint that $g(\theta) = \sum_i \theta_i - 1 = 0$. Again, we use Karush-Kuhn-Tucker theorem. We introduce the Lagrange Multiplier λ for g and μ_k for inequality constraint $h_k(\theta) = \sum_{\theta_i \in A_k} \theta_i - \alpha_k \leq 0$. According to Theorem 2.1.2, we are looking for the optimum $\hat{\theta}$ among the solutions of the system:

$$\begin{cases} \nabla_\theta l(\hat{\theta}) - \lambda \cdot \nabla_\theta g(\hat{\theta}) - \sum_k \mu_k \cdot \nabla_\theta h_k(\hat{\theta}) = 0 \\ g(\hat{\theta}) = 0 \\ \mu_k \cdot h_k(\hat{\theta}) = 0 \\ h_k(\hat{\theta}) \leq 0 \\ \mu_k \geq 0 \end{cases}$$

From the first equation we obtain:

$$\hat{\theta}_i = \frac{N_i}{\lambda + \mu_k} \text{ if } \theta_i \in A_k$$

Therefore, $\sum_{\theta_i \in A_k} \hat{\theta}_i = \frac{N_{A_k}}{\lambda + \mu_k}$. Based on whether constraint k is tight or not we have:

- If $h_k(\hat{\theta}) = 0$ i.e. $k \in K$, then $\frac{N_{A_k}}{\lambda + \mu_k} = \alpha_k$. This implies $\hat{\theta}_i = \frac{N_i}{\lambda + \mu_k} = \alpha_k \cdot \frac{N_i}{N_{A_k}}$.

- If $h_k(\hat{\theta}) < 0$ i.e. $k \notin K$, then $\mu_k = 0$ and therefore we have $\sum_{\theta_i \in A_k} \hat{\theta}_i = \frac{N_{A_k}}{\lambda}$.

Summing up over all parameters not involved in the tight constraints, we get:

$$(1 - \sum_{j \in K} \alpha_j) = \sum_{\theta_i \in A_k, k \notin K} \theta_i = \frac{\sum_{j \notin K} N_{A_j}}{\lambda}$$

We obtain $\lambda = \frac{\sum_{m \notin K} N_{A_m}}{1 - \sum_{j \in K} \alpha_j}$ and further: $\hat{\theta}_i = (1 - \sum_{j \in K} \alpha_j) \cdot \frac{N_i}{\sum_{m \notin K} N_{A_m}}$ if $\theta_i \in A_k$ and $k \notin K$. From Sufficiency Criterion 2.1.1, it follows that $\hat{\theta}$ is the set of Maximum Likelihood estimators. This concludes our derivation of the Maximum Likelihood estimators when we know in advance which constraints are satisfied by our estimators. $\square$

Next we describe the algorithm that finds the set K of tight constraints:

Algorithm 5.2.1. (Finding the set of tight constraints if $\sum_j \alpha_j \neq 1$)

STEP 1. *Start with $K = \emptyset$ and at each step add a constraint to K.*

STEP 2. *If $K = \{k_1, \ldots, k_l\}$, let $\lambda_l = \frac{\sum_{m \notin K} N_{A_m}}{1 - \sum_{j \in K} \alpha_j}$ as in the above theorem.*

STEP 3. *If there exists $k_l \notin K$ such that $\frac{N_{A_{k_l}}}{\alpha_{k_l}} \geq \lambda_l$, let $K = K \cup \{k_l\}$ and GO TO Step 2. Otherwise STOP and declare K the set of tight constraints.*

Proof. **(Correctness of Algorithm)** We start by making the following observation, based on the proof of Theorem 5.2.1:

- If h_k tight, then $\frac{N_{A_k}}{\lambda + \mu_k} = \alpha_k$. Because $\mu_k \geq 0$, we must have $\frac{N_{A_k}}{\alpha_k} \geq \lambda$.
- If h_k not tight, then $\mu_k = 0$ and therefore we have $0 > h_k(\hat{\theta}) = \frac{N_{A_k}}{\lambda} - \alpha_k$ and therefore we must have $\frac{N_{A_k}}{\alpha_k} < \lambda$. It is obvious that $\lambda \geq 0$ must hold, otherwise we would have negative parameters.

We have just developed a criterion to test if a set K of constraints is the set of tight constraints:

Lemma 5.2.1. *Given λ computed as in Theorem 5.2.1, K is the set of tight constraints if and only if $\frac{N_{A_k}}{\alpha_k} \geq \lambda$ for all $k \in K$ and $\frac{N_{A_k}}{\alpha_k} < \lambda$ for all $k \notin K$.*

Before proving that our algorithm produces the set of tight constraints, let us prove another useful result:

Lemma 5.2.2. *If $\sum_j \alpha_j \neq 1$ then $N = \lambda_0 \geq \lambda_1 \geq \ldots$, and the quantity $1 - \sum_{j \in K} \alpha_j$ is always strictly positive.*

Proof. **(of lemma)** Since initially $K = \emptyset$, it is obvious that $1 - \sum_{j \in K} \alpha_j \geq 0$. It is also obvious $\lambda_0 = N$. Let us verify the induction step.

From $\frac{N_{A_{k_l}}}{\alpha_{k_l}} \geq \lambda_l$ and because $1 - \sum_{j \in K} \alpha_j > 0$ we get:

$$N_{A_{k_l}} \cdot (1 - \alpha_{k_l} - \sum_{j \in K} \alpha_j) \geq \alpha_{k_l} \cdot \sum_{m \notin K \cup \{k_l\}} N_{A_m} \tag{5.1}$$

It follows $(1 - \sum_{j \in K \cup k_l} \alpha_j) \geq 0$ with equality if and only if we processed all constraints, in which case we have $1 = \sum_j \alpha_j$ and it is obvious that all constraints must be tight. However, since we assumed $\sum_j \alpha_j \neq 1$, we must have $(1 - \sum_{j \in K \cup k_l} \alpha_j) > 0$ and the first part of the induction step is proved.

If in both sides of inequality 5.1 we add the quantity $(1 - \alpha_{k_l} - \sum_{j \in K} \alpha_j) \cdot \sum_{m \notin K \cup \{k_l\}} N_{A_m}$, we obtain:

$$(1 - \alpha_{k_l} - \sum_{j \in K} \alpha_j) \cdot \sum_{m \notin K} N_{A_m} \geq (1 - \sum_{j \in K} \alpha_j) \cdot \sum_{m \notin K \cup \{k_l\}} N_{A_m}$$

which, given that $1 - \alpha_{k_l} - \sum_{j \in K} \alpha_j > 0$, is equivalent to $\lambda_l \geq \lambda_{l+1}$. This concludes the proof of our lemma. $\square$

Applying Lemma 5.2.2, it follows that, in the case when $\sum_j \alpha_j \neq 1$, the Algorithm 5.2.1 ends at a step l such that $\frac{N_{A_{k_j}}}{\alpha_{k_j}} \geq \lambda_j \geq \lambda_l$ for all $k_j \in K$ and $\frac{N_{A_k}}{\alpha_k} < \lambda_l$ for all $k \notin K$. From Lemma 5.2.1 it follows that K is the set of tight constraints in the case when $\sum_j \alpha_j \neq 1$ and therefore Algorithm 5.2.1 is correct. Another case is when all constraints are processed and we are not left with a λ_l to compare with. This situation can not happen, because, at the last step, we would have:

$$\frac{N_{A_{k_s}}}{1 - \sum_{j \neq k_s} \alpha_j} \leq \frac{N_{A_{k_s}}}{\alpha_{k_s}}$$

and therefore either $\sum_j \alpha_j = 1$ or $\sum_j \alpha_j < 1$. In the second case, the constraints are contradictory, which can not happen because we assume the domain expert provides accurate domain knowledge. If $\sum_j \alpha_j = 1$ (case which is not covered by Algorithm 5.2.1), it is obvious that the all constraints must be tight not only for the Maximum Likelihood estimators, but for every feasible value of θ. $\square$

Chapter 6

Equality Constraints for Continuous Variables

In this chapter we illustrate how to efficiently compute Maximum Likelihood estimators that take advantage of Parameter Domain Knowledge in the case of Bayesian Networks with continuous random variables. Unlike in the discrete case, there are many different types of continuous random variables, each parameterized in a different way, so we decided to focus our investigation on the most commonly used type: Gaussian random variables. We study parameter sharing and parameter proportionality for learning in Gaussian Bayesian Networks as well as parameter sharing in Hidden Process Models that involve Gaussian random variables.

6.1 Parameter Sharing in Gaussian Bayesian Networks

Here we look at Gaussian Bayesian Networks, where the conditional probability of each variable is a Gaussian whose mean is a linear combination of the value of the parents. If $PA_i = (Y_1, \ldots, Y_k)$ is the set of parents of variable X_i, we can write $X_i|PA_i \sim N(PA_i{\cdot}\theta_i, \sigma_i^2)$ where $\theta_i = (\theta_{i1}, \ldots, \theta_{ik})^T$ is a column vector of parameters of length k.

In this section we will present Maximum Likelihood estimators for the parameters θ_i, provided that the domain expert is telling us that *”The parameters in coefficient vector θ_i can be partitioned in subsets $T_1, \ldots, T_s$ such that all parameters in any given subset are equal”*. If a parameter is not shared, then its corresponding set T_j has only one element. Intuitively, one can think of the coefficients θ_{ip} as the magnitude of the contribution of parent Y_p to the value of X_i. Consequently, the above parameter sharing assumptions specify that the contribution of several of its parents is same as important to the value of X_i. For example, one can predict the stock of computer maker *DELL* as a weighted sum of the stocks of software maker *Microsoft (MSFT)* and chip maker *Intel (INTL)*.

Parameter sharing corresponds to the statement that *MSFT* and *INTL* have the same importance (weight) for predicting the value of stock *DELL*.

Let θ'_{ij} be the common value of parameters in T_j. Therefore, the new vector of parameters to estimate will be $\theta'_i = (\theta'_{i1}, \ldots, \theta'_{is})^T$. Similar constraints can be independently specified on variables other than X_i. Let us introduce a new variable Z_j to denote the sum of parents of X_i corresponding to parameters in subset T_j i.e. $Z_j = \sum_{\theta_{ip} \in T_j} Y_p$. We call $PA'_i = (Z_1, \ldots, Z_s)$ the vector of *Abstracted Parents* of X_i. If we denote by X_i^l the value of X_i and by PA_i^l the value of PA'_i in training example d_l, we can define A_i to be the *Abstracted Parents Matrix* and b_i to be the *Variable Vector* for variable X_i such that each has a row corresponding to each example in the training set:

$$A_i = \begin{pmatrix} PA_i'^1 \\ PA_i'^2 \\ \ldots \\ PA_i'^m \end{pmatrix} \text{ and } b_i = \begin{pmatrix} X_i^1 \\ X_i^2 \\ \ldots \\ X_i^m \end{pmatrix}$$

With these notations, the Maximum Likelihood estimators are given by the following theorem:

Theorem 6.1.1. *If $A_i^T \cdot A_i$ is non-singular, the Maximum Likelihood Estimators of the parameters for Gaussian variable X_i are given by:*

$$\hat{\theta}_i' = (A_i^T \cdot A_i)^{-1} \cdot A_i^T \cdot b_i$$
$$\hat{\sigma}_i^2 = \frac{||A_i \cdot \hat{\theta}_i - b_i||^2}{m}$$

Proof. The decomposability of log-likelihood, allows us to compute Maximum Likelihood estimators by solving a set of independent optimization problems, one for each conditional probability distribution in the Gaussian Bayesian Network. Because $X_i | PA_i \sim N(PA_i \cdot \theta_i, \sigma_i^2)$, based on the parameter sharing assumptions provided by the expert, we get: $X_i | PA_i \sim N(PA'_i \cdot \theta'_i, \sigma_i^2)$. Combined with the independence of the optimization problems corresponding to different variables in the network, this allows us to substitute the parents of X_i with its *Abstracted Parents* and therefore the result follows directly from Theorem 2.2.4. $\square$

6.2 Parameter Proportionality in Gaussian Bayesian Networks

Parameter Proportionality in Gaussian Bayesian Networks is a straightforward extension of Parameter Sharing. In this type of domain knowledge, parameters are not necessarily equal to each other, but they are proportional to some constants given by the domain expert. In other words, the expert knows the relative strengths of the contributions of several parents to the value of a variable X_i in the network. Let us extend the example in the previous section. Here we want to predict the stock of computer maker *DELL* as a weighted sum of the stocks *MSFT*, *INTL* and the stock of a *Power Supply Maker (PSUPPLY)*. While the expert is stating that *MSFT* and *INTL* have the same importance (weight) for predicting the value of stock *DELL*, he also states that *PSUPPLY* has 3 times less importance (weight).

The Parameter Proportionality constraints supplied by the expert will look like: *"The parameters in coefficient vector θ_i can be partitioned in subsets $T_1, \ldots, T_s$ such that the parameters in T_j are proportional to some known constants"*. If $T_j = \{\theta_{ij_1}, \ldots, \theta_{ij_t}\}$ then there exists an unknown value θ'_{ij} such that $T_j = \theta'_{ij} \cdot \{c_{j_1}, \ldots, c_{j_t}\}$. The set of parameters to estimate becomes $\theta'_i = (\theta'_{i1}, \ldots, \theta'_{is})^T$. Parameter Sharing presented in the previous section can be seen as a particular case when all constants c are equal to one.

In this case, let $Z_1, \ldots, Z_s$ be the *Abstracted Parents* defined as $Z_j = \sum_{\theta_{ij_p} \in T_j} c_{j_p} \cdot Y_p$. It is now easy to see that the Maximum Likelihood estimators in the presence of Parameter Proportionality constraints within one conditional probability distribution can be found by using Theorem 6.1.1 based on our new *Abstracted Parents*.

6.3 Parameter Sharing in Hidden Process Models

A *Hidden Process Model (HPM)* is a probabilistic framework that predicts the value of a *target variable X* at a given point in time as the sum of the values of certain *Hidden Processes* that are active. This model is inspired from observations of the fMRI signal in the brain when a subject performs a cognitive task. One can think of the target variable as the value of the fMRI signal in one small cube inside the brain (also called a voxel). A hidden process may be thought of as the fMRI activity that happens as a response to an external *stimulus*. For example, a *"Picture"* process may describe the fMRI signal that happens in the brain starting when the subject is presented with a picture. A *"Sentence"* process may provide the same characterization for the situation when a subject is reading a sentence. In the real world several stimuli may be active at some point in time and it is conjectured that the observed fMRI signal is the sum of the corresponding processes, translated according to their starting times.

Formally, a *Hidden Process Model* is a collection of time series (also called hidden processes): $P_1, \ldots, P_K$. For each process P_k with $1 \leq k \leq K$, denote by P_{kt} the value of its corresponding time series at time t after the process started. Also, let X_t be the value of the target variable X at time t. If process P_k starts at time t_k, then a Hidden Process Model is predicting the X_t using the following distribution:

$$X_t \sim N(\sum_k P_{k(t-t_k+1)}, \sigma^2)$$

where σ^2 is considered to be the variance in the measurement and it is kept constant across time. For the above formula to make sense, we consider $P_{kt} = 0$ if $t < 0$. While in the real world it may happen that the subject is presented with the same kind of stimulus multiple times across time, here we make the assumption that each process is active at most once within each example. Figure 6.1 shows an example of a Hidden Process Model for the fMRI activity in a voxel in the brain during a cognitive task involving reading a sentence and looking at a picture.

In an fMRI experiment, the subject may perform the same cognitive task multiple times and that leaves us with multiple observations about X_t, the value of X at time t. In our framework we denote by X_{nt} the value of X_t and by t_{nk} the starting point of process P_k in example n in the dataset of observations about X, where each observation tracks the value of X across time given a combination of processes that can start at any times. Let N be the total number of observations. Now we can write:

$$X_{nt} \sim N(\sum_k P_{k(t-t_{nk}+1)}, \sigma^2)$$

While not entirely necessary for our method to work, several assumptions will allow us to make a more compact presentation of Hidden Process Models, based on some characteristics of the fMRI dataset that we are going to use in Chapter 8. First, we assume that X is tracked across the same time length in each observation. Let T be the length of every such observation (trial). Since we are not modelling what happens when $t > T$, we can also consider that each process has length T. Second, in our fMRI dataset, we know in advance when the stimuli are presented and therefore in our model we assume that t_{nk}, the starting times of the processes, are fully observable.

The same stimuli may influence the activity in multiple voxels of the brain during one cognitive task. For example, looking at a picture may activate many voxels in the visual cortex. The activation in these voxels may be different at each given point in time. Intuitively, that means the same stimulus may produce different hidden processes in different voxels. However, certain groups of voxels that are close together often have similar shape time series, but with different amplitude. In this case, we believe it is reasonable to assume that the underlying hidden processes corresponding to these voxels are proportional to each other. Experiments performed in Chapter 8 will prove that this assumption will help learn better models than the ones that choose to ignore it.

In the above paragraph we explained intuitively that sometimes it makes sense to share the same base processes across several time-varying random variables, but allow for different scaling factors. Formally, we say that time-varying random variables $X^1, \ldots, X^V$ share their corresponding *Hidden Process Models* if there exist base processes $P_1, \ldots, P_K$ and constants c_k^v for $1 \leq v \leq V$ such that:

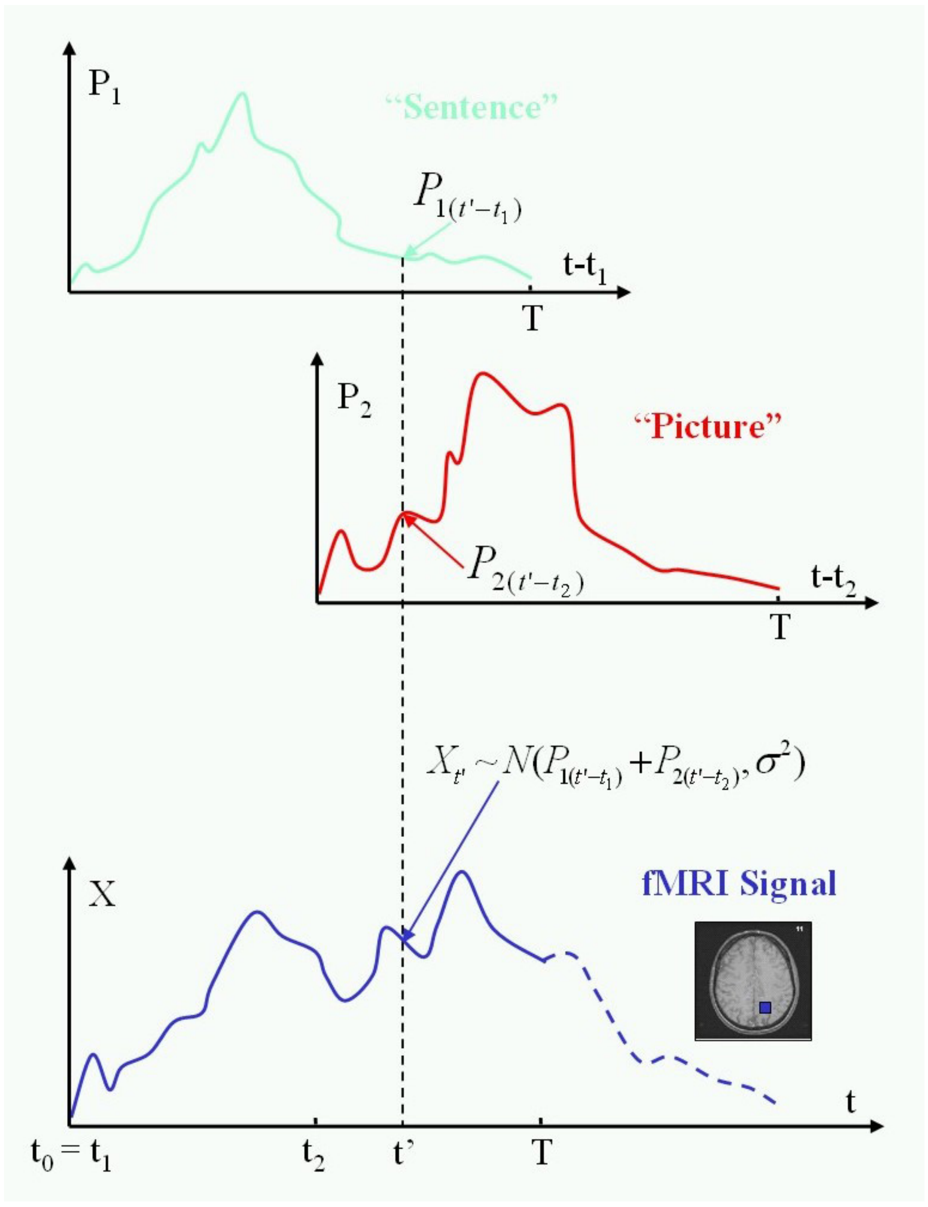

Figure 6.1: A Hidden Process Model for the cognitive task when a subject is asked to read a sentence and to look at a picture. In half of the observations, the sentence is presented first, then the picture is shown. In the other half of the observations, the picture is presented first. The activity in a given voxel X in the brain is modelled as a Hidden Process Model with two processes: "Sentence" (P_1) and "Picture" (P_2). Each observation has length $T = 32$ fMRI snapshots (16 seconds) and the same holds for both processes. This figure shows an observation where the sentence is presented at time $t_1 = 1$ and the picture is shown at $t_2 = 17$ (8 seconds after t_1). After time t_2, the two processes overlap and the fMRI signal $X_{t'}$ is the sum of the corresponding values of the two processes plus $N(0, \sigma^2)$ measurement variance. The blue dotted line represents the fMRI activity that would happen after time T.

$$X_{nt}^v \sim N(\sum_k c_k^v \cdot P_{k(t-t_{nk}^v+1)}, \sigma^2)$$

and the values of different variables X^v are independent given the parameters of the model. Here σ^2 represents the variance in measurement which is also shared across these variables.

Next we will study how to efficiently perform Maximum Likelihood estimation of the parameters of the variables $X^1, \ldots, X^V$, assuming that they share their corresponding Hidden Process Model parameters as described above. We make the additional assumption that we know the starting times and identities of the hidden processes (HPMs have been studied under this assumption in [Dal99]). The parameters to estimate are the base process parameters P_{kt} where $1 \leq k \leq K$ and $1 \leq t \leq T$, the scaling constants c_k^v (one for each variable V and process k) where $1 \leq v \leq V$ and the common measurement variance σ^2. Let $P = \{P_{kt} \mid 1 \leq k \leq K,\ 1 \leq t \leq T\}$ be the set of all parameters involved in the base processes and let $C = \{c_k^v \mid 1 \leq k \leq K,\ 1 \leq v \leq V\}$ be the set of scaling constants. We remind the reader that N represents the number of observations. The log-likelihood of the model is given by:

$$l(P, C, \sigma) \;=\; -\frac{NTV}{2} \cdot \log(2\pi) - NTV \cdot \log(\sigma) - \frac{1}{2 \cdot \sigma^2} \cdot \sum_{n,t,v} (x_{nt}^v - \sum_k c_k^v \cdot P_{k(t-t_{nk}^v+1)})^2$$

It is easy to see that the value of (P, C) that maximizes l is the same for all values of σ. Therefore, in order to maximize l, we can first minimize $l'(P, C) = \sum_{n,t,v}(x_{nt}^v - \sum_k c_k^v \cdot P_{k(t-t_{nk}^v+1)})^2$ with respect to (P, C) and then maximize l with respect to σ based on the minimum point for l'. One may notice that l' is a sum of squares, where the quantity inside each square can be seen as a linear function in both P and C. Therefore one can imagine an iterative procedure that first minimizes with respect to P, then with respect to C using the Least Squares method. Once we find $M = \min l'(P, C) = l'(\hat{P}, \hat{C})$, the value of σ that maximizes l is given by $\hat{\sigma}^2 = \frac{M}{NVT}$. This can be derived in a straightforward fashion by enforcing $\frac{\partial l}{\partial \sigma}(\hat{P}, \hat{C}, \hat{\sigma}) = 0$. With these considerations, we are now ready to present an algorithm to compute Maximum Likelihood estimators $(\hat{P}, \hat{C}, \hat{\sigma})$ of the parameters in the shared Hidden Process Model:

Algorithm 6.3.1. (Maximum Likelihood Estimators in a Shared Hidden Process Model)
Let $\bar{X}$ be the column vector of values x_{nt}^v. Start with a random guess $(\hat{P}, \hat{C})$ and then repeat Steps 1 and 2 until they converge to the minimum of the function $l'(P, C)$.

STEP 1. *Write $l(\hat{P}, \hat{C}) = ||A \cdot \hat{P} - \bar{X}||^2$ where A is a NTV by KT matrix that depends on current estimator $\hat{C}$ of the scaling constants. Minimize with respect to $\hat{P}$ using ordinary Least Squares to get a new estimator $\hat{P} = (A^T \cdot A)^{-1} \cdot A^T \cdot \bar{X}$.*

STEP 2.*Write $l(\hat{P}, \hat{C}) = ||B \cdot \hat{C} - \bar{X}||^2$ where B is a NTV by KV matrix that depends on current estimator $\hat{P}$ of the base processes. Minimize with respect to $\hat{C}$ using ordinary Least Squares to get a new estimator $\hat{C} = (B^T \cdot B)^{-1} \cdot B^T \cdot \bar{X}$.*

STEP 3. *Once convergence is reached by repeating the above two steps, let $\hat{\sigma}^2 = \frac{l'(\hat{P},\hat{C})}{NVT}$.*

It might seem that this is a very expensive algorithm because it is an iterative method. However, we applied it in our experiments of modelling the fMRI signal during a cognitive task and it turns out it usually converges in 3-5 repetitions of Steps 1 and 2. We believe that the main reason why this happens is because at each partial step during the iteration we compute a closed form global minimizer on either $\hat{P}$ or $\hat{C}$ instead of using a potentially expensive gradient descent algorithm. In Chapter 8 we will experimentally prove the benefits of this algorithm over methods that do not take advantage of parameter sharing assumptions, i.e. the shared Hidden Process Models corresponding to neighboring voxels in the brain when the subject is performing a cognitive task.

Chapter 7

Formal Guarantees

In the Introduction, we motivated that taking advantage of Parameter Domain Knowledge can be beneficial to learning because, intuitively, it has the effect of lowering the variance in parameter estimators by shrinking the degrees of freedom of the model. In this chapter we provide a formal proof of this fact on Parameter Sharing within One Distribution type of domain knowledge, which was introduced in section 4.3. In order for our proof to work, we make the assumption that the true distribution factorizes according to the given Bayesian Network structure and that it obeys the parameter sharing assumptions. The second interesting result presented in this chapter will give theoretical guarantees in the case when the Parameter Domain Knowledge provided by the expert might not be entirely accurate. We will prove this result on another type of domain knowledge: Parameter Sharing across Multiple Distributions, first introduced in section 4.7. While we only investigate two different types of Parameter Domain Knowledge, we strongly believe that the same kind of formal guarantees describe all other types of domain knowledge presented in this thesis.

7.1 Variance Reduction by Using Parameter Domain Knowledge

Assume we want to learn a Bayesian Network in the case when a domain expert provides Parameter Domain Knowledge constraints specifying that certain parameters appear multiple times (are shared) within a conditional probability distribution (see section 4.3). Each conditional probability distribution in the Bayesian Network can have its own such constraints. Also, the case when all parameters are distinct within one such distribution may be seen as a particular case of Parameter Sharing within One Distribution, where each parameter is shared exactly once.

We have two ways to perform Maximum Likelihood parameter learning in the above Bayesian Network. First, we may choose to ignore the domain knowledge given by the expert and use Theorem 2.2.1 to estimate parameters. A second option is to incorporate the domain knowledge in the learning method, in which case we can use the results described in Section 4.3. One would intuitively expect that taking advantage of the domain knowledge provided by the expert would reduce the variance in parameter estimates when compared to the first approach.

Let us first derive the variance in standard Maximum Likelihood parameter estimators in a Bayesian Network, ignoring any additional Parameter Sharing assumptions:

Lemma 7.1.1. *Let θ be the true parameters of a Bayesian Network describing a distribution P that factorizes according to that network. Let $\hat{\theta}$ be the maximum likelihood estimators of these parameters as given by Theorem 2.2.1. Then the variance of $\hat{\theta}_{ijk}$ is given by:*

$$Var[\hat{\theta}_{ijk}] = \theta_{ijk} \cdot (1 - \theta_{ijk}) \cdot E[\frac{1}{N_{ik}} | N_{ik} \neq 0]$$

Proof. According to Theorem 2.2.1, $\hat{\theta}_{ijk} = \frac{N_{ijk}}{N_{ik}}$ and it is well defined conditioned on the fact that $N_{ik} \neq 0$. To compute the variance of a random variable X we use: $Var(X) = E[X^2] - E^2[X]$. Given that $\theta_{ijk} = P(X_i = x_{ij} | PA_i = pa_{ik})$, it follows that the number of times that we are going to observe $X_i = x_{ij}$ in b independent trials when $PA_i = pa_{ik}$ is a binomial random variable $P(N_{ijk} | N_{ik} = b) \sim \ Binomial(b, \theta_{ijk})$. Using this observation, let us first compute the expected value of $\hat{\theta}_{ijk}$:

$$\begin{aligned}
E[\hat{\theta}_{ijk}] &= E[\frac{N_{ijk}}{N_{ik}} | N_{ik} \neq 0] \\
&= \sum_{b \geq 1, 0 \leq a \leq b} \frac{a}{b} \cdot P(N_{ijk} = a, N_{ik} = b | N_{ik} \neq 0) \\
&= \sum_{b \geq 1, 0 \leq a \leq b} \frac{a}{b} \cdot P(N_{ijk} = a | N_{ik} = b) \cdot P(N_{ik} = b | N_{ik} \neq 0) \\
&= \sum_{b \geq 1, 0 \leq a \leq b} \frac{a}{b} \cdot \binom{b}{a} \theta_{ijk}^{a} \cdot (1 - \theta_{ijk})^{b-a} \cdot P(N_{ik} = b | N_{ik} \neq 0) \\
&= \sum_{b \geq 1} P(N_{ik} = b | N_{ik} \neq 0) \cdot \sum_{0 \leq a \leq b} \frac{a}{b} \cdot \binom{b}{a} \theta_{ijk}^{a} \cdot (1 - \theta_{ijk})^{b-a} \\
&= \sum_{b \geq 1} P(N_{ik} = b | N_{ik} \neq 0) \cdot \sum_{1 \leq a \leq b} \binom{b-1}{a-1} \theta_{ijk}^{a} \cdot (1 - \theta_{ijk})^{b-a} \\
&= \sum_{b \geq 1} P(N_{ik} = b | N_{ik} \neq 0) \cdot \theta_{ijk} \cdot \sum_{1 \leq a \leq b} \binom{b-1}{a-1} \theta_{ijk}^{a-1} \cdot (1 - \theta_{ijk})^{(b-1)-(a-1)} \\
&= \theta_{ijk} \cdot \sum_{b \geq 1} P(N_{ik} = b | N_{ik} \neq 0) \cdot (\theta_{ijk} + (1 - \theta_{ijk}))^{b-1} \\
&= \theta_{ijk} \cdot \sum_{b \geq 1} P(N_{ik} = b | N_{ik} \neq 0) \\
&= \theta_{ijk}
\end{aligned}$$

The formula for $E[\hat{\theta}^2_{ijk}]$ can be computed in a similar fashion:

$$\begin{aligned}
E[\hat{\theta}^2_{ijk}] &= E[\frac{N^2_{ijk}}{N^2_{ik}}|N_{ik} \neq 0] \\
&= \sum_{b\geq 1, 0\leq a\leq b} \frac{a^2}{b^2} \cdot P(N_{ijk} = a, N_{ik} = b|N_{ik} \neq 0) \\
&= \sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0) \cdot \sum_{0\leq a\leq b} \frac{a^2}{b^2} \cdot \binom{b}{a} \theta^a_{ijk} \cdot (1-\theta_{ijk})^{b-a} \\
&= \sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0) \cdot \frac{1}{b} \cdot \sum_{1\leq a\leq b} a \cdot \binom{b-1}{a-1} \theta^a_{ijk} \cdot (1-\theta_{ijk})^{b-a} \\
&= \sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0) \cdot \frac{\theta_{ijk}}{b} \cdot (\sum_{1\leq a\leq b} \binom{b-1}{a-1} \theta^{a-1}_{ijk} \cdot (1-\theta_{ijk})^{(b-1)-(a-1)} + \\
&\quad +(b-1) \cdot \theta_{ijk} \cdot \sum_{2\leq a\leq b} \binom{b-2}{a-2} \theta^{a-2}_{ijk} \cdot (1-\theta_{ijk})^{(b-2)-(a-2)}) \\
&= \sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0) \cdot \frac{\theta_{ijk}}{b} \cdot (1 + (b-1) \cdot \theta_{ijk}) \\
&= \theta_{ijk} \cdot \sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0) \cdot (\frac{1}{b} + \theta_{ijk} - \frac{\theta_{ijk}}{b}) \\
&= \theta^2_{ijk} + \theta_{ijk} \cdot (1-\theta_{ijk}) \cdot E[\frac{1}{N_{ik}}|N_{ik} \neq 0]
\end{aligned}$$

Therefore, we have $Var[\hat{\theta}_{ijk}] = E[\hat{\theta}^2_{ijk}] - E^2[\theta_{ijk}] = \theta_{ijk} \cdot (1-\theta_{ijk}) \cdot E[\frac{1}{N_{ik}}|N_{ik} \neq 0]$ □

We have just seen how to compute the variance in standard Maximum Likelihood estimators in a Bayesian Network. The following lemma gives us a way of calculating the variance in Maximum Likelihood parameter estimators that take advantage of parameter sharing constraints that hold within one conditional probability distribution:

Lemma 7.1.2. *Assume a domain expert is specifying constraints of the form:* **"within distribution $X_i|PA_i = pa_{ik}$, parameters $\theta_{ij_1k}, \ldots, \theta_{ij_sk}$ are shared (have equal value)"**. *If we compute $\hat{\theta}$, the Maximum Likelihood estimators that take advantage of this domain knowledge via Theorem 4.3.1, then the variance in $\hat{\theta}$ is given by:*

$$Var[\hat{\theta}_{ij_tk}] = \theta_{ij_tk} \cdot (\frac{1}{s} - \theta_{ij_tk}) \cdot E[\frac{1}{N_{ik}}|N_{ik} \neq 0] \ \ \forall\, 1 \leq t \leq s$$

Proof. When the expert reveals us that true probability distribution described by the Bayesian Networks satisfies $\theta_{ij_1k} = \ldots = \theta_{ij_sk}$, we denote by $N_{ij_{1..s}k}$ the sum of observed counts corresponding to these parameters i.e. $N_{ij_{1..s}k} = \sum_{1\leq t\leq s} N_{ij_tk}$. In other words, $N_{ij_{1..s}k}$ is the number of times $X_i \in \{x_{ij_1}, \ldots, x_{ij_s}\}$ when $PA_i = pa_{ik}$. Theorem 4.3.1 yields the following Maximum Likelihood estimator: $\hat{\theta}_{ij_tk} = \frac{N_{ij_{1..s}k}}{s\cdot N_{ik}}$. Given that $P(X_i \in \{x_{ij_1}, \ldots, x_{ij_s}\}|PA_i = pa_{ik}) = s\theta_{ij_tk}$, we conclude that, in b independent trials in which $PA_i = pa_{ik}$ is observed, $N_{ij_{1..s}k}|N_{ik} = b$ is a binomial variable distributed $P(N_{ij_{1..s}k}|N_{ik} = b) \sim Binomial(b, s\theta_{ij_tk})$. With this observation, let us first compute the expected value of θ_{ij_tk}:

$$\begin{aligned}
E[\hat{\theta}_{ij_tk}] &= E[\frac{N_{ij_{1..s}k}}{s\cdot N_{ik}}|N_{ik} \neq 0] \\
&= \sum_{b\geq 1, 0\leq a\leq b} \frac{a}{s\cdot b}\cdot P(N_{ij_{1..s}k} = a, N_{ik} = b|N_{ik} \neq 0) \\
&= \sum_{b\geq 1, 0\leq a\leq b} \frac{a}{s\cdot b}\cdot P(N_{ij_{1..s}k} = a|N_{ik} = b)\cdot P(N_{ik} = b|N_{ik} \neq 0) \\
&= \sum_{b\geq 1, 0\leq a\leq b} \frac{a}{s\cdot b}\cdot \binom{b}{a}(s\theta_{ij_tk})^a\cdot(1 - s\theta_{ij_tk})^{b-a}\cdot P(N_{ik} = b|N_{ik} \neq 0) \\
&= \frac{1}{s}\cdot\sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0)\cdot\sum_{0\leq a\leq b}\frac{a}{b}\cdot\binom{b}{a}(s\theta_{ij_tk})^a\cdot(1 - s\theta_{ij_tk})^{b-a} \\
&= \frac{1}{s}\cdot\sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0)\cdot\sum_{1\leq a\leq b}\binom{b-1}{a-1}(s\theta_{ij_tk})^a\cdot(1 - s\theta_{ij_tk})^{b-a} \\
&= \frac{1}{s}\cdot\sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0)\cdot s\cdot\theta_{ij_tk}\cdot(s\theta_{ijk} + (1 - s\theta_{ijk}))^{b-1} \\
&= \theta_{ij_tk}\cdot\sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0) \\
&= \theta_{ij_tk}
\end{aligned}$$

Using the same approach, we can compute the formula for $E[\hat{\theta}^2_{ijk}]$:

$$\begin{aligned}
E[\hat{\theta}^2_{ij_tk}] &= E[\frac{N^2_{ij_{1..s}k}}{s^2\cdot N^2_{ik}}|N_{ik} \neq 0] \\
&= \sum_{b\geq 1, 0\leq a\leq b}\frac{a^2}{s^2\cdot b^2}\cdot P(N_{ij_{1..s}k} = a, N_{ik} = b|N_{ik} \neq 0) \\
&= \frac{1}{s^2}\cdot\sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0)\cdot\sum_{0\leq a\leq b}\frac{a^2}{b^2}\cdot\binom{b}{a}(s\theta_{ij_tk})^a\cdot(1 - s\theta_{ij_tk})^{b-a} \\
&= \frac{1}{s^2}\cdot\sum_{b\geq 1} P(N_{ik} = b|N_{ik} \neq 0)\cdot\frac{1}{b}\cdot\sum_{1\leq a\leq b} a\cdot\binom{b-1}{a-1}(s\theta_{ij_tk})^a\cdot(1 - s\theta_{ij_tk})^{b-a}
\end{aligned}$$

$$
\begin{aligned}
&= \frac{1}{s^2} \cdot \sum_{b \geq 1} P(N_{ik} = b | N_{ik} \neq 0) \cdot \frac{s\theta_{ij_tk}}{b} \cdot (1 + (b-1) \cdot s\theta_{ij_tk}) \\
&= \frac{\theta_{ijk}}{s} \cdot \sum_{b \geq 1} P(N_{ik} = b | N_{ik} \neq 0) \cdot (\frac{1}{b} + s\theta_{ij_tk} - \frac{s\theta_{ij_tk}}{b}) \\
&= \theta_{ij_tk}^2 + \theta_{ij_tk} \cdot (\frac{1}{s} - \theta_{ij_tk}) \cdot E[\frac{1}{N_{ik}} | N_{ik} \neq 0]
\end{aligned}
$$

Therefore, we have $Var[\hat{\theta}_{ij_tk}] = E[\hat{\theta}_{ij_tk}^2] - E^2[\theta_{ij_tk}] = \theta_{ij_tk} \cdot (\frac{1}{s} - \theta_{ij_tk}) \cdot E[\frac{1}{N_{ik}} | N_{ik} \neq 0]$ $\square$

Combining the above two lemmas, we obtain the following theorem:

Theorem 7.1.1. *Assuming a domain expert can specify parameter sharing assumptions that take place inside the conditional probability distributions of a Bayesian Network, the Maximum Likelihood estimators that use this domain knowledge as computed with Theorem 4.3.1 have lower variance than standard Maximum Likelihood estimators computed with Theorem 2.2.1, ignoring the domain knowledge. More specifically, for one parameter θ_{ijk} that is shared $s \geq 1$ times within $P(X_i | PA_i = pa_{ik})$, denote by $\hat{\theta}_{ijk}^{ML}$ the Maximum Likelihood estimator that ignores domain knowledge and by $\hat{\theta}_{ijk}^{PS}$ the Maximum Likelihood estimator that uses the parameter sharing assumptions specified by the expert. We have the following identity:*

$$Var[\hat{\theta}_{ijk}^{ML}] - Var[\hat{\theta}_{ijk}^{PS}] = \theta_{ijk} \cdot (1 - \frac{1}{s}) \cdot E[\frac{1}{N_{ik}} | N_{ik} \neq 0] \geq 0$$

Proof. From Lemma 7.1.1 and Lemma 7.1.2 we obtain:

$$Var[\hat{\theta}_{ijk}^{ML}] = \theta_{ijk} \cdot (1 - \theta_{ijk}) \cdot E[\frac{1}{N_{ik}} | N_{ik} \neq 0]$$

and

$$Var[\hat{\theta}_{ijk}^{PS}] = \theta_{ijk} \cdot (\frac{1}{s} - \theta_{ijk}) \cdot E[\frac{1}{N_{ik}} | N_{ik} \neq 0]$$

The difference in variance is then given by:

$$Var[\hat{\theta}_{ijk}^{ML}] - Var[\hat{\theta}_{ijk}^{PS}] = \theta_{ijk} \cdot (1 - \frac{1}{s}) \cdot E[\frac{1}{N_{ik}} | N_{ik} \neq 0] \geq 0$$

with equality when $s = 1$ i.e. θ_{ijk} is not shared multiple times. $\square$

7.2 Performance with Potentially Inaccurate Domain Knowledge

Sometimes it may happen that the Parameter Domain Knowledge provided by an expert is not completely accurate. In all our methods so far, we assumed that the domain knowledge is correct and therefore errors in domain knowledge can prove detrimental to the performance of our learned models. In this section we investigate the relationship between the true, underlying distribution of the observed data and the distribution estimated using our methods based on Parameter Domain Knowledge. In particular, we come up with an upper bound on how well our estimated model can perform given a set of potentially incorrect Parameter Domain Knowledge constraints. While in the previous section we illustrated the main result via Parameter Sharing within One Distribution type of domain knowledge, here we will prove our results for the general Parameter Sharing framework described in section 4.7. In this type of domain knowledge, a parameter can be either shared once in every conditional probability distribution in a given set or it is local within its corresponding probability distribution. Shared parameters are also called global parameters.

In order to present these results, we will follow the notations in section 4.7. Assume an expert specified a set of Parameter Sharing assumptions across a set of conditional probability distributions in our Bayesian Network as described above. Let us introduce the notion of *True Probabilistic Counts(TPC)*. Suppose P is the true distribution from which data is sampled. If θ_{lck} is the local parameter of the graphical model that is supposed to describe $P(X = x|PA(X) = pa)$, then let $TPC_{lck} = P(X = x, Pa(X) = pa)$. If θ_{gk} is the global parameter of the graphical model that is supposed to describe the set of *allegedly* equal parameters $\{P(X_1 = x_1|PA(X_1) = pa_1), \ldots, P(X_s = x_s|PA(X_s) = pa_s)\}$, let $TPC_{gk} = \sum_{i=1}^{s} P(X_i = x_i, PA(X_i) = pa_i)$. Let P^* be the distribution that factorizes according to the structure provided by the expert and has parameters given by theorem 4.7.1 where the observed counts are replaced by the *True Probabilistic Counts*.

Theorem 7.2.1. *P^* is the closest distribution to P(in terms of $KL(P, \cdot)$) that factorizes according to the given structure and obeys the expert's parameter sharing assumptions.*

Proof. Let Q be such a distribution. Minimizing $K(P, Q)$ is equivalent to maximizing $\sum_d P(d) \cdot \log Q(d)$. Let θ be the set of parameters that describe this distribution Q. After breaking the logarithms into sums of logarithms based on the factorization given by the provided structure, our optimization problem reduces to the maximization of $\sum TPC_{gk} \cdot \log \theta_{gk} + \sum TPC_{lck} \cdot \log \theta_{lck}$. This is exactly the objective function used in theorem 4.7.1. This is equivalent to the fact that P^*(see the definition above) minimizes $KL(P, \cdot)$ out of all the distributions that factorize according to the given structure and obey the expert's sharing assumptions. $\square$

Theorem 7.2.2. *With an infinite amount of data, the distribution $\hat{P}$ given by the Maximum Likelihood estimators in Theorem 4.7.1 converges to P^* with probability 1.*

Proof. Assume the number of data points in a dataset sampled from P is denoted by n. According to the Law of Large Numbers, we have $lim_{n\rightarrow\infty}\frac{N_{lck}}{n} = TPC_{lck}$ and $lim_{n\rightarrow\infty}\frac{N_{gk}}{n} = TPC_{gk}$ with probability 1. This is equivalent to the fact the $\hat{P}$ converges to P^* with probability 1. □

Corollary 7.2.1. *If the true distribution P factorizes according to the given structure and if the parameter sharing provided by the expert is completely accurate, then the distribution $\hat{P}$ given by the Maximum Likelihood estimators in Theorem 4.7.1 converges to P with probability 1.*

Again, we mention that we analyzed the formal guarantees presented in this chapter using only two different types of Parameter Domain Knowledge. We are confident that these results can be extended to all other types of Parameter Domain Knowledge for which we derived closed form solutions in this thesis.

Chapter 8

Experiments

In this chapter we present experiments that demonstrate the benefits of Bayesian Network models that take advantage of Parameter Domain Knowledge when compared to similar models which choose to ignore this kind of knowledge. We present experiments on both synthetic and real world data. The purpose of creating artificial data is threefold. First, it allows us to control and know the Parameter Domain Knowledge involved in the model, in the absence of a Domain Knowledge expert. Second, we are able to assess the performance of our models in terms of KL divergence from the true underlying distribution, which would be impossible in a real world situation. Finally, we can control and study the effect of variations of different parameters in the true distribution (e.g. the fraction of parameters that are truly shared) as well as the effect of varying the size of the training set. However, for real world data, we do not have access to the true underlying distribution and therefore we cannot compute the KL divergence. In this case we will assess our models using the Average Log Score described in Chapter 2. A reason to use this measure is that, according to the Law of Large Numbers, this score converges to the negative of the Cross-Entropy between the true and estimated distributions when the number of test examples goes to infinite. Another reason to use the Average Log Score is that it is proportional to the Log-Likelihood of the test data.

Previously published experiments involving learning with Module Networks, HMMs, DBNs or Context Specific Independence all support the theory presented in this thesis since they are particular cases of our Parameter Sharing Framework. However, the parameter sharing assumptions in these earlier models are at the level of either entire conditional probability table or entire conditional probability distribution. In the first two sections of this chapter we present experimental results showing the benefits of incorporating finer-grained parameter sharing assumptions when training Bayesian Networks on a task of learning a discrete probability distribution and on a task of modelling email coming from various sources. Third section will show how parameter sharing in the case of Hidden Process Models can help to better describe the fMRI signal associated with a cognitive task.

8.1 Synthetic Data - Estimating Parameters of a Discrete Variable

In this section we present experiments on one of the simplest forms of Parameter Domain Knowledge: Parameter Sharing within One Distribution. The purpose of these experiments is purely demonstrative and more complicated scenarios will be presented in the following sections of this chapter.

8.1.1 Experimental Setup

Our task is to estimate the set of parameters of a Bayesian Network which consists of one discrete variable X. We assume that the distribution of X shares some parameters and that the sharing can be provided by a domain expert. Without loss of generality, we may consider that the Parameter Domain Knowledge states that the parameters to estimate are given by $\theta = \{\theta_1, \ldots, \theta_n\}$ where each θ_i appears in $k_i \geq 1$ known places in the distribution of X.

Let us see how our synthetic dataset was created. First, we randomly generated a distribution T (the "true distribution") that exhibits parameter sharing. This distribution described a variable X with 50 values, which had a total of roughly 50% shared parameters i.e. $\sum_{k_i>1} k_i \approx \sum_{k_i=1} k_i$. Each distinct parameter appeared at most 5 times. We start with an empty distribution and generate a uniformly random parameter v between 0 and 1. Then we generate a random integer s between 2 and 5 and share v in the first s places of the distribution. We continue to generate shared parameters until we reach 25 (50% of 50 parameters). After that, we generate the rest of parameters uniformly randomly between 0 and 1. After all 50 parameters are obtained using this procedure, we normalize to yield a valid probability distribution. Once this distribution was generated, we sampled it to obtain a dataset of 1000 examples which were used subsequently to perform parameter estimation.

In our experiments we compare two models that estimate the parameters of distribution T over X. One is a standard Bayesian Network (STBN) that is learnt using standard Bayesian Networks estimators from Theorem 2.2.1. The second model (PDKBN) is a Bayesian Network that is learnt by using the results in 4.3 assuming the correct parameter sharing was specified by an oracle. While STBN needs to estimate $\sum_{i=1}^{n} k_i$ parameters, PDKBN only needs to estimate n parameters. To deal with potentially zero observed counts, we used priors on the parameters of the two models and then perform Maximum Aposteriori estimation. For STBN we introduced a Dirichlet count of 2 for each parameter while for PDKBN we used a Constrained Dirichlet count of $k_i + 1$ for each distinct parameter θ_i.

8.1.2 Results and Discussion

We performed parameter estimation of models STBN and PDKBN by varying the number of examples in the training set from 1 to 1000. Since we were using synthetic data, we were able to assess performance by computing KL(T,STBN) and KL(T,PDKBN), the KL divergence from the true distribution T.

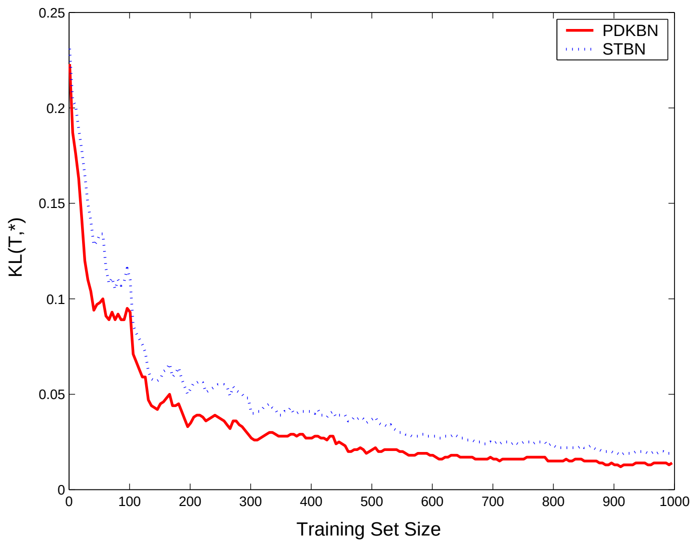

Figure 8.1: KL divergence of PDKBN and STBN with respect to correct model T.

Figure 8.1 shows a graphical comparison of the performance of the two models. It can be seen that our model (PDKBN) that takes advantage of Parameter Domain Knowledge consistently outperforms the standard Bayesian Network model. The difference between the two models is higher when the learning is performed from a smaller number of examples. The highest observed difference between KL(T,STBN) and KL(T,PDKBN) was 0.05 and was observed when the two models were trained using 30 examples. As expected, when the amount of training data increases, the difference in performance between the two models decreases dramatically, since both STBN and PDKBN are unbiased models that will eventually converge to the true distribution T.

Training Examples	KL(T,PDKBN)	Examples needed by STBN
5	0.191	16
40	0.094	103
200	0.034	516
600	0.018	905
650	0.017	> 1000

Table 8.1: Equivalent training set size so that STBN achieves the same performance as PDKBN.

To get a better idea how beneficial Parameter Domain Knowledge is in this case, we want to see *"how far STBN is behind PDKBN"*. For a model PDKBN learnt from a dataset of a given size, this can be measured by the number of examples that STBN needs to be learnt from in order to achieve the same performance. Table 8.1 provides these numbers for several training set sizes for PDKBN. For example, STBN uses 16 examples to achieve same KL divergence as PDKBN at 5 examples, which is a factor of 3.2 (the maximum observed) increase in the number of training samples required by STNB. On the average, STBN needs 1.86 times more examples to perform as well as PDKBN.

As mentioned previously, this section was intended to be only a proof of concept. Next we will see experimental results on much more complex tasks involving multiple random variables and Parameter Domain Knowledge constraints across several conditional probability distributions.

8.2 Semi-synthetic Data - Email Experiments

Automatic modelling of email documents is a problem of considerable interest, and has been studied as a means of automatically sorting email into a user's email subfolders, or into other topical categories such as "email spam" [SDH98]. Given a set (population) of email, one natural way to model it is by using a Bayesian Multinetwork where each distinct email *author* is regarded as a generator of email from a different distribution, forming a different subpopulation. This is reasonable because different people use somewhat different vocabularies and figures of speech. At the same time, there are many content words that are shared (used in a similar proportion) by all authors when emails address specific topics. The conditional probabilities of these words given that the email is about a certain topic should therefore be specified as globally shared parameters across the subpopulations in the Bayesian Multinet.

8.2.1 Experimental Setup

In our experiments we generated synthetic data that captures the characteristics of a real email data set: the PW CALO email corpus, produced by people in a role-playing game at SRI. During four days, a group of six players assumed different work roles (e.g. project leader, finance manager, researcher, administrative assistant, etc), and communicated via email. The number of emails sent per author varied from 19 to 52, forming a total corpus containing 215 distinct emails. One common task performed through these email exchanges was to setup a meeting and so the emails were manually labelled in emails about or not about meetings. Of the 215 emails, 68 were about meetings, with the per-author fraction of meeting emails varying from 0.17 to 0.47. After eliminating stop words, the entire email corpus contained a vocabulary of 1070 distinct words. With few exceptions, the emails contained less than 150 words each.

Consider the task of scheduling a meeting. In this case, each email author may have a different style of wording meeting invitations. Some authors may use more formal wording ("would you please meet with me ..."), whereas others may use a different more informal phrasing ("let's get together ..."). Given these different styles, it is reasonable to model incoming email in terms of a Multinetwork of author-specific Bayesian networks. Notice there are certain words whose probability given a meeting email is likely to be globally shared across users, such as the words "monday" or "tuesday". The conditional probabilities of these words should therefore be specified as globally shared parameters across the Bayesian Multinetwork.

Based on the corpus mentioned above, we generated several artificial datasets. Each data point consists of a triple: (*Author*, *Email* and *Topic/Class*). The Topic says whether or not the email is about a meeting. In all experiments we generated simulated email from six authors, using the same prior probabilities of an email belonging to an author as in the PW CALO corpus, and generating emails from each author to match that author's topic priors in this corpus. For each given author and topic, we generate emails according to a "Bag of Words" probability model, where each email contains between 0 and 150 words (to be consistent with the data observed in PW, even though we are not including stop words in our artificial dataset). The words are chosen from a vocabulary of 1070 words (same size as in PW). The word given topic probability models are generated randomly and differ from user to user, but also have some fraction of parameters in common (the so called shared parameters in our framework). The fraction f of shared parameters was varied from 0 to 1. First we uniformly randomly pick an f fraction of the word given topic parameters that are going to be shared across all email authors. We uniformly at random generate the word given topic probability distributions for the first author (we generate randomly parameters between 0 and 1 and then normalize to obtain valid probability distributions), then we copy the shared parameters across the distributions corresponding to all authors. After this step, for all authors except the first one, the non-shared (local) parameters are generated uniformly randomly between 0 and 1 and then normalized so that, when summed up with the shared parameters, they yield valid probability distributions.

In our experiments we compare three models. First, a General Naive Bayes model (GNB) learned from all training examples. Second, a Bayesian Multinet (SSNB) in which each component network is a Naive Bayes model, and for which an oracle has indicated which parameters are shared when generating the data. Finally, a Bayesian Multinet (PSNB) identical to SSNB, but with no parameters shared among component networks. Note all three models are essentially Bayesian Multinets, conditioned on the email author which is observed in the header of the email. Each component network in each Bayes Multinet is a full naive Bayes model including both the Class/Topic variable and the word features in the email. One can think of GNB as a Bayesian Multinet where the component Bayes nets are copies of the GNB learned model. The only difference among the three is in the training procedure. They differ in their sharing of parameters (all shared in GNB, some shared in SSNB, not shared in PSNB). There is also a slight difference in the way we assign

Dirichlet priors (we train all three models using MAP estimates, as is common when training Naive Bayes models from sparse data). In the case of GNB the effect is to increase each word count by one (equivalent to Dirichlet priors with all parameters equal to 2). In the case of PSNB and SSNB, for each subpopulation the effect is to increase that subpopulation's word counts by one. For PSNB, this is equivalent to Dirichlet priors with all parameters equal to 2 for each subpopulation, while for SSNB this is equivalent to assigning subpopulation-specific Dirichlet priors with parameters equal to 7 for shared parameters (7 is the number of subpopulations plus 1 in our experiments), and equal to 2 for local parameters.

Notice also that whereas the GNB model is biased (i.e., unable to represent precisely the distribution used to generate the data), both the SSNB and PSNB models are unbiased. Furthermore, the SSNB model has the additional benefit that it mixes global and local parameters in modelling the generating distribution, resulting in lower variance parameter estimates than PSNB.

8.2.2 Results and Discussion

We trained the three different models while varying the number of training examples, and the fraction of word-given-class model parameters that were global (identical across authors). For each model, we measured both the KL divergence $KL(T, M)$ of the learned model M to the true generating model T, and the accuracy of the corresponding *meeting versus non-meeting* classifier.

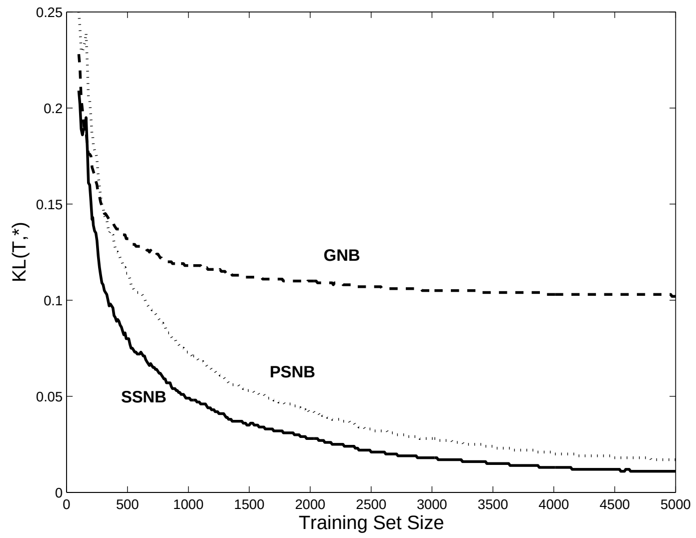

Figure 8.2: KL divergence of learned models with respect to correct model T.

Figure 8.2 shows a plot of the KL divergence for each of the three models, as the number of training examples varies, keeping the fraction of general parameters constant at 0.5. As expected,

$KL(T, *)$ decreases with increasing training set size for all three models (here $*$ stands for any possible model we are studying). However, SSNB outperforms the other two models across the entire range of training set size. It dominates PSNB especially at small training set sizes, because its shared global parameters allow it to produce lower-variance parameter estimates, especially when data is sparse. The highest difference between KL(T,PSNB) and KL(T,SSNB) was 0.045 and was observed for a training set of 160 emails. SSNB dominates GNB especially with larger training sets, because it is capable of representing the correct model, whereas GNB considers a strictly smaller model class that does not contain the correct model. The highest difference between KL(T,GNB) and KL(T,SSNB) was 0.092 and was observed for a training set of 5000 emails. Note that asymptotically, as the training set size approaches infinity, the SSNB and PSNB models will both converge to the correct model, whereas GNB will not.

We also study the impact of varying the fraction of global parameters in the true underlying probability model from 0 to 1, while holding the training set size constant at 1000 emails. Here the KL(T,PSNB) is essentially constant as the fraction of true global parameters varies, because PSNB does not take advantage of parameter sharing. In contrast, both GNB and SSNB improve considerably with an increasing fraction of global parameters. However, GNB performs poorly when parameters are not shared, because it assumes all parameters are shared. Again, SSNB dominates the other two methods, as it can mix global and local parameters in its model.

In addition to KL divergence, we also considered classification accuracies of the models, again varying training set size and holding the fraction of global parameters at 0.5. Training accuracy was measured over a set of 10,000 examples which was not touched by training. The relative performance of GNB, SSNB, and PSNB followed the same trends as when measuring KL divergence, although the margins separating the methods were less dramatic. In fact, accuracies of all three methods were quite high, reaching .978, .988, and .985 respectively when training on 5000 examples. Similar trends with lower accuracies were observed when using a reduced number of words per document. It is well known that the naive Bayes algorithm can in some cases achieve high accuracies even when the underlying generative model of the data is inaccurate [DP97].

Another interesting way to analyze the results is to look at the training set size needed by PSNB, the model that does not use Parameter Domain Knowledge, to achieve the same KL divergence or accuracy as the SSNB model. Table 8.2 shows these numbers for several training set sizes used by model SSNB. For example, PSNB uses 670 examples to achieve same KL divergence as SSNB at 350 examples, which is a factor of 1.91 (the maximum observed) increase in the number of training samples needed by PSNB. As an another example, PSNB uses 1300 examples to achieve the same accuracy as SSNB at 550 examples, which is a factor of 2.36 (the maximum observed) increase in training set size for PSNB. On the average, PSNB needs 1.69 times more examples than SSNB to achieve the same KL divergence and it needs 1.97 times more examples to achieve the same

Training Examples	KL(T,SSNB)	Examples needed by PSNB	Accuracy of SSNB	Examples needed by PSNB
100	0.209	180	0.937	180
350	0.097	670	0.976	620
550	0.074	970	0.981	1300
1000	0.049	1690	0.983	2100
3000	0.018	4790	0.987	> 5000

Table 8.2: Equivalent training set size so that PSNB achieves the same performance as SSNB.

accuracy. Note that it does not make sense to perform a similar comparison between SSNB and GNB because, on the long run, GNB can never catch up with $SSNB$ in terms of KL divergence.

To summarize, taking advantage of Parameter Domain Knowledge on this email modelling task had not only the effect of obtaining much better estimators for small training set size, but also had the effect of reducing by a big factor the number of examples that would otherwise be needed by a model which does not use any Parameter Domain Knowledge to achieve the same performance.

8.3 Real World Data - fMRI Experiments

Functional Magnetic Resonance Imaging (fMRI) is a technique for obtaining three-dimensional images of activity in the brain throughout time. More precisely, fMRI measures the ratio of oxygenated hemoglobin to deoxygenated hemoglobin in the blood with respect to a control baseline, at many individual locations within the brain. This is often referred to as the blood oxygen level dependent (BOLD) response. The BOLD response is taken as an indicator of neural activity.

An fMRI scanner records a 3D image of the brain as a collection of parallel slices. Each such slice contains a collection of small cells, called voxels. A voxel has a resolution of few tens of cubic milliliters and can contain hundreds of thousands of neurons. In our dataset, there are eight parallel slices for each fMRI snapshot and the dimension of each voxel is few tens of cubic millimeters. Typically, there are ten to fifteen thousand voxels in a human brain. However, only a part of them are available in our dataset.

During an fMRI experiment, a subject is asked to perform several trials of a cognitive task while the fMRI scanner is monitoring the BOLD signal. It is common for a trial to last few tens of seconds, a snapshot of the brain being captured once or twice per second. A common use for the data collected in these trials is to come up with regions of the brain that are active during the performed cognitive task. A slightly different approach was taken in [Mit02, Mit03, Mit04], where the authors used the fMRI signal to classify different cognitive states in which the subject may be at different points in time. As is the case with the Naive Bayes classifier, it is well known that models that perform very well on a classification task may represent poorly the underlying structure of the data.

As opposed to the approaches above, in this section we present a generative model of the activity in the brain during a cognitive task based on parameter sharing assumptions for the Hidden Process Models (see section 6.3) that describe the fMRI signal. These parameter sharing assumptions are not readily available, but we successfully employ the methods described in section 3.6 to automatically discover clusters of voxels that can be learnt together using Shared Hidden Process Models. We show that our methods far outperform the baseline Hidden Process Model that is learnt on a per voxel basis.

8.3.1 Experimental Setup

We experimented using the *StarPlus* dataset ([CJK99]). The StarPlus experiment was designed to engage several different cortical areas, in order to look at their interaction. In this dataset, each subject first sees a sentence(semantic stimulus) for 4 seconds, such as “The plus sign is above on the star sign.”, then a blank screen for 4 seconds, and finally a picture(symbol stimulus) such as

$$\frac{+}{*}$$

for another 4 seconds. At any time after the picture was presented, the subject may press a button for “yes” or “no”, depending on whether the sentence matches the picture seen or not. The subject is instructed to rehearse the sentence in his/her brain until the picture is presented rather than try to visualize the sentence immediately. The second variant switches the presentations of sentences and pictures, and the instruction is to keep the picture in mind until the presentation of the sentence.

In this dataset, the voxels are grouped in 24 ROIs (Regions of Interest, defined based on brain anatomy), each voxel having a resolution of 3 by 3 by 5 millimeters. A snapshot of the brain is taken every half second. In this dataset there are three main conditions: *fixation* (the subject is looking at a point on the screen), *sentence followed by picture* and *picture followed by sentence*. We have 10 trials in *fixation* and 20 in each of the other two conditions. For each trial, we kept 32 (16 seconds) snapshots of the brain.

Our goal is to come up with a model that best explains the activity in the brain when a subject is either reading a sentence or looking at a picture. After we discard the fixation trials (they contain no information relevant to our task), we are left with a total of 40 examples per subject, each example consisting of activity generated by both stimuli. While there is data available for multiple subjects, there are difficulties in merging this data for the purpose of parameter estimation. This happens because different subjects have different brain shapes and because different subjects exhibit different intensity in activity when presented with the same cognitive task. Therefore we limited to analyzing data separately for each subject. The results reported in this section are all based on the same subject (04847). For this particular subject, our dataset tracked the activity of 4698 voxels.

We consider that the activity in each voxel is described by a Hidden Process Model with two processes, corresponding to the two stimuli: a *Sentence* process and a *Picture* process. The starting time of these processes are known in advance, given the structure of the trials described above. In half of the trials, the *Sentence* process starts at time 1 and in the other half it starts at time 17. The same holds for the *Picture* process. We make the assumption that the activity in different voxels is independent given the hidden processes corresponding to these voxels.

In our experiments we compare three models. Since the true underlying distribution is not available in this case, we use the Average Log Score (see section 2.2.7) to assess performance. Because the data is scarce, we can not afford to keep a held-out testing set. Therefore we employ a leave-two-out cross-validation approach to estimate the performance of our models. First model *StHPM*, which we will consider as a baseline, consists of a standard Hidden Process Model learnt independently for each voxel. The second model *ShHPM* is a Hidden Process Model, shared for all the voxels in an ROI. In other words, all voxels in a specific ROI share the same shape hidden processes, but with different amplitudes (see Section 6.3 for more details). *ShHPM* is learned using Algorithm 6.3.1.

With only 40 training examples, the task of estimating the parameters can prove more than challenging. Therefore we would definitely benefit from an expert's domain knowledge saying which groups of neighboring voxels are described by a Shared Hidden Process Model. We have seen that *ShHPM* makes the assumption that each ROI is a Shared Hidden Process Model. However, this assumption might not always be true. In the absence of a domain expert, we propose an algorithm which allows us to both automatically discover clusters of voxels that form a Shared Hidden Process Model and estimate the corresponding parameters. This third model (*HieHPM*) uses a nested cross-validation hierarchical approach to both come up with a partition of the voxels in clusters that form a Shared Hidden Process Model and estimate its corresponding performance on examples not used in training:

Algorithm 8.3.1. (Hierarchical Partititioning and Hidden Process Models learning)

STEP 1. *Split the 40 examples in a set of 20 folds $F = \{F_1, \ldots, F_{20}\}$, each fold containing one example where the sentence is presented first and an example where the picture is presented first.*

STEP 2. *For all $1 \leq k \leq 20$, keep fold F_k aside and learn a model from the remaining folds using Steps 3-5.*

STEP 3. *Start with a partition of all voxels in the brain by their ROIs and mark all subsets as* Not Final.

STEP 4. *While there are subsets in the partition that are* Not Final*, take any such subset and try to split it using equally spaced hyperplanes on all three directions (in our experiments we split each subset in 2 by 2 by 4 smaller subsets). If the cross-validation Average Log Score of the model learnt from these new subsets using Algorithm 6.3.1 (based on folds $F \setminus F_k$) is lower than the cross-validation Average Log Score of the initial subset for folds in $F \setminus F_k$, then mark the initial subset as* Final *and discard its subsets. Otherwise remove the initial subset from the partition and replace it with its subsets which then mark as* Not Final.

STEP 5. *Given the partition computed by STEPS 3 and 4, based on the 38 data points in $F \setminus F_k$, learn a Hidden Process Model that is shared for all voxels inside each subset of the partition. Use this model to compute the log score for the examples/trials in F_k.*

STEP 6. *In Steps 2-4 we came up with a partition for each fold F_k. To come up with one single model, compute a partition using STEPS 3 and 4 based on all 20 folds, then, based on this partition learn a model as in STEP 5 using all 40 examples. The Average Log Score of this last model can be estimated by averaging the numbers obtained in STEP 5.*

8.3.2 Results and Discussion

We estimated the performance of our models using the Average Log Score (described in section 2.2.7) based on a leave two out cross-validation approach, where each fold contains an example where the sentence is presented first and an example where the picture is presented first.

Our first set of experiments, summarized in Table 8.3, compared the three models based on their performance in the Visual Cortex (CALC). This is one of the ROIs actively involved in this cognitive task and contains 318 voxels. The training set size was varied from 6 examples to all 40 examples, in multiples of two. As in the previous sections, sharing parameters of Hidden Process Models proved very beneficial and the impact was observed best when the training set size was the smallest. With an increase in the number of examples, the performance of *ShHPM* starts to degrade because it makes the biased assumption that all voxels in CALC can be described by a single Shared Hidden Process Model. While this assumption paid off with small training set size because of the reduction in variance, it definitely hurt in terms of bias with larger sample size. Even though the bias was obvious in CALC, we will see in other experiments that in certain ROIs, this assumption holds and it those cases the gains in performance may be pretty big. Also, note that the Average Log Score computed at small sample size may not be a very reliable measure of the true performance.

Training Trials	No Sharing (StHPM)	All Shared (ShHPM)	Hierarchical (HieHPM)	Cells (HieHPM)
6	-30497	-24020	-24020	1
8	-26631	-23983	-23983	1
10	-25548	-24018	-24018	1
12	-25085	-24079	-24084	1
14	-24817	-24172	-24081	21
16	-24658	-24287	-24048	36
18	-24554	-24329	-24061	37
20	-24474	-24359	-24073	37
22	-24393	-24365	-24062	38
24	-24326	-24351	-24047	40
26	-24268	-24337	-24032	44
28	-24212	-24307	-24012	50
30	-24164	-24274	-23984	60
32	-24121	-24246	-23958	58
34	-24097	-24237	-23952	61
36	-24063	-24207	-23931	59
38	-24035	-24188	-23921	59
40	-24024	-24182	-23918	59

Table 8.3: The effect of training set size on the Average Log Score of the three models in the Visual Cortex (CALC) region.

However, it is the best we can do based on a small dataset.

As expected, the hierarchical model *HieHPM* performed better than both *StHPM* and *ShHPM* because it takes advantage of Shared Hidden Process Models while not making the restrictive assumption of sharing across whole ROIs. The highest difference between *HieHPM* and *StHPM* is observed at 6 examples, in which case *StHPM* basically fails to learn a ”decent” model while the highest difference between *HieHPM* and *ShHPM* happened with the maximum number of examples, when *ShHPM* started to be hurt by its bias. As the amount of training data increases, both *StHPM* and *HieHPM* tend to perform better and better and one can see that the difference in performance given by the addition of two new examples tends to shrink as both models approach convergence. While with infinite amount of data, one would expect $StHPM$ and $HieHPM$ to converge to the true model, at 40 examples, *HieHPM* still outperforms the baseline model *StHPM* by a difference of 106 in terms of Average Log Score, which is an improvement of e^{106} in terms of data likelihood.

Probably the measure that shows best how much better is *HieHPM* than the baseline *StHPM* is given by how many more examples *StHPM* needs to achieve the same performance as *HieHPM*. It turns out that on the average, *StHPM* needs roughly 2.9 times more examples in order to perform same as well as *HieHPM* in the Visual Cortex (CALC).

The last column of Table 8.3 displays the number of clusters of voxels in which *HieHPM* partitioned CALC. As one may notice, at small sample size, *HieHPM* draws its performance from gains in variance by using only one cluster of voxels. However, as the amount of data increases, *HieHPM* improves by finding more and more refined partitions. This number tends to stabilize around 60 clusters once the number of examples reaches 30, which means an average of more than 5 voxels per cluster given that CALC is made of 318 voxels. For a training set of 40 examples, the largest cluster has 41 voxels while a lot of clusters are made of only one voxels.

The second set of experiments (see Table 8.4) describes the performance of the three models on all 24 ROIs of the brain as well on all the brain. While we have seen that *ShHPM* was biased in CALC, we may see here that there are several ROIs where it makes sense to characterize all voxels by a Shared Hidden Process Model. In fact, in most of these regions, *HieHPM* finds only one cluster of voxels. Actually, *ShHPM* outperforms the baseline model *StHPM* in 18 out of 24 ROIs while *HieHPM* outperforms *StHPM* in 23 ROIs. One may ask how can possibly *StHPM* outperform *HieHPM* on a ROI, since *HieHPM* may also represent the case when there is no sharing? The explanation is that the hierarchical approach can get stuck in a local maximum of the data log-likelihood over the search space if it cannot improve by splitting at a specific step since it does not look beyond that split for a finer grained partition. Fortunately, this problem is extremely rare, as we have seen in our experiments.

Over the whole brain, *HieHPM* outperforms *StHPM* by a factor of e^{1792} in terms of data likelihood while *ShHPM* outperforms *StHPM* only by a factor of e^{464}. However, the main drawback of the *ShHPM* is that it can be biased and therefore our experiments recommend *HieHPM* as the clear winner. Next we are going to give the reader a feel of what the model *HieHPM* looks like.

As mentioned above, *HieHPM* automatically learns clusters of voxels that can be represented using a Shared Hidden Process Model. Figure 8.3 shows the portions of these learned clusters in slice five of the eight vertical slices of the image of the brain taken by the fMRI scanner. Neighboring voxels that were assigned by *HieHPM* to the same cluster are pictured with the same color. Note that there are several very large clusters in this picture. This may be because of the fact that it makes sense to represent whole ROIs using a Shared Hidden Process Model if the studied cognitive task does not involve those areas of the brain. However, large clusters are also found in areas like CALC, which we know is directly involved in any visual activity.

In Figure 8.4 we can see the learned *Sentence* hidden process for the voxels in the Visual Cortex (CALC). Again, the graphs corresponding to voxels that belong to the same cluster have been

ROI	Voxels	No Sharing (StHPM)	All Shared (ShHPM)	Hierarchical (HieHPM)	Cells Hierarchical
CALC	318	-24024	-24182	-23918	59
LDLPFC	440	-32918	-32876	-32694	11
LFEF	109	-8346	-8299	-8281	6
LIPL	134	-9889	-9820	-9820	1
LIPS	236	-17305	-17187	-17180	8
LIT	287	-21545	-21387	-21387	1
LOPER	169	-12959	-12909	-12909	1
LPPREC	153	-11246	-11145	-11145	1
LSGA	6	-441	-441	-441	1
LSPL	308	-22637	-22735	-22516	4
LT	305	-22365	-22547	-22408	18
LTRIA	113	-8436	-8385	-8385	1
RDLPFC	349	-26390	-26401	-26272	40
RFEF	68	-5258	-5223	-5223	1
RIPL	92	-7311	-7315	-7296	11
RIPS	166	-12559	-12543	-12522	20
RIT	278	-21707	-21720	-21619	42
ROPER	181	-13661	-13584	-13584	1
RPPREC	144	-10623	-10558	-10560	1
RSGA	34	-2658	-2654	-2654	1
RSPL	252	-18572	-18511	-18434	35
RT	284	-21322	-21349	-21226	24
RTRIA	57	-4230	-4208	-4208	1
SMA	215	-15830	-15788	-15757	10
All Brain	4698	-352234	-351770	-350441	299

Table 8.4: Per ROI performance of the three models when learned using all 40 examples.

painted in the same color, which is also the same with the color used in Figure 8.3. To make these graphs readable, we only plotted the base process, disregarding the scaling (amplitude) constants corresponding to each voxel within a given cluster (consult Section 6.3 for more details about Shared Hidden Process Models).

A magnified example of the base *Sentence* hidden process in one voxel from CALC is shown

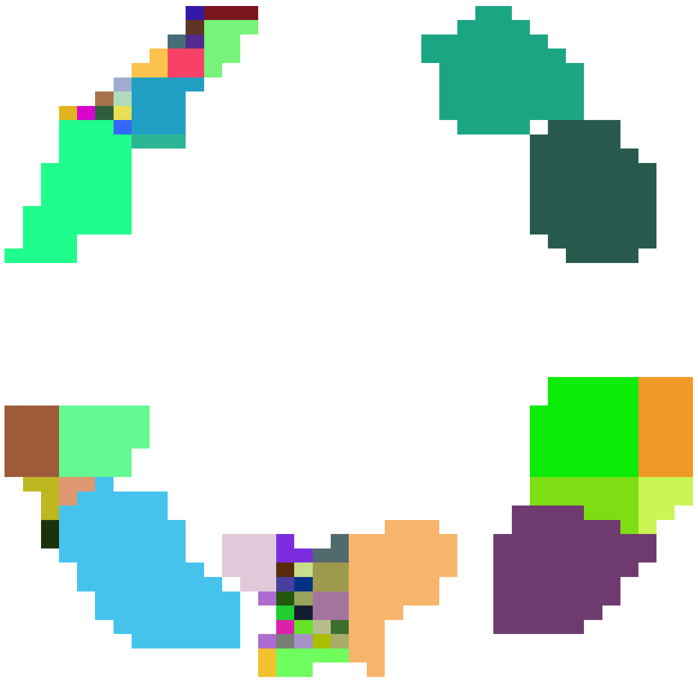

Figure 8.3: Parameter Sharing found using model *HieHPM*. Slice five of the brain is showed here. Shared neighboring voxels have the same color.

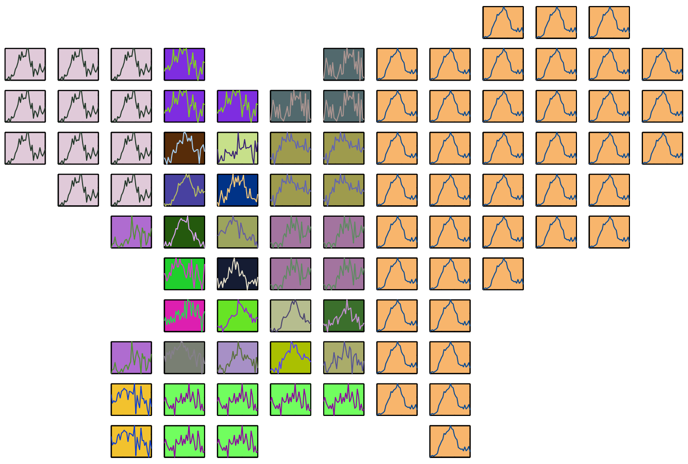

Figure 8.4: Per voxel base *Sentence* processes in the Visual Cortex(CALC).

in figure 8.5. This curve is consistent with a typical BOLD response given a stimulus: when the stimulus is presented, there is a surge in activity in a voxel which peaks after several seconds, then,

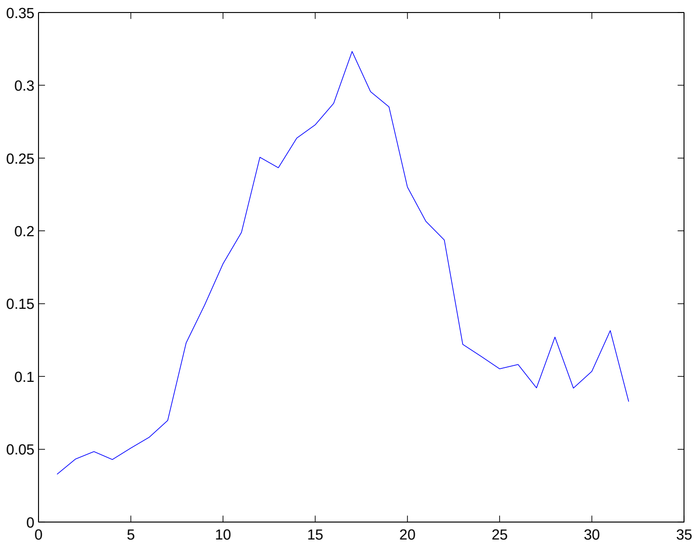

Figure 8.5: Magnified example of the *Sentence* process in a shared voxel in the Visual Cortex (CALC). The horizontal axis represents the fMRI snapshot (one each half second) after the sentence was presented and the vertical axis represents the value of the fMRI signal corresponding to the *Sentence* process.

after the stimulus is gone, the activity in that voxel will eventually drop back to the baseline rest signal.

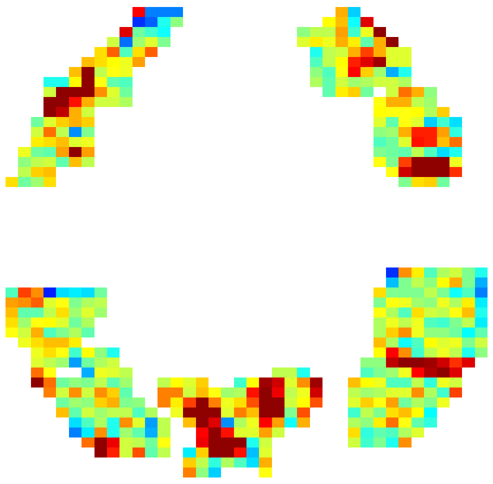

Figure 8.6: Magnitude of the scaling constants corresponding to the *Sentence* process.

While in Figure 8.4 we presented the base *Sentence* process for each voxel in CALC, in Figure 8.6 we color coded the scaling/amplitude of the *Sentence* signal in each of these voxels. Same color amplitude for voxels that belong to the same cluster means those voxels exhibit the same activity when the subject is reading a sentence. Color *Red* corresponds to higher values of the amplitude, while *Blue* stands for low such values. It is easy to see there exists a slight grouping of the voxels based on their amplitudes: within one cluster, there tends to exist a group of voxels with high amplitude while when going farther away from that group, the amplitude decreases.

To summarize, in this section we learned three different generative models for the fMRI signal during a cognitive task, all based on Hidden Process Models. We proved that Parameter Sharing for Hidden Process Models (as defined in Section 6.3) can greatly benefit learning. Our hierarchical *HieHPM* model outperformed the other two models because it is both unbiased and able to reduce variance in the parameter estimators by automatically finding clusters of voxels that can be described by a Shared Hidden Process Model.

Chapter 9

Conclusions and FutureWork

The research presented in this thesis is intended to be a first step in methods aimed at taking advantage of Parameter Domain Knowledge for the task of learning Bayesian Networks. We foresee a lot of potential for additional research in this area. In this chapter we summarize the contributions of this thesis and we suggest several interesting directions for future work.

9.1 Conclusions

Building accurate models from a small amount of available training data can sometimes prove to be a great challenge. Expert domain knowledge can be often used to alleviate this burden. In this thesis we presented the basis of a sound mathematical framework for incorporating Parameter Domain Knowledge in learning procedures for Bayesian Networks. We proved both theoretically and experimentally that the standard methods of performing parameter estimation in Bayesian Networks can be naturally extended to take advantage of Parameter Domain Knowledge that can be provided by a domain expert.

The most important contribution of this thesis was the development of a unified framework for incorporating general Parameter Domain Knowledge constraints in estimators for the parameters of a Bayesian Network by phrasing the goal as a constraint optimization problem. We showed how to compute Maximum Likelihood estimators using an iterative procedure based on Newton-Raphson method. This procedure can be computationally expensive, but fortunately, in practice, the optimization problem can be broken down into a set of many smaller, independent, optimization subproblems. Then, we discussed how to define Constrained Parameter Priors and perform learning from a Bayesian point of view. We also demonstrated how our methods can be extended in the case when the data is partially observable.

Parameter Domain Knowledge Type	Results
Known Parameters, Discrete	Closed Form MLEs, MAP, Normalization Constant
Parameter Sharing, One Distribution, Discrete	Closed Form MLEs, MAP, Normalization Constant
Proportionality Constants, One Distribution, Discrete	Closed Form MLEs, MAP, Normalization Constant
Sum Sharing, One Distribution, Discrete	Closed Form MLEs
Ratio Sharing, One Distribution, Discrete	Closed Form MLEs
General Parameter Sharing, Multiple Distributions, Discrete	Closed Form MLEs, MAP, Normalization Constant
Hierarchical Parameter Sharing, Multiple Distributions, Discrete	Closed Form MLEs, MAP, Normalization Constant
Sum Sharing, Multiple Distributions, Discrete	Closed Form MLEs
Ratio Sharing, Multiple Distributions, Discrete	Closed Form MLEs
Inequalities between Sums of Parameters, One Distribution, Discrete	Closed Form MLEs
Upper Bounds on Sums of Parameters, One Distribution, Discrete	Closed Form MLEs
Parameter Sharing, One Distribution, Continuous	Closed Form MLEs
Proportionality Constants, One Distribution, Continuous	Closed Form MLEs
Parameter Sharing for Hidden Process Models	Efficient Iterative Method to Compute MLEs
Twice Differentiable with Continuous Second Derivatives	Iterative Methods: Frequentist, Bayesian, Complete and Incomplete Data

Table 9.1: Domain Knowledge Types studied in this thesis: description and results.

The iterative procedure mentioned above can be quite expensive. Therefore, it is preferable to derive closed form solutions for our estimators. This is not possible for general domain knowledge constraints but fortunately it is possible for several types of constraints, including parameter sharing of different kinds, as well as relationships among groups of parameters. Table 9.1 summarizes the results that we derived for these types of Parameter Domain Knowledge. Examples for each specific type of domain knowledge were described in Tables 9.2 and 9.3. We approached learning of both discrete and continuous variables, in the presence of both equality and inequality constraints. While for most of these types of Domain Knowledge we can derive closed form Maximum Likelihood estimators, we come up with a very efficient iterative algorithm to perform the same task for Shared Hidden Process Models. In many of these cases, for discrete variables, we are also able to compute closed form normalization constants for the corresponding Constrained Parameter Priors, which allows us to perform closed form MAP and Bayesian estimation when the data is complete. We want to point out here that our General Parameter Sharing Framework can encompass models including HMMs, Dynamic Bayesian Networks, Module Networks and Context Specific Independence as particular cases, but allows for much finer grained sharing, at parameter level, across different variables and across distributions of different lengths. It is also important to note that we can mix different types of Parameter Domain Knowledge constraints when learning the parameters of a Bayesian Network as long as the scopes of these constraints do not overlap.

Experimental results on fMRI data proved that taking advantage of domain knowledge can be very beneficial for learning. Since the domain knowledge was not always readily available, we developed methods to automatically uncover this knowledge. Using these methods we discovered clusters of voxels that can be learned together using Shared Hidden Process Models. Our results showed that the effect of the learned Parameter Domain Knowledge can be equivalent to almost tripling the size of the training set on this task. This was a pessimistic estimate of the benefits since we had to extract the domain knowledge from the training data itself via the cross-validation approach described in section 3.6. Experiments on synthetic data were also performed and they exhibited the same beneficial effect of incorporating Parameter Domain Knowledge.

A very important result that we managed to prove was that the estimators taking advantage of a simple form of Parameter Sharing achieved total variance lower than the one of estimators that ignored such domain knowledge. We conjecture that similar results hold for other types of domain knowledge, but their proof is left as future work.

In all the approaches above, we assumed the domain knowledge is correct. However, even when the domain expert makes mistakes, we proved that, with infinite amount of data, our Maximum Likelihood estimators would converge to the ”best distribution” (the closest in terms of KL distance from the true distribution) that obeys the expert’s assumptions and factorizes according to the given structure.

DK Type 1: Known Parameters, Discrete
Example: If a patient has a heart attack (Disease = ”Heart Attack”), then there is a 90% probability that the patient will experience chest pain.
DK Type 2: Parameter Sharing, One Distribution, Discrete
Example: Given a combination of risk factors, several diseases are equally likely.
DK Type 3: Proportionality Constants, One Distribution, Discrete
Example: Given a combination of risk factors, disease A is twice as likely to occur than disease B is.
DK Type 4: Sum Sharing, One Distribution, Discrete
Example: A patient who is a smoker has the same chance of having a Heart Disease (Heart Attack or Congestive Heart Failure) as having a Pulmonary Disease (Lung Cancer or Chronic Obstructive Pulmonary Disease).
DK Type 5: Ratio Sharing, One Distribution, Discrete
Example: In a bilingual corpus, the relative frequencies of certain groups of words are the same, even though the aggregate frequencies of these groups may be different. Such groups of words can be: ”words about computers” (”computer”, ”mouse”, ”monitor”, ”keyboard” in both languages) or ”words about business”, etc. In some countries computer use is more extensive than in others and one would expect the aggregate probability of ”words about computers” to be different. However, it would be natural to assume that the relative proportions of the ”words about computers” are the same within the different languages.
DK Type 6: General Parameter Sharing, Multiple Distributions, Discrete
Example: The probability that a person will have a heart attack given that he is a smoker with a family history of heart attack is the same no matter whether the patient lives in a polluted area.
DK Type 7: Hierarchical Parameter Sharing, Multiple Distributions, Discrete
Example: The frequency of several *international words* (for instance ”computer”) may be shared across both Latin languages (Spanish, Italian) and Slavic languages (Russian, Bulgarian). Other Latin words will have the same frequency only across Latin languages and the same holds for Slavic Languages. Finally, other words will be language specific (for example names of country specific objects) and their frequencies will not be shared with any other language.
DK Type 8: Sum Sharing, Multiple Distributions, Discrete
Example: ”The frequency of nouns in Italian is the same as the frequency of nouns in Spanish.

Table 9.2: Domain Knowledge Types studied in this thesis: description and examples.

DK Type 9: Ratio Sharing, Multiple Distributions, Discrete	Example: In two different countries (A and B), the relative frequency of Heart Attack to Angina Pectoris as the main diagnosis is the same, even though the the aggregate probability of Heart Disease (Heart Attack and Angina Pectoris) may be different because of differences in lifestyle in these countries.
DK Type 10: Inequalities between Sums of Parameters, One Distribution, Discrete	Example: The aggregate probability mass of adverbs is no greater than the aggregate probability mass of the verbs in a given language.
DK Type 11: Upper Bounds on Sums of Parameters, One Distribution, Discrete	Example: The aggregate probability of nouns in English is no greater than 0.4.
DK Type 12: Parameter Sharing, One Distribution, Continuous	Example: The stock of computer maker *DELL* as a Gaussian whose mean is a weighted sum of the stocks of software maker *Microsoft (MSFT)* and chip maker *Intel (INTL)*. Parameter sharing corresponds to the statement that *MSFT* and *INTL* have the same importance (weight) for predicting the value of stock *DELL*.
DK Type 13: Proportionality Constants, One Distribution, Continuous	Example: Suppose we also throw in the stock of a Power Supply maker (PSUPPLY) in the linear mix in the above example. The expert may give equal weights to INTL and MSFT, but five times lower to PSUPPLY.
DK Type 14: Parameter Sharing for Hidden Process Models	Example: Several neighboring voxels in the brain exhibit similar activation patterns, but with different amplitudes when a subject is presented with a given stimulus.

Table 9.3: Domain Knowledge Types studied in this thesis: description and examples.

This research provided a new perspective of looking at learning procedures in the presence of domain knowledge about relationships among parameters. We feel that there is a lot of room for additional improvement in this exciting area of Parameter Domain Knowledge, an area which was barely explored so far.

9.2 Future Work

9.2.1 Interactions among Different Types of Parameter Domain Knowledge

In chapter 3 we presented an iterative method for estimating parameters in the presence of arbitrary constraints that respect some smoothness assumptions. Since this method can be expensive due to the dimensionality of the problem, it is of course preferable to be able to compute closed form estimators whenever possible. In chapters 4,5 and 6 we showed how this can be done for several types of domain knowledge. It is easy to see that our results do not break if we have a mixture of these types in the same Bayesian Network, in the case when different domain knowledge types are specified over disjoint sets of parameters. However, our results do not hold anymore if, for example, same set of distributions are constrained together by both a parameter sharing assumption and a ratio sharing assumption. It would be interesting to see if our methods can be extended to compute close form estimators for constraints of different domain knowledge types that have overlapping scopes in the space of parameters.

9.2.2 Parameter Domain Knowledge for Learning Bayesian Network Structure

Whereas this research considered the task of parameter learning when the structure of the Bayesian Network is known in advance, we believe that Parameter Domain Knowledge can also help perform automatic structure learning of a Bayesian Network. This can prove challenging because when the structure changes, other parameters are involved in the model. It would be very expensive to make the expert specify Parameter Domain Knowledge for each intermediary structure in the search. Ideally one would like the expert to specify a set of Parameter Domain Knowledge assumptions at startup and then guide the search according to these constraints.

Assume the structure search is performed via a hill climbing algorithm using the Bayesian Dirichlet (BD) score. The structure at a given step was obtained from the previous structure by either adding or deleting or inverting an edge such that the marginal likelihood increases. We suggest that the initial Parameter Domain Knowledge constraints can be adapted to the current parameterization of the Bayesian Network by using a simple change of variable. Most of the parameters will be the same in the current and previous structure, except for the ones involving the two variables involved in the transition. Additional research is needed to validate this approach.

9.2.3 Hard versus Soft Domain Knowledge Constraints

The Domain Knowledge constraints that we studied in this thesis are all *hard constraints*, in the sense that they are stated with 100% confidence. However, in real life, even an expert may have a certain amount of doubt about a constraint. We would like to be able to allow the expert to assign confidences to the specified constraints, creating what we call *soft constraints*. For example, such a constraint may state: *"I am 90% confident that parameters a and b have equal values.*" Incorporating soft constraints may prove to be a difficult task because it may require development of probabilistic constrained optimization techniques, which are not readily available as far as we are aware.

9.2.4 Parameter Domain Knowledge for Undirected Graphical Models

Our research so far focused on Bayesian Networks, which are directed graphical models. We conjecture it is much easier to specify domain knowledge about directed models because their parameters are easier to interpret. Undirected graphical models describe the joint probability distribution over a set of variables as the normalized exponential of an energy function. Commonly, this energy function is a quadratic function in the values of the random variables, where a coefficient can be thought of as the strength of relationship between two variables. Multivariate normal distributions and Boltzmann Machines are some examples of such undirected graphical models. It would be interesting to investigate if Parameter Domain Knowledge can help learn undirected graphical models, to the extent that it is intuitive to acquire such knowledge from an expert.

9.2.5 Other Extensions

We would like to study the possibility of deriving closed form parameter estimators for other types of Parameter Domain Knowledge constraints. Immediate candidates include hierarchical versions of sum sharing and ratio sharing (which we expect are similar to the hierarchical variant of parameter sharing), as well as domain knowledge about continuous variables of types other than gaussian. In the case of gaussian variables, more work needs to be done to define proper Constrained Parameter Priors that allow us to derive closed form Maximum Aposteriori estimators. Another direction to investigate would be to compute closed form estimators by taking advantage of inequality constraints involving more than one conditional probability distribution. Finally, we would definitely benefit from algorithms that extend the results in section 3.6 to automatically learn Parameter Domain Knowledge constraints.

Bibliography

[Arf85] G. Arfken (1985). Lagrange Multipliers. *Mathematical Methods for Physicists*, 3rd ed., 945-950. Orlando, FL: Academic Press.

[BF96] Y. Bengio and P. Frasconi (1996). Input/output HMMs for sequence processing. *IEEE Trans. on Neural Networks*, **7(5)**:12311249.

[BFG96] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller (1996). Context-specific independence in Bayesian networks. *Proc. of 12th UAI*, 115-123.

[Bil00] J. Bilmes (2000). Dynamic Bayesian Multinets. *Proceedings of UAI 2000*, 38-45.

[BLN01] O. Bangso, H. Langseth, and T. Nielsen (2001). Structural learning in object oriented domains. *Proc. 14th Intl Florida Articial Intelligence Research Society Conference (FLAIRS-2001)*.

[BNO03] D. Bertsekas, A. Nedic, and A. Ozdaglar (2003). *Convex Analysis and Optimization*. Athena Scientific.

[Bra96] M. Brand (1996). Coupled hidden Markov models for modeling interacting processes. *Technical Report 405*, MIT Lab for Perceptual Computing.

[BV04] S. Boyd and L. Vandenberghe (2004). *Convex Optimization*. Cambridge University Press.

[BW00] O. Bangso and P. Wuillemin (2000). Top-down construction and repetitive structures representationin Bayesian networks. *Proc. 13th Intl Florida Articial Intelligence Research Society Conference (FLAIRS-2000)*.

[CJK99] P. A. Carpenter, M. A. Just, T. A. Keller, W. F. Eddy, and K. R. Thulborn (1999). Time course of fMRI-activation in language and spatial networks during sentence comprehension. *NeuroImage*, **10**:216-224.

[CDL99] R. G. Cowell, A. P. Dawid, S. L. Lauritzen, and D. J. Spiegelhalter (1999). *Probabilistic Networks and Expert Systems*. Springer-Verlag.

[CDS96] R. G. Cowell, A. P. Dawid, and P. Sebastiani (1996). A comparison of sequential learning methods for incomplete data. *Bayesian Statistics* **5**:533-541.

[CG01] J. Cheng and R. Greiner (2001). Learning bayesian belief network classifiers: algorithms and system. *Proceedings of the Canadian Conference on Artificial Intelligence (CSCSI01).*

[CGH94] D. M. Chickering, D. Geiger, and D. Heckerman (1994). Learning Bayesian Networks is NP-Hard. *Technical Report MSR-TR-94-17,* Microsoft Research.

[CH92] G. Cooper and E. Herskovits (1992). A Bayesian method for the induction of probabilistic networks from data. *Machine Learning,* **9**:309-347.

[CHM97] D. M. Chickering, D. Heckerman, and C. Meek (1997). A Bayesian Approach to Learning Bayesian Networks with Local Structure, *Technical Report, MSR-TR-97-07.*

[Coo87] G. F. Cooper (1987). Probabilistic Inference Using Belief Networks is NP-hard. *Technical Report KSL-87-27,* Medical Computer Science Group, Stanford University.

[Cor96] D. Corbit (1996). Numerical Integration: From Trapezoids to RMS. *Dr. Dobb's Journal,* **252**:117-120.

[CT91] T. Cover and J. Thomas (1991). *Elements of Information Theory*. Wiley and Sons.

[Dal99] A. M. Dale (1999). Optimal Experimental Design for Event-Related fMRI. *Human Brain Mapping,* **8**:109-114.

[Dan63] G. B. Danzig (1963). *Linear Programming and Extension*. Princeton University Press.

[Dec96] R. Dechter (1996). Bucket elimination: a unifying framework for probabilistic inference. *Proceedings of 12th Annual Conference on Uncertainty in Artificial Intelligence,* 211-219. Morgan Kaufmann.

[Den91] S. Y. Dennis (1991). On the Hyper-Dirichlet Type 1 and Hyper-Liouville distributions. *Communications in Statistics - Theory and Methods,* **20(12)**:4069-4081.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork (2001). *Pattern Classification,* 2nd edition. Wiley.

[DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royas Statistical Society,* Series B, **39(1)**:1-38.

[DP97] P. Domingos and M. Pazzani (1997). On the Optimality of the Simple Bayesian Classifier under Zero-One Loss. *Machine Learning,* **29**:103-130.

[Edw00] D. Edwards (2000). *Introduction to Graphical Modelling,* 2nd ed. Springer-Verlag.

[FG96a] N. Friedman and M. Goldszmidt (1996). Discretization of continuous attributes while learning Bayesian networks. *Proceedings of ICML 1996*, 157-165.

[FG96b] N. Friedman and M. Goldszmidt (1996). Learning Bayesian Networks with Local Structure, *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence*, 252-262. Morgan Kaufmann Publishers, CA.

[FGG97] N. Friedman, D. Geiger, and M. Goldszmidt (1997). Bayesian network classifiers. *Machine Learning*, **29**:131-161.

[FGK99] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer (1999). Learning Probabilistic Relational Models. *Proc. of 16th IJCAI*, 1300-1307.

[FGL98] N. Friedman, M. Goldszmidt, and T. J. Lee (1998). Bayesian Network Classification with Continuous Attributes: Getting the Best of Both Discretization and Parametric Fitting. *Proceedings of ICML 1998*, 179-187.

[FKP98] N. Friedman, D. Koller, and A. Pfeffer (1998). Structured representation of complex stochastic systems. *Proceedings of AAAI 1998*.

[Fri97] N. Friedman (1997). Learning Bayesian networks in the presence of missing values and hidden variables. *Proceedings of ICML 1997*, 125-133.

[Fri98] N. Friedman (1998). The Bayesian structural EM algorithm. *Proceedings of UAI 1998*, 129-138.

[GG84] S. Geman and D. Geman (1984). Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **6**:721-741.

[GH94] D. Geiger and D. Heckerman (1994). Learning Gaussian Networks. Technical Report MSR-TR-94-10.

[GH96] D. Geiger and D. Heckerman (1996). Knowledge representation and inference in similarity networks and Bayesian multinets. *Artificial Intelligence*, **82**:45-74.

[GH97] D. Geiger and D. Heckerman (1997). A characterization of the Dirichlet distribution through global and local parameter independence. *The Annals of Statistics*, **25**:1344-1369.

[GJ97] Z. Ghahramani and M. Jordan (1997). Factorial hidden Markov models. *Machine Learning*, **29**:245-273.

[GMC99] D. Golinelli, D. Madigan, and G. Consonni (1999). Relaxing the local independence assumption for quantitative learning in acyclic directed graphical models through hierarchical partition models. *Proceedings of Artificial Intelligence and Statistics '99*, 203-208. D. Heckerman and J. Whittacker, eds.

[GRS96] W. Gilks, S. Richardson, and D. Spiegelhalter (1996). *Markov Chain Monte Carlo in Practice*. Chapman and Hall.

[Hec90] D. Heckerman (1990). Probabilistic Similarity Networks. *Technical Report No. STAN-CS-90-1316*. Stanford University Press.

[Hec99] D. Heckerman (1999). A Tutorial on Learning with Bayesian Networks. In M. Jordan (ed.), *Learning in Graphical Models*. Cambridge, MA:MIT Press.

[HGC95] D. Heckerman, D. Geiger, and M. Chickering (1995). Learning Bayesian networks: the combination of knowledge and statistical data. *Machine Learning*, **20(3)**:197-243.

[Hoo04] P. Hooper (2004). Dependent Dirichlet Priors and Optimal Linear Estimators for Belief Net Parameters. *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, 251-259. AUAI Press.

[Jen96] F. V. Jensen (1996). *An introduction to Bayesian Networks*. UCL Press.

[Jen01] F. V. Jensen (2001). *Bayesian Networks and Decision Graphs*. Springer.

[Jor99] M. I. Jordan (1999). *Learning in Graphical Models*. MIT Press.

[Kal60] R. E. Kalman (1960). A New Approach to Linear Filtering and Prediction Problems. *Transactions of the ASME–Journal of Basic Engineering*, Series D, **82**:35-45.

[Kar39] W. Karush (1939). *Minima of Functions of Several Variables with Inequalities as Side Constraints*. MSc Thesis, Department of Mathematics, University of Chicago.

[KP97] D. Koller and A. Pfeffer (1997). Object Oriented Bayesian Networks. *Proc. of 13th UAI*, 302-313.

[KT51] H. W. Kuhn and A. W. Tucker (1951). Nonlinear programming. *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, 481-492. University of California Press.

[Lau96] S. Lauritzen (1996). *Graphical Models*. Oxford.

[LB01] H. Langseth and O. Bangso (2001). Parameter learning in object oriented Bayesian networks. *Annals of Mathematics and Artificial Intelligence*.

[LCT02] Y. Li, C. Campbell, and M. E. Tipping (2002). Bayesian automatic relevance determination algorithms for classifying gene expression data. *Bioinformatics,* **18(10)**:1332-1339.

[LM97] K. Laskey and S. M. Mahoney (1997). Network fragments: Representing knowledge for constructing probabilistic models. *Proceedings of UAI 1997*.

[Mac98] D. MacKay (1998). Introduction to Monte Carlo methods. In *Learning in graphical models*. MIT Press.

[May79] P. S. Maybeck (1979). *Stochastic Models, Estimation, and Control*, Volume 1. Academic Press.

[Min00a] T. Minka (2000). Estimating a Dirichlet distribution. Technical report, MIT.

[Min00b] T. Minka (2000). Bayesian linear regression. Technical report, MIT.

[Min99] T. Minka (1999). The Dirichlet-tree distribution. Paper available online at: http://www.stat.cmu.edu/ minka/papers/dirichlet/minka-dirtree.pdf.

[Mit97] T. Mitchell (1997). *Machine Learning.* McGraw Hill.

[Mit02] T. Mitchell, R. Hutchinson, M. Just, S. Newman, R. S. Niculescu, F. Pereira, and X. Wang (2002). Machine Learning of fMRI Virtual Sensors of Cognitive States. Workshop paper presented at NIPS 2002 workshop on fMRI.

[Mit03] T. Mitchell, R. Hutchinson, M. Just, S. Newman, R. S. Niculescu, F. Pereira, and X. Wang (2003). Classifying Instantaneous Cognitive States from fMRI Data. *Proceedings of the 2003 AMIA Annual Symposium*, 465-469. Mark Musen, MD, PhD, ed.

[Mit04] T. Mitchell, R. Hutchinson, M. Just, S. Newman, R. S. Niculescu, F. Pereira, and X. Wang (2004). Learning to Decode Cognitive States from Brain Images. *Machine Learning Journal* **57 (1-2)**: 145-175.

[MK96] G. J. McLachlan and T. Krishnan (1996). *The EM Algorithm and Extentions*. Wiley.

[Mur02] K. P. Murphy (2002). Dynamic Bayesian Networks: Representation, Inference and Learning. PhD Thesis. UC Berkeley, Computer Science Division.

[Nea93] R. Neal (1993). Probabilistic Inference Using Markov Chain Monte Carlo Methods. *Technical report CRG-TR-91-1*. Univ. Toronto.

[NH98] R. M. Neal and G. E. Hinton (1998). A new view of the EM algorithm that justifies incremental, sparse and other variants. In *Learning in Graphical Models*. MIT Press.

[NMR05] R. S. Niculescu, T. Mitchell, and B. Rao (2005). Parameter Related Domain Knowledge for Learning in Graphical Models. *Proceedings of SIAM Data Mining conference 2005.*

[Pea88] J. Pearl (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

[Pea00] J. Pearl (2000). *Causality*. Cambridge.

[Pfe00] A. Pfeffer (2000). Probabilistic Reasoning for Complex Systems. PhD thesis, Dept. Comp. Sci., Stanford University.

[PLL02] J. M. Pea, J. A. Lozano, and P. Larraaga (2002). Learning Recursive Bayesian Multinets for Data Clustering by Means of Constructive Induction. *Machine Learning*, **47** (1):63-89.

[PTV93] W. H. Press, S. A. Teukolsky, and W. T. Vetterling (1993). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.

[PV91] J. Pearl and T. Verma (1991). A theory of inferred causation. *Knowledge Representation*, 441-452.

[Rab89] R. L. Rabiner (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech recognition. *Proceedings of the IEEE*, **77** (2):257-286.

[RG99] S. Roweis and Z. Ghahramani (1999). A Unifying Review of Linear Gaussian Models. *Neural Computation*, **11(2)**:305-345.

[RR03] R. H. Rao and R. B. Rao (2003). Quality Assurance through Comprehensive Extraction from Existing (non structured) Patient Records. Presentation at HIMMS 2003.

[RSN02] R. B. Rao, S. Sandilya, R. S. Niculescu, C. Germond, and A. Goel (2002). Mining Time-dependent Patient Outcomes from Hospital Patient Records. *Proceedings of the 2002 AMIA Annual Symposium*.

[RSN03] R. B. Rao, S. Sandilya, R. S. Niculescu, C. Germond, and H. Rao (2003). Clinical and Financial Outcomes Analysis with Existing Hospital Patient Records. *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 416-425.

[RN95] S. Russell and P. Norvig (1995). *Artificial Intelligence: A Modern Approach*. Prentice Hall.

[SDH98] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz (1998). A Bayesian Approach to Filtering Junk E-Mail. *AAAI'98 Workshop on Learning for Text Categorization.* Madison, WI.

[SGS00] P. Spirtes, C. Glymour, and R. Scheines (2000). *Causation, Prediction, and Search*, 2nd edition. MIT Press.

[Sha98] R. Shachter (1998). Bayes-ball: The rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams). *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98)*, 480-487. Morgan Kaufmann.

[SL90] D. J. Spiegelhalter and S. L. Lauritzen (1990). Sequential updating of conditional probabilities on directed graphical structures. *Networks* **20**:579-605.

[SPR03] E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman (2003). Learning Module Networks. *Proc. of 19th UAI*, 525-534.

[SS90] G. Shafer and P. P. Shenoy (1990). Probability propagation. *Annals of Mathematics and Artificial Intelligence*, **2**:327-352.

[SSR03] E. Segal, M. Shapira, A. Regev, D. Pe'er, D. Botstein2, D. Koller, and N. Friedman (2003). Module networks: identifying regulatory modules and their condition-specific regulators from gene expression data. *Nature Genetics* **34**:166-176.

[TF00] J. B. Tenenbaum and W. T. Freeman (2000). Separating style and content with bilinear models. *Neural Computation* **12(6)**:1247-1283.

[TI76] D. M. Titterington (1976). Updating a diagnostic system using unconfirmed cases. *Applied Statistics*, **25**:238-247.

[WB95] G. Welch and G. Bishop (1995). An Introduction to the Kalman Filter. *Technical Report TR 95-041*. University of North Carolina.

[Whi90] J. Whittaker (1990). *Graphical Models in Applied Multivariate Statistics*. Wiley.

[Zha98a] N. Zhang (1998). Probabilistic Inference in Influence Diagrams. *Computational Intelligence*, **14(4)**:475-497.

[Zha98b] N. Zhang (1998). Inference in Bayesian Networks: The Role of Context-Specific Independence. *Technical Report HKUST-CS98-09*, Dept. Comp. Sci., Hong Kong University.

[ZL01] C. Zhai and J. Lafferty. A Study of Smoothing Methods for Language Models Applied to ad hoc Information Retrieval. *Proc. of SIGIR 2001*, 334-342.

[ZP96] N. Zhang and D. Poole (1996). Exploiting causal independence in Bayesian network inference. *Journal of AI Research*, 301-328.

[ZP99] N. L. Zhang and D. Poole (1999). On the role of context-specific independence in probabilistic reasoning. *Proceedings of IJCAI 1999*, 1288-1293.

[Zwi03] D. Zwillinger (2003). Lagrange Multipliers. *CRC Standard Mathematical Tables and Formulae*, 31st Ed., 389-390. Boca Raton, FL: CRC Press.