University of Alberta

LEARNING BAYESIAN NETWORKS FROM DATA: STRUCTURE OPTIMIZATION AND PARAMETER ESTIMATION

by

Yuhong Guo 

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Doctor of Philosophy**.

Department of Computing Science

Edmonton, Alberta
Fall 2007

Your file Votre référence
ISBN: 978-0-494-32972-6
Our file Notre référence
ISBN: 978-0-494-32972-6

Abstract

Bayesian networks have become one of the most prevalent and useful formalisms for representing uncertain knowledge, and have been applied to problems that involve both generative data modeling and discriminative pattern classification. The problem of learning Bayesian networks from data embodies two key sub problems: structure optimization—that is, determining the directed acyclic graph defining the model; and parameter estimation—determining the conditional probability distributions to be associated with each variable. This thesis investigates both the challenges of learning structures and parameters from data. The main contributions of this thesis include: (1) a novel convex optimization algorithm for Bayesian network structure learning; (2) a new globally regularized risk minimization approach for gene regulatory network induction; (3) a new discriminative model selection criterion for score-based structure learning of Bayesian network classifiers; (4) a novel maximum margin discriminative parameter estimation algorithm for learning Bayesian network classifiers; and (5) a novel convex optimization algorithm for Bayesian network parameter learning with hidden variables.

To my parents

Acknowledgments

My deepest gratitude goes to my supervisors, Dr. Dale Schuurmans and Dr. Russell Greiner. Without their wise guidance, constant support, unvarying encouragement and assistance, I could not have completed this thesis. I would also like to thank my committee, Dr. Robert Holte, Dr. Peter Hooper and Dr. Kevin Murphy, for spending time reading the thesis and providing helpful comments and suggestions.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

We are living in an age where complex data objects are routinely collected and analyzed. The various components of a complex data object, such as a text or video, are correlated. Even in the real world, observations of different variables are often dependent. For example, if it is raining, the grass will be wet; if there is a fire, a fire alarm will ring. Generally speaking, the relationship between events is usually more complicated than just a a one to one direct causal relationship: the grass is also wet when the sprinkler is on; when the fire alarm is broken, it will not ring even when there is a fire, and so on. Therefore, the dependencies between observation variables are full of uncertainty, and can appear in a probabilistic manner. The question is how to model the uncertain dependence relationships between variables. Furthermore, how can one discover these dependence relationships, or their independence, from what we have: the data. These are critical issues for data analysis and machine learning.

As a combination of probability theory and graph theory, Bayesian networks appear to be a suitable framework for tackling these problems. A Bayesian network is a probabilistic graphical model that encodes a joint probability distribution over a set of random variables using a directed acyclic graph associated with a set of conditional probabilistic parameters. Specifically, the probabilistic dependencies between variables are asserted through the conditional independence assumptions encoded by the directed edges, and numerical uncertainties are specified by the local conditional probability parameters.

As an important tool for knowledge representation and reasoning under uncertainty, Bayesian networks possess several benefits [47]. First, Bayesian networks facilitate the translation of expert knowledge into a probabilistic representation, and can conveniently incorporate prior domain knowledge into the learning process using Bayesian statistical techniques. Second, Bayesian networks provide unique semantic clarity and understand-

ability by humans. In many situations, a Bayesian network can be given a causal interpretation. Users are more likely to gain further insights from Bayesian networks. Third, Bayesian networks allow one to learn and represent cause-effect relationships that are useful for gaining better understanding about a problem domain and its structure. For example, with a Bayesian network, it is easy to determine the relevant features for making classification decisions, and identify the dependences among the various features. Fourth, Bayesian networks can easily handle noisy and incomplete data. In many applications, the data used for learning cannot be fully observed: either feature values are randomly missing from instances or hidden variables exist. Bayesian networks, with their dependence encoded structures, offer a natural way to fill in the missing data or handle the learning issues in such cases.

As a result, Bayesian networks have become one of the most prevalent and useful formalisms for representing uncertain knowledge [71]. They have been widely used for modeling knowledge in bioinformatics, medicine, engineering, image processing and decision support systems. Recently, they have also been used to address discriminative classification problems in these domains.

Although Bayesian networks are useful, they are typically expensive or impractical to obtain from experts for most problem domains, whereas data is often cheap to obtain. The availability of data has led to a growing research interest in learning Bayesian networks from data.

Learning a Bayesian network from data is a challenging problem and has been well studied. The general learning task involves two sub-issues: structure learning—determining a directed acyclic graph over a set of random variables specifying the conditional independence properties; and parameter estimation—determining the conditional probability distributions associated with each variable. Learning Bayesian network structure has been proved to be an NP-hard problem when a consistent scoring criterion is used [16]. Current heuristic algorithms have yet to robustly yield satisfactory solutions in practice, and generally suffer from local optima. Furthermore, new learning challenges for both structure optimization and parameter estimation are posed when Bayesian networks are used for discriminative classification, where the goal is different from that of modeling the underlying joint distribution. Therefore, Bayesian network learning remains a critical research issue and new learning techniques are still needed.

This thesis investigates the problem of learning Bayesian networks, both their structure and parameters, for generative data modeling as well as discriminative pattern classification.

The main contributions of this thesis are the following.

First, I present a novel relaxed convex optimization approach for Bayesian network structure learning from data. The convex approach I propose introduces feature indicator variables to optimize the structure by searching through a space of features. Unlike traditional heuristic search methods that suffer from local optima, the proposed convex approach can converge to an approximated global optima. Moreover, the formulated optimization is polynomial in both the size of the training data and the number of variables, and therefore avoids an NP-hard computational problem for general Bayesian network structure learning.

Second, using the feature selection idea developed above, I present a new globally regularized risk minimization approach for inferring gene regulatory network structure from gene expression profile data. Unlike the typical gene regulatory network identification methods, which either deal with each gene individually or deal with a group of genes using a single prototype, my new approach is able to encourage the genes with similar profiles to share regulators by introducing global feature selection variables, while permitting individual differences by using an L1 regularizer on local weights.

Third, since Bayesian networks have become popular for use as classifiers, I propose two discriminative model selection criteria that are specifically directed at optimizing classification performance. Unlike standard generative model selection criteria, which measure the quality of the structure for modeling the underlying joint distribution, the proposed discriminative criteria measure the generalization classification performance of the structure, consistent with the goal of learning classifiers. A comprehensive empirical comparison between the proposed discriminative criteria and standard generative criteria provides a useful reference for future research on discriminative structure learning.

Fourth, besides discriminative structure learning, I also present a discriminative parameter estimation approach that extends the popular maximum margin criterion to Bayesian networks. The resulting maximum margin Bayesian network learning technique combines the advantages of discriminative SVMs with the ability of Bayesian networks to encode prior causal knowledge. This approach can also be nicely extended to the multiple class variable case, providing for example a discriminative technique for training directed hidden Markov models.

Finally, beyond just considering complete data, I present a novel convex relaxation of a standard form of EM algorithm to address the problem of parameter estimation in the presence of hidden variables. Standard (Viterbi) EM algorithms only converge to a local optima, while any attempt to overcome this shortcoming in the case of hidden variables by naively

convexifying the problem fails, because the symmetry over configurations of hidden variable values leads to vacuous results. To bypass this problem, I formulate the optimization in terms of equivalence relations over the hidden variable values rather than distributions over the values themselves. This approach makes new progress towards achieving global EM approximations.

The remainder of the thesis is structured as follows: Chapter 2 introduces background on learning Bayesian networks from data. Chapter 3 then presents the convex structure learning approach I propose. Chapter 4 presents a globally regularized minimization method to infer the gene regulatory network structure from time series expression data. Chapter 5 proposes new discriminative model selection criteria and provides a comprehensive empirical study of various discriminative and generative criteria for selecting good structures for Bayesian network classifiers. Chapter 6 presents a new maximum margin parameter estimation technique for learning Bayesian network classifiers. Chapter 7 presents a novel convexification of Viterbi EM for learning Bayesian network parameters in the presence of hidden variables. Finally, Chapter 8 concludes the thesis work.

Chapter 2

Background

In this chapter, I introduce the general background for Bayesian networks. I first introduce the Bayesian network representation and its properties in Section 2.1. Then I discuss the learning issues that arise with Bayesian networks in Section 2.2. Finally, Section 2.3 presents alternative parameterizations of Bayesian networks that I exploit later in the thesis.

2.1 Bayesian Networks

A Bayesian network (BN) is a directed probabilistic graphical model that typically has two key components: a graph structure $\mathcal{G}$ and an associated set of parameters $\boldsymbol{\theta}$. The graph structure $\mathcal{G}$ is specified by a directed acyclic graph (DAG) where each node represents a random variable and the directed edges represent statistical dependencies or cause-effect relationships between the variables. Alternatively, the edges could also be interpreted as specifying conditional independence assumptions. A Bayesian network is usually taken as an I-map of the underlying distribution. That is, every conditional independence implied in the graph is satisfied by the underlying distribution [71]. The conditional independence properties specified by directed graphs are defined by the concept of *d-separation*.

Definition 2.1 *Suppose A, B and C are non-intersecting sets of nodes in the graph. We say A is* **d-separated** *from B by C, written $A \perp\!\!\!\perp B | C$, if every path from any node in set A to any node in set B satisfies either one of the following two conditions:*

1. *there is a node R on the path without two incoming directed edges and R is in C,*
2. *there is a node R on the path with two incoming directed edges, but neither R, nor any descendants of R is in C.*

A good way to explain d-separation in a directed graph is through a *reachability* algorithm known as the *Bayes Ball* algorithm; for details see [51, 67].

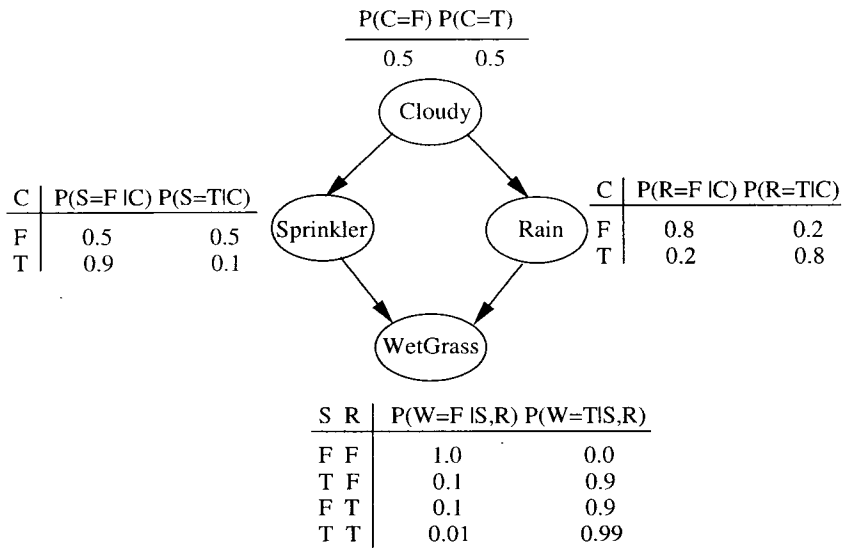


Figure 2.1: A simple Bayesian network adopted from [67]

The conditional independence properties can also be equivalently captured by the Markov properties of directed graphical models [62]. The global Markov property is the same as d-separation; the local Markov property says that a node A is independent of all its non-descendants given its parents; the pairwise Markov property says that a node A is independent of a node B given all its non-descendants and B is not a descendant or parent of A. It can be shown that all these definitions are equivalent [62].

According to the conditional independence properties encoded by the edges, the joint distribution represented by a Bayesian network can be factored into products of local functions, each of which is a local conditional probability associated with a local variable (node). For example, given a Bayesian network that represents a joint distribution over a set of random variables $\mathbf{X} = (X_1, X_2, ..., X_n)$, the joint probability can be compactly represented in a factored form according to the graph structure by

$$P(\mathbf{x}) = \prod_{j=1}^{n} P(x_j|\mathbf{x}_{\pi(j)}) \tag{2.1}$$

where $P(x_j|\mathbf{x}_{\pi(j)})$ is the conditional probability of X_j taking value x_j given its parent configuration $\mathbf{x}_{\pi(j)}$.

The variables in a Bayesian network can be continuous or discrete. This thesis mostly focuses on the discrete case. In this case, the local conditional probabilities associated with each variable can be encoded as entries in a conditional probability table (CPT). These comprise the parameters of a Bayesian network in the standard parameterization. (I will discuss alternative parameterizations below.) For each variable X_j, let a denote the set

of possible values of x_j, and let $\mathbf{b}$ denote the set of values for its parent configuration $\mathbf{x}_{\pi(j)}$. Then the CPT for variable X_j stores the entries $\{\theta_{ja\mathbf{b}}, \forall a, \mathbf{b}\}$, where these parameters $\theta_{ja\mathbf{b}} = P(x_j = a|\mathbf{x}_{\pi(j)} = \mathbf{b})$ must obey the simple nonnegativity constraints $\theta_{ja\mathbf{b}} \geq 0$ for all $a, \mathbf{b}$ and also the *local normalization constraints* $\sum_a \theta_{ja\mathbf{b}} = 1$ for every $\mathbf{b}$.

Figure 2.1 depicts a simple Bayesian network adopted from [67], showing a directed graphical structure and the associated CPTs.

Bayesian networks are one of the most prevalent and useful formalisms for representing uncertain knowledge [71]. Along with Markov networks, they share the advantage of providing a sound probabilistic foundation for inference and learning, and can represent complex distributions compactly. However, Bayesian networks offer a distinct advantage in interpretability, since each parameter can be interpreted in isolation as a conditional probability assertion over a subset of variables in the domain. They also offer computational benefits over Markov networks, by permitting more efficient parameter estimation for example.

2.2 Learning Bayesian Networks

Bayesian networks have been widely used either for generative data modeling or for discriminative data classification. Since manually specifying the complete structure and parameters for a Bayesian network is often either difficult or impossible in practice, learning Bayesian networks from data is an important problem. Learning a Bayesian network from data, therefore, involves two key problems: estimating the parameters of the model, and more interestingly, inferring the structure of the network. I first introduce the parameter learning problem for a given Bayesian network structure, and then discuss the structure learning problem.

2.2.1 Parameter Learning

For a given structure $\mathcal{G}$, traditional parameter learning methods estimate the parameters by maximizing the joint likelihood of data, which is usually referred to as *generative parameter learning* [36]. (I will drop the notation $\mathcal{G}$ for the remainder of this section on parameter learning, since the structure is assumed fixed.) Recently, with Bayesian networks becoming widely used as classifiers, some algorithms have been proposed to learn the parameters by maximizing conditional likelihood of the class variable given the observed evidence variables. These methods are referred to as *discriminative parameter learning* [39, 98, 99].

Generative Parameter Learning

Given a set of training data D, comprised of a set of independent and identically distributed instances $[\mathbf{x}^1; \ldots; \mathbf{x}^N]$, where all components are observed, generative parameter learning methods estimate the parameters of a Bayesian network either by directly maximizing the joint likelihood of training data, or by computing the posterior over parameters $\boldsymbol{\theta}$ given a prior distribution $P(\boldsymbol{\theta})$.

The first method is called maximum likelihood (ML) estimation. Using the conditional independence assumptions encoded in the structure, the joint log likelihood of training data can be factored in the following way

$$\begin{aligned} \log P(D|\boldsymbol{\theta}) &= \sum_{i=1}^{N}\sum_{j=1}^{n} \log P(x_j^i|\mathbf{x}_{\pi(j)}^i, \boldsymbol{\theta}) \\ &= \sum_{i=1}^{N}\sum_{j=1}^{n} \log \left(\prod_{a\mathbf{b}} \theta_{ja\mathbf{b}}^{1_{(x_j^i=a, \mathbf{x}_{\pi(j)}^i=\mathbf{b})}} \right) \\ &= \sum_{ja\mathbf{b}} \#_{ja\mathbf{b}} \log \theta_{ja\mathbf{b}} \end{aligned}$$

where $\#_{ja\mathbf{b}}$ is the count of instances that satisfy the configuration $[x_j = a, \mathbf{x}_{\pi(j)} = \mathbf{b}]$; that is $\#_{ja\mathbf{b}} = \sum_{i=1}^{N} 1_{(x_j^i=a, \mathbf{x}_{\pi(j)}^i=\mathbf{b})}$, where $1_{(\cdot)}$ denotes the indicator function. By maximizing $\log P(D|\boldsymbol{\theta})$ with respect to the local normalization constraints using standard Lagrange multipliers, one can obtain a closed form solution

$$\hat{\theta}_{ja\mathbf{b}} = \frac{\#_{ja\mathbf{b}}}{\#_{j\mathbf{b}}}$$

where $\#_{j\mathbf{b}} = \sum_a \#_{ja\mathbf{b}}$. It is easy to see that these estimated parameters satisfy a parameter independence property: the estimate for $\hat{\theta}_j$ is independent of $\hat{\theta}_{j'}$ for $j \neq j'$; the estimate for $\hat{\theta}_{j\mathbf{b}}$ is independent of the estimate for $\hat{\theta}_{j\mathbf{b}'}$ for $\mathbf{b} \neq \mathbf{b}'$.

The second generative method is called maximum a posteriori estimation (MAP). As mentioned in previous section, one of the benefits of Bayesian networks is the convenience of incorporating prior domain knowledge. Prior knowledge can often be used to effectively avoid overfitting, especially when one has limited training data. Given a prior density $P(\boldsymbol{\theta})$ over parameters $\boldsymbol{\theta}$, one learns the parameters to maximize the posterior

$$\log P(\boldsymbol{\theta}|D) = \log \frac{P(\boldsymbol{\theta})P(D|\boldsymbol{\theta})}{P(D)}$$

This is equivalent to maximizing $\log(P(\boldsymbol{\theta})P(D|\boldsymbol{\theta}))$ since $P(D)$ is invariant with respect to $\boldsymbol{\theta}$.

One usually assumes that the parameters in a Bayesian network satisfy two parameter independence assumptions [48]: the global parameter independence assumption and the local parameter independence assumption. The global parameter independence assumption states that the parameters associated with different variables in a network structure are independent. That is, $\boldsymbol{\theta}_j$ and $\boldsymbol{\theta}_{j'}$ are independent for $j \neq j'$. The local parameter independence assumption asserts that the parameters associated with the same variable but different parent configurations are independent. That is, $\boldsymbol{\theta}_{j\mathbf{b}}$ and $\boldsymbol{\theta}_{j\mathbf{b}'}$ are independent for $\mathbf{b} \neq \mathbf{b}'$. Taken together, these two parameter independence assumptions imply that the prior $P(\boldsymbol{\theta})$ satisfies

$$P(\boldsymbol{\theta}) = \prod_{j\mathbf{b}} P(\boldsymbol{\theta}_{j\mathbf{b}})$$

Typically $P(\boldsymbol{\theta}_{j\mathbf{b}})$ is assumed to have a Dirichlet distribution

$$Dir(\boldsymbol{\theta}_{j\mathbf{b}}|\alpha_{ja=1\mathbf{b}}, ..., \alpha_{ja=|Vals(x_j)|\mathbf{b}})$$

where $\alpha_{ja\mathbf{b}}$ denotes the Dirichlet priors. Thus after observing the training data, the posterior distribution of $\boldsymbol{\theta}_{j\mathbf{b}}$ will still have a Dirichlet distribution

$$P(\boldsymbol{\theta}_{j\mathbf{b}}|D) = Dir(\boldsymbol{\theta}_{j\mathbf{b}}|\alpha_{ja=1\mathbf{b}} + \#_{ja=1\mathbf{b}}, ..., \alpha_{ja=|Vals(x_j)|\mathbf{b}} + \#_{ja=|Vals(x_j)|\mathbf{b}})$$

and one can simply obtain the MAP parameters from this posterior distribution

$$\hat{\theta}_{ja\mathbf{b}} = \frac{\alpha_{ja\mathbf{b}} + \#_{ja\mathbf{b}}}{\alpha_{j\mathbf{b}} + \#_{j\mathbf{b}}}.$$

Note again that the same parameter independence properties hold in the posterior as in the prior. This is a consequence of the parameter independence property holding in both the prior and the maximum likelihood estimation.

An alternative approach to both ML and MAP estimation is the Bayesian learning approach. Instead of obtaining a single set of maximum likelihood parameters from the posterior distribution, one explores all parameter possibilities by just keeping the posterior parameter distribution for later use. That is, one explicitly maintains the posterior

$$P(\boldsymbol{\theta}|D) = \prod_{j=1}^{n} \prod_{\mathbf{b}} \Gamma(\alpha_{j\mathbf{b}} + \#_{j\mathbf{b}}) \prod_{a} \frac{\theta_{ja\mathbf{b}}^{\alpha_{ja\mathbf{b}}+\#_{ja\mathbf{b}}-1}}{\Gamma(\alpha_{ja\mathbf{b}} + \#_{ja\mathbf{b}})}$$

where Γ is the gamma function.

For the simple complete data case, whether using ML or MAP for parameter estimation, one obtains simple closed form solutions as shown above. But when the data is incomplete, that is, the values of some variables are not observed in some instances, the parameter

learning problem becomes much harder because the log likelihood function cannot be factored to estimate the parameters for each variable independently. When a variable is always unobserved in the training data, one refers to it as a hidden variable.

The EM (Expectation Maximization) algorithm [26] is a classical algorithm that is widely used for Bayesian network learning with incomplete data [36, 47, 61]. Let $\mathbf{X}$ be the observed variables and $\mathbf{Y}$ be the hidden variables in a given learning scenario. Then the standard EM algorithm estimates the parameters $\boldsymbol{\theta}$ by maximizing the marginal likelihood of the observed data, $\log P(\mathbf{x}|\boldsymbol{\theta})$. The basic algorithm can be derived as follows. First, note that the marginal log likelihood can be lower bounded in a general way [68]

$$\begin{aligned} l(\boldsymbol{\theta};\mathbf{x}) &= \log P(\mathbf{x}|\boldsymbol{\theta}) \\ &= \log \sum_{\mathbf{y}} P(\mathbf{x},\mathbf{y}|\boldsymbol{\theta}) \\ &= \log \sum_{\mathbf{y}} q(\mathbf{y}|\mathbf{x}) \frac{P(\mathbf{x},\mathbf{y}|\boldsymbol{\theta})}{q(\mathbf{y}|\mathbf{x})} \\ &\geq \sum_{\mathbf{y}} q(\mathbf{y}|\mathbf{x}) \log \frac{P(\mathbf{x},\mathbf{y}|\boldsymbol{\theta})}{q(\mathbf{y}|\mathbf{x})} \text{ (by Jensen's inequality)} \\ &= \mathcal{L}(q,\boldsymbol{\theta}) \end{aligned}$$

for any conditional distribution q. Thus one can conveniently lower bound the marginal log likelihood by a simple expectation (with respect to an arbitrary distribution q) of a function of the complete data log likelihood.

The EM algorithm exploits this lower bound to indirectly maximize $l(\boldsymbol{\theta};\mathbf{x})$ by instead maximizing $\mathcal{L}(q,\boldsymbol{\theta})$ iteratively. EM works by alternately maximizing $\mathcal{L}(q,\boldsymbol{\theta})$ with respect to q and $\boldsymbol{\theta}$ in two alternating steps

$$\begin{aligned} \textbf{E-step:} \quad & q^{(k+1)} = \arg\max_{q} \mathcal{L}(q,\boldsymbol{\theta}^{(k)}) \\ \textbf{M-step:} \quad & \boldsymbol{\theta}^{(k+1)} = \arg\max_{\boldsymbol{\theta}} \mathcal{L}(q^{(k+1)},\boldsymbol{\theta}) \end{aligned} \tag{2.2}$$

The E-step maximizes $\mathcal{L}(q,\boldsymbol{\theta}^{(k)})$ with respect to q, and is called the *expectation step*. It is easy to show that the maximum of the E-step is obtained when $q^{(k+1)}(\mathbf{y}|\mathbf{x}) = P(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}^{(k)})$, which causes $\mathcal{L}(q,\boldsymbol{\theta}^{(k)})$ to be equal to its upper bound $l(\boldsymbol{\theta}^{(k)};\mathbf{x})$

$$l(\boldsymbol{\theta}^{(k)};\mathbf{x}) = \mathcal{L}(q,\boldsymbol{\theta}^{(k)}) = \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x},\boldsymbol{\theta}^{(k)}) \log P(\mathbf{x}|\boldsymbol{\theta}^{(k)})$$

The M-step is called the *maximization step*. It maximizes $\sum_{\mathbf{y}} q^{(k+1)}(\mathbf{y}|\mathbf{x}) \log P(\mathbf{x},\mathbf{y}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$ after the E-step.

The EM algorithm is intuitive and easy to implement. It is very effective for learning directed graphical models with incomplete data. However, it suffers from a major drawback of non-convexity (local maxima). I will address this issue in Chapter 7.

Discriminative Parameter Learning

Since Bayesian networks have widely been used for discriminative classification tasks, it has been realized that the standard generative parameter learning methods are not the best way to train them for classification. The weakness of generative learning for classification problems is that it optimizes a criterion, such as maximum likelihood or MAP, that is not consistent with classification performance [33]. In recent years, researchers considered a new discriminative approach for supervised parameter learning that instead takes conditional likelihood as the optimization criterion [39, 75, 83, 98, 99]. For classification, one variable Y is typically denoted as the class variable and all the other variables $\mathbf{X}$ are considered to be evidence variables. More generally, one can have more than one class variable $\mathbf{Y}$ [1, 19, 88]. Thus the discriminative maximum conditional likelihood (MCL) parameter estimation would train parameters by maximizing the following log likelihood

$$\sum_i \log P(y^i|\mathbf{x}^i, \boldsymbol{\theta})$$

Here I first assume the training data is complete. It is then easy to see the difference between joint likelihood and conditional likelihood by decomposing the joint log likelihood

$$\log P(\mathbf{x}, y|\boldsymbol{\theta}) \;=\; \log P(y|\mathbf{x}, \boldsymbol{\theta}) + \log P(\mathbf{x}|\boldsymbol{\theta})$$

Note that the first term $\log P(y|\mathbf{x}, \boldsymbol{\theta})$ represents the conditional likelihood, while the second term $\log P(\mathbf{x}|\boldsymbol{\theta})$ has nothing to do with classification. The conditional likelihood term can be further expressed as

$$\begin{aligned}
\log P(y|\mathbf{x}, \boldsymbol{\theta}) &= \log \frac{P(\mathbf{x}, y|\boldsymbol{\theta})}{\sum_{y'} P(\mathbf{x}, y'|\boldsymbol{\theta})} \\
&= \log \frac{P(y|\mathbf{x}_{\pi(Y)}, \boldsymbol{\theta}) \prod_{j \in C(Y)} P(x_j|\mathbf{x}_{\pi(j)}, y, \boldsymbol{\theta})}{\sum_{y'} P(y'|\mathbf{x}_{\pi(Y)}, \boldsymbol{\theta}) \prod_{j \in C(Y)} P(x_j|\mathbf{x}_{\pi(j)}, y', \boldsymbol{\theta})}
\end{aligned}$$

where $C(Y)$ represents the indices of the child variables of Y. One can see that the conditional likelihood is only affected by the variables that fall in the Markov Blanket of the class variable where the concept of Markov Blanket is defined as follows.

Definition 2.2 *For a variable Y in a Bayesian network $\mathcal{G}$, its* **Markov Blanket** *is defined as the union of the variables that are either Y's direct parents, or direct children, or share a common child variable with Y according to $\mathcal{G}$.*

In the $\mathcal{G}$-*closed* case [3], where the true generative Bayesian network structure $\mathcal{G}$ is given and the data generating distribution can be represented with this $\mathcal{G}$, maximizing the joint likelihood of data is a consistent method of estimating the joint distribution of variables; that is, the joint distribution given by the maximum likelihood parameters converges to the generating distribution. Thus the conditional distribution of the class variable with the maximum likelihood parameters will also converge to the true conditional distribution. But in the $\mathcal{G}$-*open* case—this is the typical case one encounters in most classification tasks—one does not have the true structure of the underlying generative model. In this case, maximizing the joint distribution with respect to the structure is not consistent with maximizing conditional distribution on the class variable anymore. Moreover, it has been shown [75] that maximizing the conditional likelihood can converge to a better distribution (that has lower expected conditional logloss) than maximizing joint likelihood in this case.

Although generative parameter learning given complete data is quite easy, it is difficult to find the global maximum when using the discriminative conditional likelihood criterion for general Bayesian networks. It has been proved that it is NP-hard to find the parameters for a fixed Bayesian network structure that maximize the conditional likelihood of a given sample of incomplete data [39]. Whether this remains true for complete data is an open problem.

In [39], a gradient ascent method ELR was proposed to optimize the conditional likelihood function for general Bayesian networks. ELR can deal with incomplete data without additional computational cost, since it can fill in the missing values as a by-product of the inference algorithm needed to compute the terms in the gradient. Besides this approach, a discriminative EM algorithm, which maximizes the conditional likelihood of the hidden values in the M step, has also been used addressing the discriminative learning problem with incomplete data [78].

Although conditional likelihood is an effective training criterion for classification problems, it is not the only possible discriminative criterion. Other training criteria might be just as effective and have further advantages. In particular, in Chapter 6 I investigate the use of a large margin criterion for training Bayesian networks for classification.

2.2.2 Structure Learning

In many application domains, the underlying Bayesian network structure $\mathcal{G}$ is not provided and one needs to learn the structure itself from training data. In fact, identifying the underlying cause-effect Bayesian network structures is particularly useful in some fields, such as

biology, medicine, and physics. Therefore, Bayesian network structure learning is an important and relevant research topic. Although parameter estimation from complete data is a generally well understood problem that permits effective algorithmic approaches, structure learning, on the other hand, is a much more challenging problem. For simplicity, I only consider structure learning given complete data in this thesis. Generally speaking, the many proposed structure learning approaches to date fall into one of the two groups: constraint-based (or dependency analysis based) and score-based.

Constraint-based Structure Learning

Since a Bayesian network structure encodes conditional independence assumptions among a set of variables, the network structure can be inferred if one can discover these conditional independence properties from the data. This insight leads to the group of constraint-based learning approaches.

One notable such algorithm is the SGS method [86], which detects the conditional independencies by determining d-separation. It starts from a complete undirected graph, and then tests each pair of variables and removes the edge between them if they are d-separated by a subset of the remaining variables. Finally, the graph is oriented according to the d-separation and acyclicity properties. This is done by first examining all triples of variables A, B and C such that there are edges $A - B$, $B - C$, but not $A - C$. If there is no subset that includes B that can d-separate A and C, then the directionality of $A - B$ and $B - C$ is $A \rightarrow B \leftarrow C$. Then all remaining paths such as $A \rightarrow B - C$ are oriented as $A \rightarrow B \rightarrow C$; edges $A - B$ are oriented as $A \rightarrow B$ if there is a directed path from A to B. The SGS algorithm requires an exponential number of d-separation tests in the number of variables, and is infeasible for large networks. Another algorithm presented in [86], called PC, addresses this problem by removing edges d-separated by size zero subsets first, and then by size one subsets, and so on. Furthermore, the subsets are limited to the variables adjacent to the ordered variables under consideration.

Other constraint-based algorithms perform in a similar fashion, though they might conduct the conditional independence test in different ways. For example, [13, 18] use conditional mutual information tests.

The difficulty with constraint-based approaches is that reliably identifying the conditional independence properties and optimizing the network structure are both challenging problems [64].

Score-based Structure Learning

A much more common approach for structure learning is the score-based method, where one first poses a criterion by which a candidate Bayesian network structure can be evaluated on a given data set, and then conducts a heuristic search for a Bayesian network structure that optimizes the score. Since a complex network structure is always able to fit training data better than a simpler structure, the key issue is to develop a reasonable model selection criterion that appropriately balances model complexity with the goodness of fit to the training data; hence avoiding overfitting.

One well established model selection criterion is *Minimum Description Length* (MDL) [34, 57, 73, 92]. The MDL principle is based on the idea that the best model is the one that minimizes the sum of the description length of the model and the description length of the training data given the model, measured in bits. Specifically, the MDL criterion is usually given by

$$\text{MDL}(\mathcal{G}, D) = -\log P(D|\mathcal{G}, \boldsymbol{\theta}) + \frac{k(\mathcal{G}) \log N}{2} \tag{2.3}$$

where N is the number of training instances, and $\boldsymbol{\theta}$ is the maximum likelihood parameters for the given structure $\mathcal{G}$. Here $k(\mathcal{G})$ denotes the number of free parameters in the candidate Bayesian network defined by $\mathcal{G}$

$$k(\mathcal{G}) = \sum_{j=1}^{n} q_j(r_j - 1) \tag{2.4}$$

where $r_j = |Vals(x_j)|$ is the number of values of the variable X_j, and q_j is the number of parent configurations of the jth variable X_j such that $q_j = \prod_{k \in \pi(j)} |Vals(x_k)|$. The first term in (2.3) is the negative of the loglikelihood of the training data, which approximates the description length of the training data [21, 34]. The second term in (2.3) is the description length of the CPTs, that is, the parameters. (Note that the description length for the graph structure is omitted here since it is much smaller than the description length of the parameters.) According to the MDL principle, one therefore should search for the structure that minimizes (2.3).

Another popular model selection criterion for Bayesian network model selection is a Bayesian scoring metric, developed in [20] that eventually led to the *Bayesian Dirichlet likelihood equivalent* (BDe) score of [48]. The BDe score measures the marginal likelihood of the training data over the parameter distributions for the candidate network structure. It

is written as

$$\mathrm{BDe}(\mathcal{G}, D) = \prod_{j=1}^{n} \prod_{\mathbf{b}} \frac{\Gamma(\alpha_{j\mathbf{b}})}{\Gamma(\alpha_{j\mathbf{b}} + \#_{j\mathbf{b}})} \prod_{a} \frac{\Gamma(\alpha_{ja\mathbf{b}} + \#_{ja\mathbf{b}})}{\alpha_{ja\mathbf{b}}} \tag{2.5}$$

where Γ is the gamma function. In this case, structure learning amounts to searching for the network structure that maximizes this score. Using an asymptotic approximation to Γ in (2.5), one can obtain a criterion known as *Bayesian Information Criterion* (BIC) [81]

$$\mathrm{BIC}(\mathcal{G}, D) = \log P(D|\mathcal{G}, \boldsymbol{\theta}) - \frac{k(\mathcal{G}) \log N}{2} \tag{2.6}$$

which turns out to be exactly the negative of the MDL criterion. In addition to BIC, another approximation of BDe is the *Akaike's Information Criterion* (AIC) [9], which has an even simpler regularization term

$$\mathrm{AIC}(\mathcal{G}, D) = \log P(D|\mathcal{G}, \boldsymbol{\theta}) - k(\mathcal{G}). \tag{2.7}$$

Despite their superficial differences, AIC, MDL/BIC and BDe are all asymptotically equivalent. One advantage these criteria have is that they can be decomposed into separated terms associated with each local variable according to the structure, and therefore are efficiently recomputable given local changes to the structure.

Besides the criteria introduced above, Cross-Validation has also been used for model selection in Bayesian network structure learning [92].

Once a model selection criterion is defined, the learning task reduces to conducting a search in structure space to find the structure that optimizes the criterion. This is, however, an intractable optimization problem. The problem of finding the best Bayesian network among all networks where each variable has at most k parents is NP-complete for $k > 1$, using BDe score [15]. In fact, recently it has been shown that optimizing Bayesian network structure is NP-hard for $k \geq 3$, in the large sample limit, for all consistent scoring criteria, including MDL/BIC, AIC and BDe [16].

Due to the inherent intractability of structure optimization, the literature on Bayesian network structure learning has been dominated by heuristic algorithms for searching the space of individual networks. One popular heuristic search strategy, also one of the simplest, is greedy hill climbing. It starts from the given initial network structure (an empty network or a random network) and then repeatedly applies to the current structure the local operation (adding an edge, deleting an edge and reversing an edge) that leads to the best model selection score. The search procedure terminates when a local optima is reached, that is, no local modification of the current structure improves the model selection score.

As with all other greedy local search approaches, hill climbing can and usually does get stuck in a local optimum. One simple way to escape local optima is to use random restarts. Nevertheless, [48] shows that hill climbing is quite effective for learning Bayesian networks in practice. Besides greedy local search with or without random restarts, heuristic search algorithms for structure learning also include simulated annealing and genetic algorithms [31, 38, 60, 66].

Furthermore, it has been recently observed that searching the space of variable orderings can be more effective than searching the space of network structures [60, 90], since the space of orderings is much smaller. These variable order search approaches exploit the fundamental insight of [10, 20] that, for a fixed variable order, the optimal network (of bounded in-degree) and parameters can be computed in polynomial time (but exponential in the in-degree bound).

In Chapter 3, I explore an alternative structure search approach that is based on formulating convex, continuous relaxations of a standard score-based optimization in the feature space following the MDL principle. Furthermore, in Chapter 5, I explore discriminative structure scoring criteria for learning Bayesian networks for classification.

2.3 Alternative Exponential Representations

Beyond the standard Bayesian network parameterization based on CPTs described above, in this thesis, I will make use of alternative, equally expressive parameterizations that make many of the derivations simpler and allow some new techniques to be developed. These alternative parameterizations can be easily derived as simple transformations of the standard CPT parameters $\boldsymbol{\theta}$.

First, since this thesis focuses on discrete Bayesian networks, a simple observation is that the joint probability in (2.1) can be rewritten in an equivalent exponential form as

$$P(\mathbf{x}) = \exp\Big(\sum_{ja\mathbf{b}} 1_{(\mathbf{x}_j=a,\mathbf{x}_{\pi(j)}=\mathbf{b})} \ln \theta_{ja\mathbf{b}}\Big) \tag{2.8}$$

Then by introducing a set of new variables $\boldsymbol{\omega}$ to replace $\boldsymbol{\theta}$, the joint probability in (2.8) can be further represented in a form of exponential model

$$P(\mathbf{x}) = \exp\Big(\boldsymbol{\omega}^\top \boldsymbol{\phi}(\mathbf{x})\Big) \tag{2.9}$$

using the substitution

$$\omega_{ja\mathbf{b}} = \ln \theta_{ja\mathbf{b}} \tag{2.10}$$

where $\phi(\mathbf{x})$ denotes the feature vector

$$\phi(\mathbf{x}) = (...1_{(\mathbf{x}_j=a,\mathbf{x}_{\pi(j)}=\mathbf{b})}...)^\top \tag{2.11}$$

over $j, a, \mathbf{b}$. The key aspect of this exponential form is that it expresses the joint probability $p(\mathbf{x})$ as a convex function of the parameters ω, which would seem to suggest convenient optimization problems. However, Bayesian networks also require the imposition of additional normalization constraints over each variable

$$\sum_a e^{\omega_{ja\mathbf{b}}} = 1 \text{ for all } j, \mathbf{b} \tag{2.12}$$

Unfortunately, these constraints are nonlinear and thus introduces difficulties for deriving convex techniques. Nevertheless, this new representation using ω still provides a convenient way to derive a novel convex optimization technique for maximum margin parameter learning in Chapter 6 of this thesis.

The parameterization in terms of ω is not the only representation that proves to be useful. A standard reparameterization that removes the normalization constraints in (2.12) is to introduce another set of variables $\mathbf{w}$ to replace $\boldsymbol{\theta}$, instead of using ω, via the definition

$$\theta_{ja\mathbf{b}} = \frac{e^{w_{ja\mathbf{b}}}}{\sum_{a'} e^{w_{ja'\mathbf{b}}}} \tag{2.13}$$

In this way, the local normalizations can be implicitly encoded in the representation. Using the $\mathbf{w}$ parameterization, the joint probability in (2.8) can be re-expressed in a standard exponential form that offers many advantages over the traditional CPT based representation in the later techniques derived in this thesis

$$P(\mathbf{x}) = \exp\left(\sum_j \left[\mathbf{w}_j^\top \phi_j(x_j, \mathbf{x}_{\pi(j)}) - A(\mathbf{w}_j, \mathbf{x}_{\pi(j)})\right]\right) \tag{2.14}$$

where

$$A(\mathbf{w}_j, \mathbf{x}_{\pi(j)}) = \log\left(\sum_a \exp\left(\mathbf{w}_j^\top \phi_j(a, \mathbf{x}_{\pi(j)})\right)\right)$$

Here $A(\mathbf{w}_j, \mathbf{x}_{\pi(j)})$ is the log normalization constant for the jth conditional probability distribution; $\phi_j(x_j, \mathbf{x}_{\pi(j)})$ denotes the feature vector $(...1_{(x_j=a,\mathbf{x}_{\pi(j)}=\mathbf{b})}...)^\top$ over $a, \mathbf{b}$; and $\mathbf{w}_j$ denotes the local parameter vector $(...w_{ja\mathbf{b}}...)^\top$ over $a, \mathbf{b}$. Thus, together ϕ_j and $\mathbf{w}_j$ specify the local conditional probability distribution $P(x_j|\mathbf{x}_{\pi(j)})$ and allow the traditional CPT parameter entries to be efficiently recovered by

$$\theta_{ja\mathbf{b}} = \exp(\mathbf{w}_j^\top \phi_j(a, \mathbf{b}) - A(\mathbf{w}_j, \mathbf{b}))$$

One key aspect of this general exponential form is that it expresses $\log P(\mathbf{x})$ as a *concave* function of the parameters $\mathbf{w}$, which will lead to convenient optimization problems later in this thesis. Another important advantage of the exponential form, however, is that it allows a *sparse* representation of the conditional distributions. That is, one can represent $P(x_j|\mathbf{x}_{\pi(j)})$ given a subset of features from the set of possibilities $\{1_{(x_j=a,\mathbf{x}_{\pi(j)}=\mathbf{b})} : a \in \mathit{Vals}(x_j), \mathbf{b} \in \mathit{Vals}(\mathbf{x}_{\pi(j)})\}$. In general, this allows one to represent $P(x_j|\mathbf{x}_{\pi(j)})$ compactly even if the number of parent variables is large. Such a sparse feature representation of a CPT is similar to exploiting context specific independence [7] or local structure [34]. In fact, these compact representations can be recovered as a special case. The size of a feature based representation for a CPT is never larger than the traditional table based representation, and can be arbitrarily smaller.

In Chapter 3, I show that this feature based representation is particularly advantageous from the perspective of learning a Bayesian network from data, since it nicely reduces the problem of structure learning, largely, to identifying the features used to define the model.

Chapter 3

Convex Structure Learning

3.1 Introduction

Bayesian networks are one of the most prevalent and useful formalisms for representing uncertain knowledge and complex distributions. However, one of the greatest challenges in constructing a Bayesian network representation is determining the graphical structure of the network that specifies the conditional independence assumptions being made about the domain. Effective tools for learning Bayesian network structures from observed data are therefore important, particularly in domains where prior knowledge about conditional independencies is limited. As introduced in Chapter 2, learning the structure of a Bayesian network from data poses a significantly hard computational problem, since one must cope with a combinatorial search over the space of possible structures. The two groups of structure learning approaches (constraint-based and score-based approaches) proposed in the literature have yet to provide a completely satisfactory solution so far.

In this chapter I propose an alternative approach to the problem of learning a Bayesian network model from data. My idea is based on the general exponential representation of Bayesian networks shown in (2.14). Using this feature based representation, one can conveniently formulate the structure learning problem for Bayesian networks as a combinatorial integer optimization problem. My idea is then to follow the strategy from combinatorial optimization, where, when faced with an intractable integer programming problem, one first formulates a convex relaxation that can be solved efficiently, and then rounds the "soft" solution to obtain an approximate "hard" solution to the original problem. In this approach, the problem of learning a Bayesian network from data can be decomposed into three problems: learning a set of features, learning a variable ordering, and learning a corresponding set of parameters. That is, the proposed approach tackles all three subproblems simultaneously. Here, I propose an efficient relaxation of the Bayesian network structure learning

problem that solves for the structural features that determine the graph, the variable ordering that determines the edge orientation, and the model parameters all in a single, compact optimization. In this chapter, first, I show that, given a fixed variable order, the maximum likelihood structure and parameters can be found in polynomial time and space using a sparse exponential family representation, without any restriction on the number of parents for any variable. Second, given a fixed variable order, I show how feature selection based on the minimum description length principle can be addressed simultaneously with parameter optimization. Finally, to optimize the order, I introduce a compact matrix representation of total orderings that allows a semidefinite relaxation. I evaluate the overall technique on natural and synthetic data sets, and find that convex relaxation is a very promising approach to this problem, even though the underlying search problem is inherently discrete. A preliminary version of this work was published in [42].

3.2 Parameter Estimation

Before introducing the new structure learning approach, I first establish some preliminary results that will be needed later. The first and simplest subproblem is estimating the parameters $\mathbf{w}$ given a fixed variable ordering π and feature set ϕ.

Given complete training data $D = [\mathbf{x}^1; \ldots; \mathbf{x}^N]$ and taking the general exponential representation (2.14), the negative loglikelihood loss can be expressed

$$\begin{aligned} L(\mathbf{w}) &= \sum_{i,j} \left[A(\mathbf{w}_j, \mathbf{x}^i_{\pi(j)}) - \mathbf{w}_j^\top \boldsymbol{\phi}_j(x^i_j, \mathbf{x}^i_{\pi(j)}) \right] \\ &= \sum_{j,\mathbf{b}_j} \#_{j\mathbf{b}_j} \left[A(\mathbf{w}_j, \mathbf{b}_j) - \mathbf{w}_j^\top \bar{\boldsymbol{\phi}}_{\mathbf{b}_j} \right] \end{aligned}$$

where $\bar{\boldsymbol{\phi}}_{\mathbf{b}_j} = \sum_{a_j} \frac{\#_{ja_j\mathbf{b}_j}}{\#_{j\mathbf{b}_j}} \boldsymbol{\phi}_j(a_j, \mathbf{b}_j)$, and

$$A(\mathbf{w}_j, \mathbf{b}_j) = \log \sum_a \exp\left(\mathbf{w}_j^\top \boldsymbol{\phi}(a, \mathbf{b}_j) \right) \tag{3.1}$$

Since $A(\mathbf{w}_j, \mathbf{b}_j)$ is a convex function of $\mathbf{w}_j$ [96], this leads to a convex minimization problem for $\mathbf{w}_j$. However, since overfitting is always a concern, it is advantageous to minimize the regularized loglikelihood loss

$$\begin{aligned} \tilde{L}(\mathbf{w}) &= \frac{\beta}{2}\|\mathbf{w}\|^2 + L(\mathbf{w}) \qquad (3.2) \\ &= \sum_j \left(\frac{\beta}{2}\|\mathbf{w}_j\|^2 + \sum_{\mathbf{b}_j} \#_{j\mathbf{b}_j} \left[A(\mathbf{w}_j, \mathbf{b}_j) - \mathbf{w}_j^\top \bar{\boldsymbol{\phi}}_{\mathbf{b}_j} \right] \right) \end{aligned}$$

Here β is a regularization parameter. Note that the weights that minimize $\tilde{L}(\mathbf{w})$ correspond to a MAP estimate of $\mathbf{w}$, with prior $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, \beta I)$.

The objective (3.2) decomposes as an independent sum over j, so the minimization of each individual objective can be considered separately. To reduce the notational burden, denote the jth component of $\tilde{L}(\mathbf{w})$ by

$$\tilde{L}(\mathbf{u}) = \frac{\beta}{2}\|\mathbf{u}\|^2 + \sum_{\mathbf{b}} \#_{\mathbf{b}} \left[A(\mathbf{u}, \mathbf{b}) - \mathbf{u}^\top \bar{\phi}_{\mathbf{b}} \right] \tag{3.3}$$

where $\mathbf{u}$ corresponds to $\mathbf{w}_j$.

Although $\tilde{L}(\mathbf{u})$ is a convex minimization objective, it turns out that to derive the results below I will need to work with the *dual*. The dual is derived by formulating a tight concave lower bound on $\tilde{L}(\mathbf{u})$, which can then be maximized to recover an equivalent result to $\tilde{L}(\mathbf{u})$. First, consider the log normalization function $A(\mathbf{u}, \mathbf{b})$ defined in (3.1). It is known that this is a convex function of $\mathbf{u}$ [96] and furthermore it is closed; that is, its epigraph

$$epi(A(\cdot, \mathbf{b})) = \{(t, \mathbf{u}) : t \geq A(\mathbf{u}, \mathbf{b})\}$$

is a closed set. (I will exploit these properties below.) To derive a dual optimization problem, consider the Fenchel conjugate function of A, given by the definition

$$A^*(\boldsymbol{\mu}_{\mathbf{b}}, \mathbf{b}) = \sup_{\mathbf{u}} \mathbf{u}^\top \boldsymbol{\mu}_{\mathbf{b}} - A(\mathbf{u}, \mathbf{b}) \tag{3.4}$$

By this definition, (3.4), it is clear that A^* is convex, since it is a pointwise supremum of linear function of $\boldsymbol{\mu}_{\mathbf{b}}$ [8, Section 3.2.3]. An additional property I will use below is that any conjugate function is also closed [74, Theorem 12.2]. It is also well known [96, Theorem 2] that

$$A^*(\boldsymbol{\mu}_{\mathbf{b}}, \mathbf{b}) = \begin{cases} -H(P_{\boldsymbol{\mu}_{\mathbf{b}}}) & \text{if } \boldsymbol{\mu}_{\mathbf{b}} \in \mathcal{M}_{\mathbf{b}} \\ \infty & \text{otherwise} \end{cases} \tag{3.5}$$

where

$$\mathcal{M}_{\mathbf{b}} = \{\mu : \exists P \text{ such that } E_P[\phi_{a\mathbf{b}}(\mathbf{x}) | \mathbf{b}] = \mu_{a\mathbf{b}}\}$$

and

$$H(P_{\boldsymbol{\mu}_{\mathbf{b}}}) = -\sum_a P_{\boldsymbol{\mu}_{\mathbf{b}}}(a) \log P_{\boldsymbol{\mu}_{\mathbf{b}}}(a)$$

For a discrete random variable, which I am assuming in this chapter, the set $\mathcal{M}_{\mathbf{b}}$ is a bounded closed set (in fact, a polytope [96, Proposition 7]), hence $\mathcal{M} = \prod_{\mathbf{b}} \mathcal{M}_{\mathbf{b}}$ is also closed and bounded.

The key property that the conjugate function provides is that it establishes a *concave* lower bound. In fact, since $A(\mathbf{u}, \mathbf{b})$ is a closed convex function in $\mathbf{u}$ [6, Theorem 4.2.1] it can be shown that

$$A(\mathbf{u}, \mathbf{b}) = \sup_{\boldsymbol{\mu}_\mathbf{b}} \mathbf{u}^\top \boldsymbol{\mu}_\mathbf{b} - A^*(\boldsymbol{\mu}_\mathbf{b}, \mathbf{b})$$

That is, the conjugate of the conjugate function is the original function. Using this fact, one obtains

$$\begin{aligned} & \inf_{\mathbf{u}} \tilde{L}(\mathbf{u}) \\ = & \inf_{\mathbf{u}} \frac{\beta}{2}\|\mathbf{u}\|^2 + \sum_{\mathbf{b}} \#_\mathbf{b} \left[\sup_{\boldsymbol{\mu}_\mathbf{b} \in \mathcal{M}_\mathbf{b}} \mathbf{u}^\top \boldsymbol{\mu}_\mathbf{b} - A^*(\boldsymbol{\mu}_\mathbf{b}, \mathbf{b}) - \mathbf{u}^\top \bar{\boldsymbol{\phi}}_\mathbf{b} \right] \\ = & \inf_{\mathbf{u}} \sup_{\mu \in \mathcal{M}} G(\mathbf{u}, \boldsymbol{\mu}) \end{aligned} \tag{3.6}$$

where

$$G(\mathbf{u}, \boldsymbol{\mu}) = \frac{\beta}{2}\|\mathbf{u}\|^2 + \sum_{\mathbf{b}} \#_\mathbf{b} \left[-A^*(\boldsymbol{\mu}_\mathbf{b}, \mathbf{b}) - \mathbf{u}^\top (\bar{\boldsymbol{\phi}}_\mathbf{b} - \boldsymbol{\mu}_\mathbf{b}) \right] \tag{3.7}$$

Finally, the joint function G has the strong minmax property, which allows the order of the minimization and maximization to be reversed. This follows from the following general result from convex analysis.

Theorem 3.1 *(Strong Minmax Property) Consider a joint function $f(x, y)$ defined over $x \in X$ and $y \in Y$. Assume (1) $f(\cdot, y)$ is a closed and convex for all $y \in Y$; (2) $f(x, \cdot)$ is closed and concave for all $x \in X$; (3) $\sup_{y \in Y} f(x, y) < \infty$ for all $x \in X$; and (4) $f(\cdot, y)$ has bounded level sets $\{x : f(x, y) \leq t\}$ for all $y \in Y$. Then*

$$\inf_{x \in X} \sup_{y \in Y} f(x, y) = \sup_{y \in Y} \inf_{x \in X} f(x, y)$$

and the solution value is finite.

Proof: This theorem is just a specialization of a standard result in convex analysis; specifically [74, Corollary 37.3.2] and [6, Page 95]. ∎

To apply this result to the current case, one need only verify that G satisfies the hypotheses of the theorem.

Proposition 3.1 *The function G defined in (3.7) satisfies the conditions of Theorem 3.1.*

Proof: $G(\cdot, \boldsymbol{\mu})$ is clearly closed and convex in $\mathbf{u}$ for each $\boldsymbol{\mu} \in \mathcal{M}$; similarly, $G(\mathbf{u}, \cdot)$ is closed and concave in $\boldsymbol{\mu}$ for each $\mathbf{u}$, since A^* is convex and closed; hence G satisfies the first two conditions. To verify the third condition, note that $\sup_{\mu \in \mathcal{M}} G(\mathbf{u}, \boldsymbol{\mu}) < \infty$ for all

$\mathbf{u}$ since $\mathcal{M}$ is a bounded, closed set and $-A^*(\boldsymbol{\mu}_\mathbf{b}, \mathbf{b}) = H(P_{\boldsymbol{\mu}_\mathbf{b}}) < \infty$. Finally, for any $\boldsymbol{\mu} \in \mathcal{M}$, the level sets for $G(\cdot, \boldsymbol{\mu})$ are clearly bounded because of the $\|\mathbf{u}\|^2$ term. ∎

Hence the order of the minimization and maximization in (3.6) can be reversed to yield

$$\begin{aligned} &\inf_{\mathbf{u}} \tilde{L}(\mathbf{u}) \\ &= \sup_{\boldsymbol{\mu} \in \mathcal{M}} \inf_{\mathbf{u}} \frac{\beta}{2}\|\mathbf{u}\|^2 + \sum_{\mathbf{b}} \#_\mathbf{b} \left[-A^*(\boldsymbol{\mu}_\mathbf{b}, \mathbf{b}) - \mathbf{u}^\top(\bar{\boldsymbol{\phi}}_\mathbf{b} - \boldsymbol{\mu}_\mathbf{b})\right] \end{aligned} \tag{3.8}$$

Taking the derivative of the inner objective with respect to $\mathbf{u}$ yields

$$\nabla_\mathbf{u} \;=\; \beta\mathbf{u} - \sum_{\mathbf{b}} \#_\mathbf{b} \left(\bar{\boldsymbol{\phi}}_\mathbf{b} - \boldsymbol{\mu}_\mathbf{b}\right) \;=\; 0$$

so that

$$\mathbf{u}^*(\boldsymbol{\mu}) \;=\; \frac{1}{\beta} \sum_{\mathbf{b}} \#_\mathbf{b} \left(\bar{\boldsymbol{\phi}}_\mathbf{b} - \boldsymbol{\mu}_\mathbf{b}\right) \tag{3.9}$$

and therefore

$$\begin{aligned} &\inf_{\mathbf{u}} \tilde{L}(\mathbf{u}) \\ &= \sup_{\boldsymbol{\mu} \in \mathcal{M}} -\sum_{\mathbf{b}} \#_\mathbf{b} A^*(\boldsymbol{\mu}_\mathbf{b}, \mathbf{b}) - \frac{1}{2\beta} \left\| \sum_{\mathbf{b}} \#_\mathbf{b} \left(\bar{\boldsymbol{\phi}}_\mathbf{b} - \boldsymbol{\mu}_\mathbf{b}\right) \right\|^2 \\ &= \sup_{\boldsymbol{\mu} \in \mathcal{M}} \sum_{\mathbf{b}} \#_\mathbf{b} H(P_{\boldsymbol{\mu}_\mathbf{b}}) - \frac{1}{2\beta} \left\| \sum_{\mathbf{b}} \#_\mathbf{b} \left(\bar{\boldsymbol{\phi}}_\mathbf{b} - \boldsymbol{\mu}_\mathbf{b}\right) \right\|^2 \end{aligned} \tag{3.10}$$

Thus the dual to the minimum regularized loglikelihood loss problem is a regularized concave maximum entropy problem. Given a solution $\boldsymbol{\mu}^*$ to the dual problem (3.10) a corresponding primal solution $\mathbf{u}^*$ can be easily recovered using (3.9).

For implementation, the primal problem is more convenient than the dual because it is unconstrained. In my implementation below I used a Newton method to efficiently solve (3.3). The dual formulation is required to establish my theoretical results below, however.

3.3 Strategy

Of course, the main goal of this chapter is not to perform parameter estimation, but to learn the structure of a Bayesian network model from data. The exponential family representation and maximum entropy framework introduced above offer a new perspective on this problem. Rather than scoring a Bayesian network and performing a discrete search in structure space, my goal will be to formulate a polynomial time approach that addresses each of

the three subproblems—feature generation (and selection), parameter estimation, and variable ordering—in a joint convex optimization that uses reasonable convex relaxations of the discrete subproblems when necessary.

I pursue the following strategy. First, I generate a sufficient set of features that allows one to express any maximum likelihood solution exactly. The first result below shows that in fact this can be achieved in polynomial time and space given a fixed variable ordering. Second, I then select a subset of the generated features using the minimum description length principle [57, 73]. The main result here is that, using the maximum entropy estimation framework developed above, MDL feature selection and parameter optimization can be performed simultaneously in a novel convex relaxation. Finally, I include variable ordering in the framework by extending the previous optimization formulation to also search over variable orders. Thus the third main result is that a search over variable orders can be efficiently encoded by a compact set of semidefinite constraints on a matrix representation of the ordering. Overall, this approach allows one to formulate a relaxed form of the entire Bayesian network learning problem within a polynomial convex optimization framework.

3.4 Feature Generation

The first result is that, given a fixed variable order, a set of features sufficient to represent any maximum likelihood Bayesian network can be found in polynomial time and is polynomially large. This result holds without restriction on the number of parents of any variable. In fact, the result is straightforward, but relies heavily on the sparse feature representation. The key idea is that one can use linear dependence of feature responses on augmented training data to identify key features and eliminate other features from consideration.

First, note that since the conditional probabilities are locally defined and the variable ordering is known, one can solve the feature generation problem for each variable X_j independently. Next, assume that the variable indices are sorted according to the ordering so that the set of possible parents of X_j is $\{X_1, \ldots, X_{j-1}\}$. Let $\sigma(j) = \{1, \ldots, j-1\}$ denote the set of indices for the ancestors of j under the ordering. Then given a set of complete training data (row vectors) represented in a $N \times n$ data matrix, $D = [\mathbf{x}^1; \ldots; \mathbf{x}^N]$, only the first j columns of D are relevant for X_j.

To identify a sufficient set of features, it suffices to consider a locally augmented data matrix where I copy each ancestor configuration, $\mathbf{x}^i_{\sigma(j)}$, $V_j = |\mathit{Vals}(x_j)|$ times and replace x^i_j with each of its possible values. That is, the first $j-1$ columns of D_j are copied V_j

times and stacked, with each copy receiving a different value for X_j in the jth column. Call the resulting matrix $\tilde{D}_j$; so if D_j is $N \times j$ then $\tilde{D}_j$ is $(NV_j) \times j$.

Proposition 3.2 *For any two exponential form representations $\phi_1, \mathbf{w}_1$ and $\phi_2, \mathbf{w}_2$:*
if $\mathbf{w}_1^\top \phi_1(a, \mathbf{x}^i_{\sigma(j)}) = \mathbf{w}_2^\top \phi_2(a, \mathbf{x}^i_{\sigma(j)})$ for all $i = 1, \ldots, N$ and all $a \in Vals(x_j)$,
then $P_1(x^i_j | \mathbf{x}^i_{\sigma(j)}) = P_2(x^i_j | \mathbf{x}^i_{\sigma(j)})$ for all $i = 1, \ldots, N$.

Proof: First note that the assumption implies that

$$\begin{aligned} A_1(\mathbf{w}_1, \mathbf{x}^i_{\sigma(j)}) &= \log \sum_a \exp\left(\mathbf{w}_1^\top \phi_1(a, \mathbf{x}^i_{\sigma(j)})\right) \\ &= \log \sum_a \exp\left(\mathbf{w}_2^\top \phi_2(a, \mathbf{x}^i_{\sigma(j)})\right) \\ &= A_2(\mathbf{w}_2, \mathbf{x}^i_{\sigma(j)}) \text{ for all } i. \end{aligned}$$

Therefore one must also have

$$\begin{aligned} -\log P_1(x^i_j | \mathbf{x}^i_{\sigma(j)}) &= A_1(\mathbf{w}_1, \mathbf{x}^i_{\sigma(j)}) - \mathbf{w}_1^\top \phi_1(a, \mathbf{x}^i_{\sigma(j)}) \\ &= A_2(\mathbf{w}_2, \mathbf{x}^i_{\sigma(j)}) - \mathbf{w}_2^\top \phi_2(a, \mathbf{x}^i_{\sigma(j)}) \\ &= -\log P_2(x^i_j | \mathbf{x}^i_{\sigma(j)}) \end{aligned}$$

∎

Thus if one set of features ϕ_1 spans another set ϕ_2 on the augmented data matrix $\tilde{D}_j$ for each variable X_j, then the optimal maximum likelihood parameter estimate for ϕ_1 on D has to be at least as good as the best maximum likelihood parameter estimate for ϕ_2; that is, $L(\mathbf{w}_1^*, \phi_1, D) \leq L(\mathbf{w}_2^*, \phi_2, D)$.

Of course, there are many possible features to consider. There is a unique feature $\phi_{ja\mathbf{b}}$ corresponding to an indicator function $\phi_{ja\mathbf{b}}(x_j, \mathbf{x}_{\rho(j)}) = 1_{(x_j = a, \mathbf{x}_{\rho(j)} = \mathbf{b})}$ for each particular subset of ancestor variables, $\rho(j) \subseteq \sigma(j)$, and each particular value a for x_j and value $\mathbf{b}$ for $\mathbf{x}_{\rho(j)}$. Nevertheless, it is a trivial observation that the maximum rank of any possible span of the feature response vectors on the augmented training set $\tilde{D}_j$ is bounded by NV_j, since this is the length of each feature response vector on $\tilde{D}_j$. Therefore, there must exist a set of no more than NV_j features that allows the exponential form representation to achieve the maximum likelihood score of *any* Bayesian network on the training data D_j.

To find this set of features in polynomial time I exploit the fact that every compound

feature $\phi_{ja\mathbf{b}}$ can be decomposed as a product of features defined on shorter patterns

$$\begin{aligned}
&\phi_{ja\mathbf{b}}(x_j, \mathbf{x}_{\rho(j)}) \\
&= 1_{(x_j=a, \mathbf{x}_{\rho(j)}=\mathbf{b})} \\
&= 1_{(x_j=a)} 1_{(x_{\rho_1(j)}=b_1)} \cdots 1_{(x_{\rho_k(j)}=b_k)} \\
&= \phi_{ja}(x_j)\phi_{\rho_1(j)b_1}(x_{\rho_1(j)}) \cdots \phi_{\rho_k(j)b_k}(x_{\rho_k(j)})
\end{aligned} \tag{3.11}$$

Naturally one would like to build a span consisting of the shortest possible feature patterns, since this would result in a simpler Bayesian network representation. Define the length of $\phi_{ja\mathbf{b}}$ to be the number of variables in its definition (3.11). Then we have the following proposition.

Proposition 3.3 *If a compound feature $\phi_{ja\mathbf{b}}$ is spanned by a set of shorter features, then $\phi_{ja\mathbf{b}}$ can be eliminated without affecting the maximum likelihood solution.*

Proof: Assume $\phi_{ja\mathbf{b}} = \sum_f w_f \phi_f$ on $\tilde{D}_j$ for some set of shorter feature patterns $f \in F$. Here F denotes the feature set. Then any extended feature that uses $\phi_{ja\mathbf{b}}$ can be spanned by features based on shorter patterns. In particular, if $\phi_{jc\mathbf{d}} = \phi_{ja\mathbf{b}}\phi_{g_1} \cdots \phi_{g_k}$ on $\tilde{D}_j$, then we must also have $\phi_{jc\mathbf{d}} = \sum_f w_f \phi_f \phi_{g_1} \cdots \phi_{g_k}$ on $\tilde{D}_j$, where the feature patterns in the second expansion are strictly shorter than the first. ∎

This leads to a polynomial time algorithm for generating a set of shortest features with maximum span on $\tilde{D}_j$; see Figure 3.1. To establish that this procedure does indeed run in polynomial time, consider the lattice of feature patterns. The lattice is searched from shortest patterns to longest. Once a pattern is pruned, no extension of it will ever be considered (and correctness will be preserved by Proposition 3.3). However, for each increase in rank, at most $\sum_{\ell=1}^{j-1} \mathit{Vals}(x_\ell)$ features are added, while the maximum rank of the feature response matrix is NV_j. Note that the procedure in Figure 3.1 can generate more than NV_j features, but still no more than a polynomial number in total. When extending a current feature, the procedure considers all possible singleton extensions in parallel, rather than sequentially, to mitigate the effects of an arbitrary inspection order. Appendix A provides a simple example to illustrate this generation procedure.

One drawback of this procedure is that it can generate a large number of parents for X_j, even though the representation remains polynomially large. In fact, this feature generation process is guaranteed to overfit the data, in the sense that it yields a representation that can achieve the maximum likelihood of *any* Bayesian network. Clearly, some sort of feature selection process is required to yield a reasonable model, which I next consider.

Feature generation procedure for x_j on augmented $\tilde{D}_j$:
$\Phi^{(0)} = \{\phi_a : a \in \mathit{Vals}(x_j)\}$
for $k = 1, 2, \ldots$ (while rank increased)
  $\Phi^{(k)} \leftarrow \emptyset$
  for each $\phi_f \in \Phi^{(k-1)}$
    $\Psi \leftarrow \{\phi_{b_\ell}\phi_f : \ell \notin \mathit{Vars}(f), b_\ell \in \mathit{Vals}(x_\ell)\}$
    if rank $\left(\bigcup_{\ell \leq k} \Phi^{(\ell)} \cup \Psi\right) >$ rank $\left(\bigcup_{\ell \leq k} \Phi^{(\ell)}\right)$
      $\Phi^{(k)} \leftarrow \Phi^{(k)} \cup \Psi$

Figure 3.1: Feature generation procedure

3.5 Feature Selection

I base the feature selection strategy on the minimum description length principle [34, 57, 73]. Here I continue to assume a fixed variable ordering is given. The idea is to start with a large set of sufficient features $\boldsymbol{\phi} = (\ldots, \phi_{ja\mathbf{b}}, \ldots)^\top$ generated by the procedure outlined above. To perform feature selection in this large set, I augment the exponential representation with feature selection variables $\boldsymbol{\eta}$. That is, for each feature $\phi_{ja\mathbf{b}}$ I establish a corresponding selector variable $\eta_{ja\mathbf{b}} \in \{0, 1\}$, in addition to the corresponding weight $w_{ja\mathbf{b}}$. Let $N_j = \mathrm{diag}(\boldsymbol{\eta}_j)$ be the diagonal matrix of selector values corresponding to variable X_j. One can then write

$$\tilde{L}(\mathbf{w}, \boldsymbol{\eta}) = \sum_j \left(\frac{\beta}{2}\|\mathbf{w}_j\|^2 + \sum_{\mathbf{b}_j} \#_{j\mathbf{b}_j} \left[A(\mathbf{w}_j, N_j, \mathbf{b}_j) - \mathbf{w}_j^\top N_j \bar{\boldsymbol{\phi}}_{\mathbf{b}_j} \right] \right)$$

where

$$A(\mathbf{w}_j, N_j, \mathbf{b}_j) \;=\; \log \sum_a \exp\left(\mathbf{w}_j^\top N_j \boldsymbol{\phi}_j(a, \mathbf{b}_j) \right)$$

Thus if $\eta_{ja\mathbf{b}} = 1$ then the feature $\phi_{ja\mathbf{b}}$ is selected, otherwise it is dropped.

The idea is to solve for the set of features selection variables $\boldsymbol{\eta}$ and parameters $\mathbf{w}$ that minimize the total description length of the data and the Bayesian network model (in exponential form). This can be formulated as a joint optimization

$$\min_{\boldsymbol{\eta} \in \{0,1\}^{|F|}} \mathbf{c}^\top \boldsymbol{\eta} + \frac{\log N}{2} \mathbf{1}^\top \boldsymbol{\eta} + \min_{\mathbf{w}} \tilde{L}(\mathbf{w}, \boldsymbol{\eta}) \tag{3.12}$$

Here the last term is the cost of encoding the training data $D = [\mathbf{x}^1; \ldots; \mathbf{x}^N]$ using the optimal parameters $\mathbf{w}$ for the network structure specified by $\boldsymbol{\eta}$. This uses a standard result from information theory [21, 34] that an optimal code for data D given a model $P(\mathbf{x})$ has length $-\sum_i \log P(\mathbf{x}^i)$. (Although here I alter this principle slightly to use the regularized

loss $\tilde{L}$ rather than the plain loglikelihood loss L. This simplifies the derivation below, but one could set the regularization parameter β to be arbitrarily small to approximate the plain loglikelihood.)

The first term in (3.12) measures the length of the description for an exponential family representation for the Bayesian network structure specified by $\boldsymbol{\eta}$. In particular, for each feature $\phi_{ja\mathbf{b}}$ selected by indicator $\eta_{ja\mathbf{b}}$ the description length cost is fixed to be

$$c_{ja\mathbf{b}} = |\mathbf{b}| \log n + \log |\mathit{Vals}(a)| + \sum_{\ell} \log |\mathit{Vals}(b_\ell)|$$

where the first term is the cost of encoding the list of variables in feature $\phi_{ja\mathbf{b}}$, and the remaining terms are the cost of encoding the specific values for these variables.

The second term in (3.12) is the cost of encoding each weight parameter $w_{ja\mathbf{b}}$, where the precision is chosen in the manner discussed in [34]. The notation $\mathbf{1}$ refers to the vector of all 1s.

Now I would like to solve for the structure $\boldsymbol{\eta}$ and parameters $\mathbf{w}$ that minimize the total description cost (3.12). Unfortunately, this is a combinatorial optimization problem over the discrete vector $\boldsymbol{\eta}$, and even more problematic: even if $\boldsymbol{\eta}$ were relaxed, the MDL objective (3.12) is not jointly convex in $\boldsymbol{\eta}$ and $\mathbf{w}$. Fortunately, the *dual* form of the regularized loss allows one to re-express this problem as a convex minimization over $\boldsymbol{\eta}$, ignoring the integer constraints.

Using the fact that (3.10) is equivalent to (3.6), an equivalent optimization problem to (3.12) can be obtained, but now using maximum entropy instead of loglikelihood loss

$$\min_{\boldsymbol{\eta} \in \{0,1\}^{|F|}} \mathbf{c}^\top \boldsymbol{\eta} + \frac{\log N}{2} \mathbf{1}^\top \boldsymbol{\eta} + \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_j \left(\sum_{\mathbf{b}_j} \#_{j\mathbf{b}_j} H(P_{\boldsymbol{\mu}_{\mathbf{b}_j}}) - \frac{1}{2\beta} \boldsymbol{\delta}_j^\top N_j^\top N_j \boldsymbol{\delta}_j \right) \quad (3.13)$$

where $\boldsymbol{\delta}_j = \sum_{\mathbf{b}_j} \#_{\mathbf{b}_j} (\bar{\boldsymbol{\phi}}_{\mathbf{b}_j} - \boldsymbol{\mu}_{\mathbf{b}_j})$. Note that, thus far, I have assumed that $\boldsymbol{\eta} \in \{0,1\}^{|F|}$, and therefore $N_j^\top N_j = N_j$, since $\eta_{ja\mathbf{b}}^2 = \eta_{ja\mathbf{b}}$. This allows me to rewrite (3.13) as

$$\min_{\boldsymbol{\eta} \in \{0,1\}^{|F|}} g(\boldsymbol{\eta})$$

where

$$g(\boldsymbol{\eta}) = \mathbf{c}^\top \boldsymbol{\eta} + \frac{\log N}{2} \mathbf{1}^\top \boldsymbol{\eta} + \max_{\boldsymbol{\mu} \in \mathcal{M}} \sum_j \left(\sum_{\mathbf{b}_j} \#_{j\mathbf{b}_j} H(P_{\boldsymbol{\mu}_{\mathbf{b}_j}}) - \frac{1}{2\beta} \boldsymbol{\delta}_j^\top N_j \boldsymbol{\delta}_j \right) \quad (3.14)$$

Crucially, $g(\boldsymbol{\eta})$ is a pointwise maximum of *linear* functions of $\boldsymbol{\eta}$, and is therefore convex [8].

Thus, by combining regularized maximum entropy parameter estimation with the description length penalty, I can obtain a natural convex relaxation of the minimum description length principle simply by relaxing the structure indicator variables to be soft indicators in the interval $[0, 1]$. Remarkably, this formulation allows one to simultaneously optimize the (soft) structure and parameters in a polynomial size convex optimization problem.

To solve this problem in practice, I use a quasi-Newton method, BFGS [70] with backtracking line search to efficiently minimize $g(\boldsymbol{\eta})$. BFGS progressively approximates the Hessian matrix by accumulating gradient information $\nabla g(\boldsymbol{\eta})$ at successive $\boldsymbol{\eta}$ points. Fortunately, $g(\boldsymbol{\eta})$ and $\nabla g(\boldsymbol{\eta})$ are both computable by solving the inner concave maximization on $\boldsymbol{\mu}$ (which in fact is equivalent to solving the primal minimization problem on $\mathbf{w}$). In particular, $g(\boldsymbol{\eta})$ is given by (3.14), and

$$\nabla g(\boldsymbol{\eta}) \;=\; \mathbf{c} + \frac{\log N}{2}\mathbf{1} - \frac{1}{2\beta}\sum_j (\boldsymbol{\delta}_j^*)^{\cdot 2}$$

such that $\boldsymbol{\delta}_j^* = \sum_{\mathbf{b}_j} \#_{\mathbf{b}_j} \left(\bar{\boldsymbol{\phi}}_{\mathbf{b}_j} - \boldsymbol{\mu}_{\mathbf{b}_j}^*\right)$ and $\boldsymbol{\mu}_{\mathbf{b}_j}^* = E_{P_{\boldsymbol{\mu}_{\mathbf{b}_j}^*}}\left[\boldsymbol{\phi}_{\mathbf{b}_j}(x)|\mathbf{b}_j\right]$ for the optimal inner solution $\boldsymbol{\mu}^*$ (or $\mathbf{w}^*$). Here, $(\boldsymbol{\delta}_j^*)^{\cdot 2}$ denotes componentwise squaring of $\boldsymbol{\delta}_j^*$.

3.6 Variable Ordering

The final step is to consider variable ordering as part of the optimization process. Once again, one can relax the problem of solving for the optimal ordering, while performing feature selection and parameter optimization simultaneously. The basic approach is as follows. Since no order is given, I first generate features for each variable X_j assuming all other variables are potential parents. Then, as in the previous section, feature selection variables $\boldsymbol{\eta} = (\ldots, \eta_{jf}, \ldots)^\top$ are introduced where η_{jf} corresponds to the feature with pattern f and child variable j. Finally, model complexity is reduced by minimizing the description length criterion.

As before, I begin by assuming the feature selection variables are $\{0, 1\}$ valued. The issue now is that constraints need to be added to the $\boldsymbol{\eta}$ variables to ensure that a valid Bayesian network structure is obtained. For example, since activating a feature ϕ_{jf} for one variable means that the remaining variables in the pattern f must be parents of j, no feature pattern f can be activated for more than one variable j it contains. This can be encoded locally for each feature pattern by the constraints

$$\sum_{j \in f} \eta_{jf} \;\leq\; 1 \quad \text{for all } f \tag{3.15}$$

In fact, the local constraints are simple linear inequalities that pose little additional burden on the optimization. Unfortunately, ensuring consistency locally within a feature pattern f is easy, but ensuring consistency globally *between* feature patterns f and h is more difficult.

To enforce global consistency, I introduce an auxiliary $\{0,1\}^{n\times n}$ matrix S that encodes a total ordering on the variables. In particular, let $S_{ij} = 1$ denote the case that i precedes j in the ordering, and let $S_{ij} = 0$ denotes that i follows j. For a $\{0,1\}$-valued matrix S to encode a total ordering it has to satisfy three properties

$$\begin{aligned} \text{antisymmetric:} \quad & S_{ij} = 1 - S_{ji} \text{ for all } i \neq j \\ \text{transitive:} \quad & S_{ij} + S_{jk} \leq S_{ik} + 1 \text{ for all distinct } i,j,k \\ \text{reflexive:} \quad & S_{ii} = 1 \text{ for all } i \end{aligned} \tag{3.16}$$

(The diagonal of S is not terribly important, but I set it to 1 for convenience.) The feature selection variables can then be forced to respect a global ordering via the constraints

$$\eta_{jf} \leq S_{ij} \quad \text{for all} \quad j,\ f, i \in f,\ i \neq j \tag{3.17}$$

Thus, using $\{0,1\}$-valued variables $\boldsymbol{\eta}$ and S, local and global consistency can be enforced by *linear* constraints. This can further yield an obvious convex formulation for the entire problem by relaxing the integer variables to be continuous

$$\min_{\boldsymbol{\eta}\in[0,1]^{|F|}, S\in[0,1]^{n\times n}} g(\boldsymbol{\eta}) \quad \text{subject to (3.15), (3.16) and (3.17)}$$

One remaining problem with the formulation is that it requires a large, cubic number of constraints in (3.16) to encode the transitivity property. To reduce the space requirement, I exploit the following relationship between total orderings and matrices that encode equivalence relations. Let T denote a strictly upper triangular matrix with all 1s above the main diagonal, and let I denote the identity matrix.

Proposition 3.4 *Consider a $\{0,1\}$-valued, strictly upper triangular matrix U such that $M = I + U + U^\top$ and $N = I + (T - U) + (T - U)^\top$ are both equivalence relations. Then $S = I + U + (T - U)^\top$ must encode a total ordering.*

Proof: Assume M and N are equivalence relation matrices defined by U as above. That is, in addition to reflexivity and symmetry, which are obvious from their construction, M and N are also transitive. The key part of this proof is to show that this implies S is transitive as well: If M and N are transitive, it then follows that for all i,j,k, such that $i \neq j$, $j \neq k$ and $i \neq k$, that $S_{ij} \wedge S_{jk} \Rightarrow S_{ik}$. The argument is shown by cases over the six possible orderings of i,j,k.

Case 1: If $i < j < k$, then $S_{ij} = M_{ij}$, $S_{jk} = M_{jk}$, and $S_{ik} = M_{ik}$. Therefore $S_{ij} \wedge S_{jk} \Rightarrow S_{ik}$ iff $M_{ij} \wedge M_{jk} \Rightarrow M_{ik}$.

Case 2: If $i < k < j$, then $S_{jk} = N_{jk}$, $S_{ki} = N_{ki}$, and $S_{ji} = N_{ji}$. Therefore $S_{ij} \wedge S_{jk} \Rightarrow S_{ik}$ iff $\neg S_{ij} \vee \neg S_{jk} \vee S_{ik}$ iff $S_{ji} \vee \neg S_{jk} \vee \neg S_{ki}$ iff $N_{ji} \vee \neg N_{jk} \vee \neg N_{ki}$ iff $N_{jk} \wedge N_{ki} \Rightarrow N_{ji}$.

Case 3: If $k < i < j$, then $S_{ki} = M_{ki}$, $S_{ij} = M_{ij}$, and $S_{kj} = M_{kj}$. Therefore $S_{ij} \wedge S_{jk} \Rightarrow S_{ik}$ iff $\neg S_{ij} \vee \neg S_{jk} \vee S_{ik}$ iff $\neg S_{ij} \vee S_{kj} \vee \neg S_{ki}$ iff $\neg M_{ij} \vee M_{kj} \vee \neg M_{ki}$ iff $M_{ki} \wedge M_{ij} \Rightarrow M_{kj}$.

Case 4: If $k < j < i$, then $S_{ij} = N_{ij}$, $S_{jk} = N_{jk}$, and $S_{ik} = N_{ik}$. Therefore $S_{ij} \wedge S_{jk} \Rightarrow S_{ik}$ iff $N_{ij} \wedge N_{jk} \Rightarrow N_{ik}$.

Case 5: If $j < k < i$, then $S_{jk} = M_{jk}$, $S_{ki} = M_{ki}$, and $S_{ji} = M_{ji}$. Therefore $S_{ij} \wedge S_{jk} \Rightarrow S_{ik}$ iff $\neg S_{ij} \vee \neg S_{jk} \vee S_{ik}$ iff $S_{ji} \vee \neg S_{jk} \vee \neg S_{ki}$ iff $M_{ji} \vee \neg M_{jk} \vee \neg M_{ki}$ iff $M_{jk} \wedge M_{ki} \Rightarrow M_{ji}$.

Case 6: If $j < i < k$, then $S_{ki} = N_{ki}$, $S_{ij} = N_{ij}$, and $S_{kj} = N_{kj}$. Therefore $S_{ij} \wedge S_{jk} \Rightarrow S_{ik}$ iff $\neg S_{ij} \vee \neg S_{jk} \vee S_{ik}$ iff $\neg S_{ij} \vee S_{kj} \vee \neg S_{ki}$ iff $\neg N_{ij} \vee N_{kj} \vee \neg N_{ki}$ iff $N_{ki} \wedge N_{ij} \Rightarrow N_{kj}$.

Since S satisfies the reflexivity and antisymmetry properties by construction, this ends the proof. ∎

Given this result, it is therefore possible to enforce the total ordering encoded in S by only imposing a quadratic number of constraints:

$$\begin{aligned} & S = I + U + (T - U)^\top \\ & I + U + U^\top = DD^\top \\ & E - U - U^\top = CC^\top \\ & D\mathbf{1} = \mathbf{1}, \quad C\mathbf{1} = \mathbf{1} \end{aligned} \tag{3.18}$$

where D and C are both additional square $\{0,1\}$-valued matrix variables, and E denotes the matrix of all 1s. Unfortunately, the two quadratic constraints in (3.18) are not convex, but they can be approximated by the semidefinite relaxations $I+U+U^\top \succeq DD^\top$ and $E-U-U^\top \succeq CC^\top$, which can further lead to equivalent linear matrix inequality constraints [58, Schur Complement Lemma]

$$\begin{aligned} \begin{pmatrix} I + U + U^\top & D \\ D^\top & I \end{pmatrix} &\succeq 0 \\ \begin{pmatrix} E - U - U^\top & C \\ C^\top & I \end{pmatrix} &\succeq 0 \end{aligned}$$

Here again one can then obtain a convex optimization problem by relaxing the $\{0, 1\}$ valued variables to $[0, 1]$ and using the semidefinite constraints. The implementation only requires a small modification to the BFGS strategy of the previous section, where the quasi-Newton step now needs to respect these additional constraints. To solve the constrained convex optimization problem I used a simple barrier method [8], with a log barrier function for the linear inequality constraints (3.15) and (3.17), plus a log determinant barrier to enforce the semidefinite constraints in (3.18).

The result is the first comprehensive Bayesian network technique I am aware of that solves for an approximate variable ordering, feature set, and optimal parameters in a joint, polynomial, convex optimization. Interestingly, the experimental results below suggest that this approach can produce competitive results.

One final issue to deal with is rounding a "soft" solution produced by the above convex optimization, to produce a variable ordering and a hard set of features that define a proper Bayesian network. I do not as yet have any approximation guarantees for any rounding approach I have developed so far. In my experiments below, I simply used a greedy rounding scheme that successively checks the largest non-integer η variable, determines whether it is possible for it to be set to 1 without violating any consistency constraints, and if so, rounds the variable greedily to 0 or 1 depending on which value yields the smallest value in the MDL objective (3.12) (keeping the current optimal Bayesian network parameters fixed). This is sufficient to yield reasonable results, although I would still like to investigate more sophisticated rounding approaches in the future.

3.7 Experimental Evaluation

To evaluate the proposed Bayesian network structure learning approach, I conducted a set of experiments on both synthetic and real data and compared the results to those obtained by standard greedy heuristic search techniques. To measure performance of the different learning techniques, I measured the logloss (that is, the negative loglikelihood loss) they obtained on held out test data after training. To isolate the effects of the different approximation stages, I conducted two sets of experiments: in the first set the variable ordering was held fixed, while in the second I used the relaxed ordering search of Section 3.6. In each case, for the greedy search algorithms, I considered both BDe and BIC scores.

Fixed order experiments For the fixed order experiments, I first considered three different artificial network structures, shown in Figure 3.2, each instantiated with random

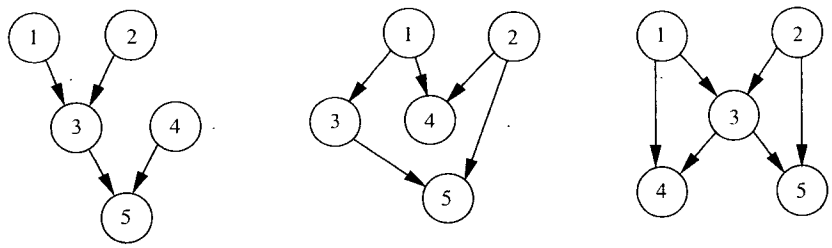


Figure 3.2: Synthetic Bayesian networks 1–3 (from left to right)

Table 3.1: Logloss results for synthetic experiments given the correct variable order

Data Set	Convex	BIC	BDe
Synthetic 1	4.48	4.57	4.53
Synthetic 2	5.34	5.43	5.39
Synthetic 3	5.16	5.27	5.18

CPT entries. For each experiment, I used an independent training and hold-out test samples generated from the synthetic network to determine the regularization parameter for the fixed order convex technique. With the chosen regularization parameter, I then repeatedly generated independent training and test samples and evaluated the logloss on test samples for each Bayesian network learned on the training data using different approaches. Here I compared the results of the convex relaxation algorithm described in Section 3.5 to the fixed order K2 search algorithm of [20], guided with both the standard BDe and BIC criteria. All algorithms were given the correct variable ordering in these synthetic experiments.

Table 3.1 shows the comparison results with training sample size 50 and test sample size 1000. The results are average logloss over 10 repeated runs. One can see that the convex technique outperforms the greedy K2 search procedures, using both the BDe and BIC scores. However, the run time for the convex relaxation procedure (including rounding) was 10, 25 and 30 seconds respectively, while the K2 algorithm only required 0.05 seconds on these problems.

I then conducted experiments on several UCI datasets: Breast, Cleve, Corral, Diabetes, Glass2, Heart, Mofn and Pima. For each dataset, I randomly split the data into training/test partition, and used the training set for Bayesian network learning and test set for performance evaluation. Each algorithm was run with the same fixed variable order, where in this case the order was just chosen randomly. Once again, for the convex relaxation technique, I

Table 3.2: Logloss results for experiments on UCI data sets given a random variable order

Data Set	Convex	BIC	BDe
Breast	5.03	5.47	5.29
Cleve	8.72	8.91	9.00
Corral	4.61	4.67	4.51
Diabetes	5.51	5.62	5.59
Glass2	3.35	3.58	3.40
Heart	8.79	8.89	8.96
Mofn	7.64	7.67	7.84
Pima	5.30	5.35	5.36

Table 3.3: Logloss results for synthetic experiments, comparing methods that learn both structure and order

Data Set	Convex	BIC	BDe
Synthetic 1	4.49	4.57	4.51
Synthetic 2	5.34	5.43	5.35
Synthetic 3	5.16	5.27	5.21

used one preliminary training/test split to set the regularization parameter. Table 3.2 shows the logloss obtained on the test set, training on a disjoint training set of size 50, for each of the learning methods. Still, the results are averages over 10 repeated runs. The results show that the convex approach holds an advantage over the K2 greedy search techniques, for both the standard BDe and BIC scores. However, again, the run times of the convex relaxation approach are greater than the K2 algorithms, requiring from 10-100 times more time to produce the final results.

Although these results are not necessarily comprehensive, they suggest that the ability to avoid local minima in a discrete structure search can lead to good solutions. The major disadvantages of the approach developed so far is that it runs slower than heuristic greedy search and requires regularization parameter to be set, whereas the BDe and BIC scores are parameter free.

Learning variable ordering Next, I repeated the previous experiments using the combined structure and order optimization algorithm of Section 3.6. Here I compared the approach to a standard score-based greedy heuristic search that uses the standard edge addi-

Table 3.4: Logloss results on UCI data sets, comparing methods that learn both structure and order

Data Set	Convex	BIC	BDe
Breast	5.26	5.23	5.25
Cleve	8.60	8.84	9.13
Corral	4.71	4.56	4.54
Diabetes	5.58	5.61	5.62
Glass2	3.49	3.58	3.40
Heart	8.59	8.87	9.10
Mofn	7.57	7.71	8.01
Pima	5.38	5.34	5.39

tion, deletion and reversal operators. Again I considered both the BDe and BIC criteria in the heuristic search. Each greedy search was started from an empty network and restarted 4 times when reaching a local minimum, by randomly adding and deleting edges. However, other than not imposing a variable ordering, the algorithms were run exactly as described above for the fixed order case.

Table 3.3 shows the results obtained by the convex relaxation technique versus the greedy search algorithms on the synthetic problems. The convex approach shows a modest improvement over the greedy search methods. However, once again, the convex relaxation procedure runs about 100 times slower. Interestingly, the solution quality is close to the fixed order case. Only a slight benefit was achieved from having the correct variable ordering.

Table 3.4 compares the results on the UCI datasets. Here the quality of the outcome is mixed. The convex relaxation procedure obtains the best solution quality on 4 out of 8 data sets, while the greedy heuristic search using BDe obtains the best results on 2 out of 8, and BIC obtains the best results on 2 out of 8. More interestingly, comparing these results to the fixed order technique, which just uses a random variable ordering, shows that the fixed order approach (with convex relaxation) still obtains the best results on 4 out of 8 data sets. This outcome seems to suggest that the relaxed ordering constraints imposed in Section 3.6 might not be sufficiently tight to ensure a good solution. Improving the quality of the relaxed ordering constraints remains an important question for future research.

3.8 Conclusion

In this chapter, I have presented what I feel is an interesting new approach to learning Bayesian network structure from data. I first presented a relaxed convex structure learning approach for given fixed variable order, and then extended it to a general technique that simultaneously searches for variable order, parameter settings, and features in a joint convex optimization. The convex approach in either the fixed variable order or order learning case showed promising experimental results on both synthetic networks and real UCI datasets, in comparison with standard score-based greedy heuristic search methods. I feel that the new technique introduced here might open the way to a new class of algorithms for learning Bayesian networks. An interesting question is whether this class of global optimization techniques is able to achieve guaranteed approximation quality.

Chapter 4

Gene Regulatory Network Induction

4.1 Introduction

Learning the structure of a gene regulatory network from time-series gene expression data is a significant challenge. It requires the identification of the cause-effect relationships between genes. Most approaches proposed in the literature to date attempt to predict the regulators of each target gene individually, but fail to share regulatory information between related genes. In this chapter, I propose a new globally regularized risk minimization approach to address this gene regulatory network induction problem. This new approach is motivated by the convex Bayesian network structure learning technique presented in Chapter 3. The idea is to introduce global feature selection variables to select common regulators for a group of genes with similar expression profiles. Here the feature selection variable controlled structure learning approach is extended to address continuous data by using a linear regression framework.

Specifically, I propose to first cluster genes according to their time-series expression profiles—identifying related groups of genes. Then I use the globally regularized risk minimization technique to identify the regulation structure for each gene, while encouraging the genes in the same cluster to share common regulators by exploiting the assumption that genes with similar expression patterns are likely to be co-regulated. The experimental results suggest that the proposed approach is more effective at identifying important transcription factor based regulatory mechanisms than the standard independent approach and a prototype based approach.

This chapter is organized as follows. First, I introduce the background knowledge and related work for gene regulatory network learning in Section 4.2 . Section 4.3 then presents the proposed globally regularized risk minimization approach and the preprocessing details. Experiments and results on both synthetic data and real cell cycle yeast gene expression data

are presented in Section 4.4 . This work was published in [44].

4.2 Background

Genes and their products do not work independently in the cell. Rather, they are jointly regulated in a coordinated fashion, both internally and externally, to achieve proper cell function. One of the key mechanisms of gene regulation takes place at the mRNA transcription level. With the emergence of high-throughput microarray techniques, the mRNA expression levels of thousands of genes can be measured simultaneously. Using computational techniques to learn gene regulatory networks from high-throughput time-series gene expression data has been an active area of research in recent years. The goal of such research is to discover the causal control relationships between genes, which would offer a fundamental understanding of how biological processes are coordinated in the cell.

A variety of computational approaches have been proposed in the literature to model gene regulatory networks from expression data. Many approaches have been based on the use of linear models to express dependence between time series profiles. For example, [27] studied a straightforward linear model for this purpose; [11] and [25] investigated linear differential equations for gene regulatory network modeling. All of these approaches suffer from risks of over-fitting, however, since they fit a number of parameters that is proportional to the size of the data itself. To counter the risk of over-fitting, other linear approaches have taken advantage of sparseness of the regulatory relationship between genes; that is, that any one gene is regulated by a small subset of the other genes. [24] have proposed to use Akaike's Information Criterion (AIC) to determine the nonzero coefficients in the linear system. Similarly, [63] used L1 regularization to conduct feature selection on the linear parent set.

Another popular approach to learning gene regulatory network structure is to exploit various forms of standard (dynamic) Bayesian network structure learning methods, since Bayesian networks can encode cause-effect relationships among a set of variables. *Dynamic* Bayesian networks in particular are a natural extension of Bayesian networks to modeling time-series data. As I have introduced in Chapter 2, learning the structure of a Bayesian network from data generally requires one of two approaches to be followed: a score-based approach—where a heuristic search is performed through the space of causal network structures to identify the most likely structure explaining the data—and a constraint-based approach—where conditional independence tests are used to determine whether a direct

causal relationship should be postulated between two variables. Many variants of these techniques have been applied to gene regulatory network learning, including search-based approaches [46, 97, 100], information-theoretic approaches [12], parameter-tying based approaches [82], and conventional dynamic Bayesian network learning approaches [2, 101].

Although these previous techniques have achieved some promising results, the fundamental limitation of the amount of data available relative to the large number of parameters estimated (e.g. distinct parameters used to predict the expression level of each gene given other genes) severely constrains their effectiveness. This difficulty is inherent to the task: orders of magnitude more expression data would be required for naive estimation approaches devoid of background knowledge and biologically relevant assumptions to succeed on this problem.

One common shortcoming in the current literature, whether using linear modeling or Bayesian network structure learning, is that nearly all proposed approaches attempt to determine the regulation structure for each target gene independently. Yet it is well known that genes that share the same expression pattern are likely to be involved in the same regulatory process, and therefore share the same (or at least a similar) set of regulators [28]. The main question I investigate is how to exploit biologically significant knowledge about co-regulation to improve the inference of the underlying gene regulatory network from expression data. Although a few previous investigators, such as [94], have proposed to group genes with similar expression profiles in a single prototypical "gene", and then model the relations between prototypical genes instead of modeling the genes individually, this is a somewhat oversimplified approach that ultimately ignores the individual differences between genes in the same group, and puts a particular high requirement on the clustering step.

Instead, in this chapter, I propose a novel approach for predicting the regulators for a given group of genes with similar mRNA expression patterns, by minimizing a globally shared regularized prediction risk that encourages similar genes to share regulators. The models I learn, however, are otherwise standard linear models. The novelty of the approach is to first cluster the genes based on their time series expression profiles, and then minimize a loss determined on a set of global indicator variables associated with the common set of possible regulatory variables. The approach does not learn an identical regulation structure for the genes in each cluster, but does encourage them to adopt similar structures. I evaluate the performance of this approach on both synthetic data and the cell cycle time-series gene expression data of [17]. My synthetic results show that this approach is able to learn the

correct structure far more effectively than standard approaches that do not take into account co-regulation knowledge. My results on the cell cycle data of Cho et al. [17] suggests this approach can identify the important transcription factors in the cell cycle genes more accurately by exploiting the co-regulation knowledge.

4.3 Method

The core of the proposed method is based on using linear regression to infer the expression level of each target gene from the expression levels of a set of potential regulator genes. However, even though linear prediction provides a simple and elegant foundation for modeling time series expression data, it cannot be applied naively. At least three significant issues need to be addressed before reasonable results can be achieved in this domain. First, time lags exist in the regulatory pathways controlling gene expression. These time lags vary between pathways and remain generally unknown *a priori* [101]. Second, the number of parameters required by a simple linear model (one parameter for each target-regulator combination) is far too many to be estimated reliably from available time series gene expression data. Some sort of effective feature selection mechanism must be employed [63]. Third, genes that serve related or synchronized functions tend to share common regulatory mechanisms. That is, related genes tend to share common regulators, and this knowledge must be exploited somehow to improve the quality of the regulation networks that are inferred. Failure to take into account any of these issues causes the linear prediction (or any other) approach to perform poorly.

I take into account all three of the above issues and modify the linear prediction approach to infer gene regulatory networks from time series expression data. The first two issues have been handled in varying ways in existing research—although I propose particularly simple ways to handle them in this chapter. The third issue comprises the main observation I make, and motivates the use of a novel form of global risk minimization that is able to share regulatory information between similar genes while simultaneously allowing individual differences.

4.3.1 Linear Modeling

First, to establish the basic linear prediction approach consider an $n \times t$ matrix Y of time series gene expression data, where each column corresponds to the expression levels of a single gene measured over a series of n time points; hence, Y stores the expression profiles for t genes. For each gene, I would like to identify which other genes measured in Y

are likely to be regulators. The fundamental hypothesis is that the expression levels of a regulator gene should be predictive of the expression levels for a regulated target gene, possibly subject to time lag and the presence of co-regulators or absence of inhibitors. To mitigate the effect of measurement errors and outliers in the expression data, I generally assume the columns of Y have been rescaled to values between 0 and 1, and thus I am only searching for explanations of *relative* increases or decreases in expression level.

A straightforward linear prediction approach proceeds as follows. Assume for a target expression profile $\mathbf{y}_j$ given by an $n \times 1$ column vector from Y, one has a set of candidate regulator profiles stored in an $n \times k$ matrix X_j consisting of k distinct columns selected from Y. (I will discuss below how such a set of candidate profiles might be inferred for a given target $\mathbf{y}_j$.) The quality of this set of candidate regulators can be assessed by how well their expression levels predict the expression levels of the target, which can be determined by solving for the combination weights of the regulator profiles that best reconstruct the target profile

$$\min_{\mathbf{w}_j} \quad \|X_j \mathbf{w}_j - \mathbf{y}_j\|_2^2 \tag{4.1}$$

Here the $k \times 1$ vector of combination weights $\mathbf{w}_j$ describes how the expression levels of the regulator genes in X_j interact to best explain the target expression levels $\mathbf{y}_j$, and the quality of the fit can be assessed by the residual error in (4.1).

4.3.2 Coping with Time Lags via Time Shifting

Unfortunately, the naive linear modeling approach (4.1) suffers from the three major drawbacks mentioned above. The first problem is that it does not account for any time lag between the expression of a regulating gene and the expression of its downstream target. In fact, the naive approach (4.1) implicitly assumes that regulation occurs instantaneously, and therefore performs quite poorly at identifying any regulatory relationship that exhibits delayed effects. To cope with this shortcoming, I modify the approach to first take into account any potential time lag between the expression of a regulator and its downstream target. In particular, for each candidate regulator measured in X_j, given by an $n \times 1$ vector $\mathbf{x}_{ij}$, I first compute an optimal shift back in time that best aligns $\mathbf{x}_{ij}$ individually with the target $\mathbf{y}_j$

$$s_{ij}^* = \arg \min_{s \in \{0,1,2,3\}} \|\mathbf{x}_{ij}(1, ..., n-s) - \mathbf{y}_j(s+1, ..., n)\|_2^2 \tag{4.2}$$

(Note that the shifts only allow time lags forward in time from the expression of the regulator to the expression of the target.) Repeating this for each candidate regulator profile in

X_j, yields a series of optimal time lags. I can then reformulate the expression matrix X_j for the candidate regulators by applying the optimal shift to each column, and truncating the columns to a common length based on the maximum shift, obtaining an $(n - s_{max}) \times k$ time-lag aligned matrix Φ_j. The target expression profile $\mathbf{y}_j$ is then also truncated to a corresponding $(n - s_{max}) \times 1$ vector $\tilde{\mathbf{y}_j}$, where $\tilde{\mathbf{y}_j} = \mathbf{y}_j(s_{max}, ..., n)$. The quality of the candidate regulators can then be assessed by the more appropriate aligned reconstruction

$$\min_{\mathbf{w}_j} \quad \|\Phi_j \mathbf{w}_j - \tilde{\mathbf{y}_j}\|_2^2 \tag{4.3}$$

4.3.3 Feature Selection via L1 Regularized Risk Minimization

Although the modified linear approach (4.3) appropriately handles time lags between regulator and target expression patterns, it still suffers from a major drawback: the set of candidate regulators for a given gene is usually very large (e.g. the complete set of remaining genes), while the number of time points sampled in a time series experiment is usually quite small (on the order of 20 to 30). Therefore a large set of combination weights $\mathbf{w}_j$ need to be inferred from a limited amount of data. Moreover, only a tiny fraction of the candidate regulators are expected to be true regulators for any given gene, meaning that, ideally, most of the weights should be set to 0 to indicate non-regulation. The bottom line is that some sort of effective form of *feature selection* is required for this problem. From a large set of candidate regulator expression profiles, most need to be discarded, and a small number retained to provide a good explanation of the target expression profile.

It is well known in the machine learning literature [69] that regularizing with the L1 norm, rather than the more conventional L2 norm, is very effective for feature selection. In this approach, one adds a penalty to the risk (the reconstruction objective) which encourages small values for $\mathbf{w}_j$:

$$\min_{\mathbf{w}_j} \quad \|\Phi_j \mathbf{w}_j - \tilde{\mathbf{y}_j}\|_2^2 + \alpha \|\mathbf{w}_j\|_1 \tag{4.4}$$

where α is a parameter that trades off the influence of the risk with the regularizer. Crucially, this regularizer encourages many of the weights to become exactly zero in the solution. To see why, note that the regularization term is non-differentiable at zero, but any movement of a weight from zero immediately creates a derivative of magnitude α encouraging movement back to zero. Thus, if the magnitude of the derivative of the risk is not greater than α, then the weight will remain at zero. These intuitions lead to an efficient optimization procedure known as grafting [85].

4.3.4 Regulation Sharing via Globally Regularized Risk Minimization

Simply solving the minimization problem in (4.4) provides no advantage over the approaches proposed in the literature however, since it does not address the problem of facing a shortage of data while trying to make inferences about a large number of genes. To mitigate this problem I propose to share regulatory information across sets of target genes. Given the hypothesis that genes with similar expression patterns are usually co-regulated and involved in the same functional process, I propose to first cluster the target genes based on their expression patterns. (This clustering can be performed in many different ways. In my implementation below I simply used a straightforward K-means method based on squared Euclidean distances.) Then, for each cluster, the goal is to identify a set of regulators that is shared among the entire set of genes in the cluster, while still allowing for differences among the regulation of individual genes. Achieving this type of information sharing in the context of regularized linear modeling (4.4) however, requires some novel technical developments.

In Chapter 3, I developed a novel convex Bayesian network structure learning approach based on introducing a set of auxiliary indicator variables to control global feature selection. Adapting this idea to the current context, I propose to use a global regularization scheme on auxiliary global feature selection variables to help identify the common candidate regulators among a group of target genes with similar expression profiles. Given that there is much more data available for sets of similar genes, as opposed to individual genes, the hope is that the common regulators can be more accurately identified.

Specifically, given a set of target genes $Y = \{\mathbf{y}_1, ..., \mathbf{y}_m\}$, I would like to identify a common set of regulators from the set of candidates $X = \{\mathbf{x}_1, ..., \mathbf{x}_l\}$. Define a set of indicator variables $\boldsymbol{\eta} = \{\eta_1, ..., \eta_l\}^\top$, corresponding to the candidate set $X = \{\mathbf{x}_1, ..., \mathbf{x}_l\}$, such that each $\eta_i \in \{0, 1\}$ indicates whether a regulator X_i is selected as an active regulator. Let $N = \text{diag}(\boldsymbol{\eta})$. Then, one can form a globally regularized version of the minimization problem (4.4) by introducing the feature selection variables $\boldsymbol{\eta}$ and adding a new global regularization term on these variables

$$\min_{\boldsymbol{\eta} \in \{0,1\}^l} \min_{\mathbf{w}} \sum_j \left(\|\Phi N \mathbf{w}_j - \tilde{\mathbf{y}}_j\|_2^2 + \alpha \|\mathbf{w}_j\|_1 \right) + \lambda \mathbf{u}^\top \boldsymbol{\eta} \tag{4.5}$$

where $\mathbf{u}$ is a positive weight vector that allows one to incorporate prior knowledge about the importance of each global feature. Although I simply set this vector to $\mathbf{1}$ in my later experiments, it will be very useful to set $\mathbf{u}$ corresponding to prior knowledges whenever they are available. Note that the global regularization term $\lambda \mathbf{u}^\top \boldsymbol{\eta}$ is in fact an L0 norm regularizer

that will automatically force a sparse solution that selects only a small set of global features for the set of target genes in a cluster. Nevertheless, the local L1 norm regularizer, $\alpha\|\mathbf{w}_j\|_1$, will still make individual choices of regulators for each specific target gene; choosing these regulators from the globally selected features identified by $\boldsymbol{\eta}$. Therefore, if the target genes in a cluster share some common regulators, the global feature selection process will be very helpful to pick them out, while the ability to individually model the regulation of each gene has not been diminished.

4.3.5 Optimization Procedure

Equation (4.5) encodes a *min-min* integer optimization problem. Unfortunately, integer optimization problems of this form are generally NP-hard. To attempt to solve the problem efficiently, I first relax it into an optimization over continuous variables, by relaxing each $\eta_i \in \{0, 1\}$ to be continuous $\eta_i \in [0, 1]$. This leads to solve the following relaxed *min-min* optimization:

$$\begin{aligned} \min_{\boldsymbol{\eta}} \min_{\mathbf{w}} \quad & \sum_j \left(\|\Phi N \mathbf{w}_j - \tilde{\mathbf{y}}_j\|_2^2 + \alpha\|\mathbf{w}_j\|_1\right) + \lambda \mathbf{u}^\top \boldsymbol{\eta} \\ \text{subject to} \quad & 0 \leq \boldsymbol{\eta} \leq 1 \end{aligned} \tag{4.6}$$

In fact, this formulation has relaxed the original L0 norm regularizer over $\boldsymbol{\eta}$ into a L1 norm regularizer. In this way I maintain feature selection ability, while gaining computational efficiency.

However, this relaxed optimization problem (4.6) is still non-convex, since the objective function is not jointly convex in both $\mathbf{w}$ and $\boldsymbol{\eta}$. In the implementation below, I conduct the optimization in two alternating steps to obtain a local optimal solution: minimization over $\mathbf{w}$, and minimization over $\boldsymbol{\eta}$. Each $\mathbf{w}$ minimization step is simply a least squares regression with L1 norm regularization, which can be implemented as a quadratic program [8], or by using a fast grafting algorithm [85]. For the $\boldsymbol{\eta}$ minimization step, I use a quasi-Newton BFGS method to perform the optimization [4].

4.4 Experiments and Results

To evaluate the proposed approach, I conducted experiments on both synthetic and real cell cycle data. In particular, I compared the proposed global regularization approach to the standard independent local prediction approach, and a prototype based linear regression method adapted from [94]. The adapted prototype based method first clusters the genes based on

their expression profiles. (In the experiments below, I use the same clustering results for both the proposed global regularization approach and the prototype based approach.) Then, for each cluster, it identifies the common regulators for the genes by treating them as one target gene

$$\min_{\mathbf{w}} \quad \sum_{j} \|\Phi\mathbf{w} - \tilde{\mathbf{y}}_j\|_2^2 + \alpha\|\mathbf{w}\|_1 \tag{4.7}$$

Synthetic experiments are useful to gauge the potential effectiveness of the approach under controlled conditions where the ground truth is available. Once the intuitive behavior of the technique is understood, I then apply the method to inferring the structure of the regulatory network of the yeast cell cycle.

In my experiments, I assume all transcription regulations work through activators, instead of inhibitors; that is, I assume the $\mathbf{w}$ parameters are nonnegative in the linear regressions. Also, to keep the $\mathbf{w}$ parameters from becoming too small and causing a threshold selection problem, I included an additional constraint $\|\mathbf{w}_j\|_1 \geq 1$ in all the three linear regression algorithms.

4.4.1 Experiments on Synthetic Data

For the synthetic experiments, I set up a small system to simulate a cell cycle process controlled by a small number of critical transcription factors (TFs). I defined 10 TFs that regulated the expression levels of 212 genes in 4 phases of a synthetic cell cycle. These 10 TFs were divided into 4 regulatory groups, with 3, 2, 3, and 2 TFs in each group respectively. Each group of TFs was associated with a specific phase of the cell cycle, and regulated the expressions of 53 genes, as well as the TFs in the next phase of the cycle. In my simulation setting, I assumed that one gene (including the TFs themselves) can be regulated by either one TF or a combination of two TFs. I generated the expression data by first simulating ideal expression levels for the TFs in a selected phase for two complete cell cycles, totaling 16 time steps. Then I generated the expression profile for each gene (or TF) in the next phase by a 2 time step delayed response from the combination ("and") of m ($m \leq 2$) randomly selected TFs in the current phase, plus Gaussian noise. Repeating this procedure for all the phases in the cycle in turn, I generated synthetic time-series profiles for the entire set of TFs and genes.

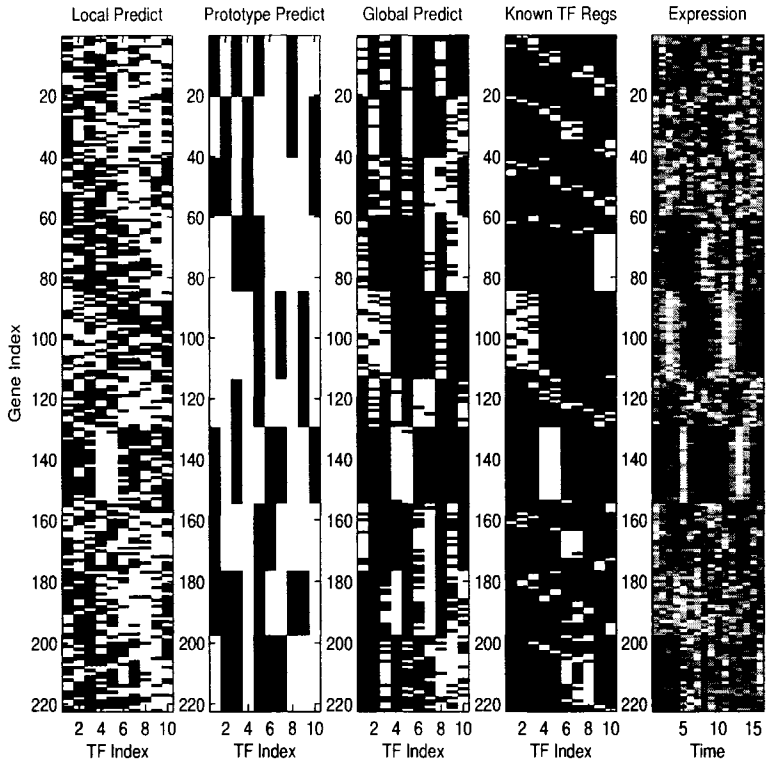
Both the proposed global regularization approach and the prototype based method require the genes to first be clustered based on their expression profiles. Although the number of clusters used has a minor effect on the performance of both algorithms, the sensitivity

to the cluster number was not significant provided that the cluster number is not extreme (neither extremely big nor extremely small). For my synthetic experiments, I simply choose to use 10 as the number of clusters.

Subfigure 5 in Figure 4.1 (rightmost subfigure) shows the expression profiles for the genes and TFs after their profiles have been clustered into 10 groups. I then learn the regulators for the genes in each group, using the globally regularized linear regression to encourage genes in the same group to share parents. I compared the results of the global approach to both the standard local approach of learning the parent regulators for each gene separately, and the prototype based approach of forcing all the genes in one group to have the exactly same set of parents. The comparison algorithms serve as controls at the two opposite extremes. I used the same L1 regularized method for parent selection in all of the algorithms. After obtaining the $\mathbf{w}$ parameters from each algorithm, all the parents indicated by $\mathbf{w} > 10^{-5}$ are determined as predicted regulators for the corresponding genes. For a fair comparison, the regularization parameters (α and λ) were chosen to yield the highest F-measure values in each case.

Subfigures 1–3 in Figure 4.1 show the regulator prediction results for the three algorithms respectively; comparing them with the true regulation information in subfigure 4. The x-axis for each subfigure indicates the candidate TFs from which a subset is selected as the set of regulators for each gene. The y-axis for each subfigure indexes the individual target genes. Each row plots the predicted regulators for each gene based on the corresponding $\mathbf{w}$ parameters for that gene. A white cell denotes a large weight ($w_{ij} > 10^{-5}$) connecting a TF j to a target gene i in the estimated linear model, indicating that j is inferred to regulate i, while a black cell denotes a small weight ($w_{ij} \leq 10^{-5}$), indicating that j is not inferred to regulate i.

The table in Figure 4.1 compares the performance of the three algorithms. The *precision* score measures true positive predictions (tp) divided by true positives plus false positive predictions (fp). That is, $precision = tp/(tp + fp)$. Similarly, *recall* score is measured in terms of the number of false negative predictions (fn), and is given by $recall = tp/(tp + fn)$. *F-measure* is a standard combination of both precision (p) and recall (r), given by $F\text{-}measure = 2\,p\,r/(p + r)$. The *accuracy* score measures the proportion of the correct predictions. That is, $accuracy = (tp + tn)/(tp + tn + fp + fn)$. Here the results show that the global regularization approach greatly outperforms both the local regularization and prototype based methods with respect to both accuracy and F-measure. The local prediction method is not able to effectively identify the true regulators due to the noise in



Performance comparison	Local regularization	Prototype method	Global regularization
accuracy (%)	57.6	47.2	73.0
precision (%)	21.4	18.1	29.9
recall (%)	71.5	74.6	63.8
F-measure	32.9	29.1	40.8

Figure 4.1: Results on synthetic data. Rows denote target genes in the synthetic experiment. Columns denote candidate regulators (transcription factors). Subfigure 1: local prediction output. Subfigure 2: prototype prediction output. Subfigure 3: global prediction output. Subfigure 4: ground truth regulatory relationships. Subfigure 5: expression level data used as input.

Table 4.1: Average comparison results for the synthetic experiments

Performance comparison	Local regularization	Prototype method	Global regularization
accuracy (%)	56.1 ± 0.4	47.3 ± 1.1	70.6 ± 0.4
precision (%)	19.7 ± 0.2	18.5 ± 0.4	26.2 ± 0.9
recall (%)	63.3 ± 1.2	74.3 ± 1.4	52.0 ± 2.6
F-measure	30.0 ± 0.3	29.6 ± 0.5	34.8 ± 1.4

the data and the limited number of time points. The prototype based method also has difficulty identifying correct regulatory relationships, and tends to choose too many parents for each gene. The reason for this is clear however. Since the prototype method is forced to choose a single set of regulators for controlling a large set of genes, it naturally chooses the union of the prospective regulators for each gene, leading to subsequently low precision and accuracy. Thus, the prototype approach depends heavily on having a more refined and accurate set of clusters from which it can make accurate regulatory inferences, but an accurate clustering is very hard to achieve in practice. Figure 4.1 shows, on the other hand, that the global regularization approach can effectively remove irrelevant candidate TFs by sharing co-regulation information within a group, while simultaneously reducing the number of spurious regulators being inferred by allowing individual differences between genes in a given cluster. The overall result is a more accurate (albeit far from perfect) recovery of the underlying regulatory structure.

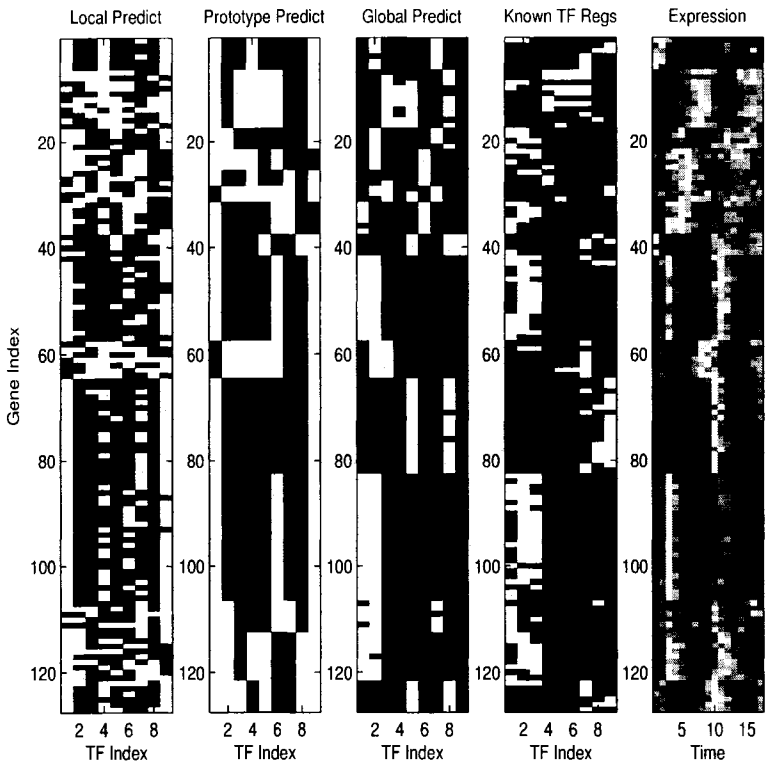
Note that the results in Figure 4.1 are obtained on one set of generated profiles using the parameters that yield the highest F-measure values. To see the significance of this comparison, I then conducted experiments in a more realistic, supervised manner. First I use one set of generated profiles to choose the parameters that optimize the F-measure values for each algorithm. Then I reran the experiments 10 times using different profiles generated from the synthetic setting, while keeping the chosen parameters fixed. The average comparison results and the standard derivations are shown in Table 4.1. These results suggest that the proposed global regularization approach is relatively more sensitive to the parameters than the other two approaches. Nevertheless, the proposed approach maintains a significantly better F-measure value.

The main question that remains is whether the higher quality inference on this synthetic model leads to improved results on real gene expression data, which I consider next.

4.4.2 Experiments on Real Data

Gene expression microarray data for the yeast cell cycle typically contains more than 6000 genes, while only a subset of these genes are cell cycle regulated. It is known there are 9 important transcription factors (TFs) that regulate the cell cycle process [84], namely: SWI4, SWI6, MPB1, FKH1, FKH2, NDD1, MCM1, ACE2 and SWI5. Since a lot of gene regulatory relationships have already been identified for yeast, this model is commonly used to evaluate learning approaches that attempt to infer gene regulatory networks from data. Here I use Cho et al. 's data [17], and focus on the task of identifying the subset of regulators from the 9 candidate TFs, for each yeast gene that is cell cycle regulated. To clearly evaluate the proposed approach, I chose a subset of 267 cell cycle regulated genes from the Cho et al. data [17], while I could obtain confirmed regulatory relationships from the previous literature [84, 50], or could obtain potential regulation relationships from existing binding data [84] for 127 genes among them. I rescaled the expression data to values between 0 and 1, and then clustered the genes into 15 clusters using K-means. (In the images shown in Figure 4.2, the genes are grouped vertically into the clusters. The number of clusters is chosen by using visual judgment to achieve a smooth clustering effect.) Finally, I tested the algorithms on each cluster. As in the synthetic experiments, after obtaining the $\mathbf{w}$ parameters from each algorithm, all the parents indicated by $\mathbf{w} > 10^{-5}$ are determined as predicted regulators for the corresponding genes. For a fair comparison, the regularization parameters (α and λ) were chosen to yield the highest F-measure values in each case.

Since the regulatory mechanisms are still not known for a portion of the 267 genes, I therefore can only evaluate the results over the 127 genes for which regulatory relationships are presumed known. Figure 4.2 shows the prediction results on 127 genes for all the three algorithms: locally regularized prediction, prototype based prediction, and globally regularized prediction. Here, again, a white cell denotes a large weight ($w_{ij} > 10^{-5}$) connecting a TF j to a target gene i in the estimated linear model, indicating that j is inferred to regulate i, and a black cell denotes a small weight ($w_{ij} \leq 10^{-5}$), indicating that j is not inferred to regulate i. Therefore the images compare the performance of the three methods on inferring regulators from among the 9 candidate TFs, and shows how they related to the known TF-based regulatory relationships. These results show that the globally regularized approach can improve the quality of both the standard locally regularized approach and the prototype based approach adapted from [94]. As in the synthetic case, the globally regularized approach has the ability to share regulatory information between genes within



Performance comparison	Local regularization	Prototype method	Global regularization
accuracy (%)	57.8	55.4	73.9
precision (%)	22.3	21.2	35.7
recall (%)	47.5	48.0	43.4
F-measure	30.4	29.4	39.2

Figure 4.2: Results on the subset of the real gene expression data from [17], restricted to genes where TF-based regulation information is known or can be inferred from other sources [84, 50]. Rows denote target genes. Columns denote candidate regulators (transcription factors). Subfigure 1: local prediction output. Subfigure 2: prototype prediction output. Subfigure 3: global prediction output. Subfigure 4: ground truth regulatory relationships. Subfigure 5: expression level data used as input.

a cluster, leading to better noise robustness than the local approach. Here too, the global technique also overcomes the problem of being overly dependent on clustering quality, like the prototype approach, by allowing regulation differences with a cluster. For example, in Figure 4.2, in the group of genes indexed between 42-58, one can see that a large set of the errors produced by the standard independent approach (subfigure 1) have been corrected by sharing parent information throughout the cluster (subfigure 3). The global regularizer correctly recognizes that this set of late-G1 genes is regulated by a subset of SWI4/SWI6 and MBP1/SWI6. Although some local errors remain in this region (and elsewhere), clearly the overall quality of the parent prediction has been improved substantially in the global method. For these genes, the prototype based method (subfigure 2) recognizes two additional parents, perhaps due to noise.

Overall, the prediction quality achieved by these methods on this data is still somewhat limited, but has improved remarkably over the past few years, and in some sense is remarkable given the noise exhibited in the expression profiles (subfigure 5).

4.5 Conclusion

In this chapter, I have proposed a new globally regularized risk minimization approach for learning regulatory networks from gene expression data, which extends the feature selection variable controlled structure learning idea presented in Chapter 3 to deal with continuous data directly. Exploiting the assumption that genes with similar expression patterns are likely to be co-regulated, the proposed approach first clusters the genes, and then learns the regulatory relationships by encouraging genes with similar expression patterns to share regulators. The experimental results on both synthetic data and real cell cycle data show that this new approach is more effective at identifying important (transcription factor based) regulatory mechanisms than the standard independent approach, and a prototype based approach.

Thus far, I have considered using only gene expression data in the learning process. Further prediction improvements are likely to come from incorporating further sources of biologically relevant data, such as binding information [84], or other forms of prior knowledge beyond the co-regulation assumption made here. These informations can be nicely incorporated into the global risk minimization approach by using the $\mathbf{u}$ parameter vector.

Chapter 5

Discriminative Model Selection

5.1 Introduction

While Bayesian networks have often been used for modeling a joint probability distribution over a set of variables, they have also recently been widely used to address discriminative classification tasks. This has motivated a growing body of work on learning effective Bayesian network classifiers from data [49].

Learning Bayesian network classifiers poses the challenging problem of discriminative structure learning, in addition to parameter estimation. This is not a trivial challenge. For example, one can typically improve classification performance on training data by increasing the complexity of the model, which, however, can lead to inferior generalization performance on unseen test data. Although one can still use the generative score-based or constraint-based structure learning methods (discussed in chapters 2 and 3) to identify structures for Bayesian network classifiers, this is not an optimal approach because these methods optimize a goal that is different from discriminative classification. To clarify this issue, consider the classification task over a set of variables $X_1, ..., X_n, Y$, where Y is the class variable. Here, the class label for an instance $\mathbf{x}$ is determined only by the conditional model $P(y|\mathbf{x})$, which is different from the joint distribution $P(\mathbf{x}, y)$ that previous generative Bayesian network learning methods aim to identify.

Realizing that classification poses a distinct problem for Bayesian network structure learning, many researchers have investigated this issue. This dates back (at least) to naive Bayes classifiers [29], and has continued with various approaches that include feature selection [59], and alternative structures [14, 33], where [33] has also explicitly stated the discrepancy between obtaining good predictive accuracy and a good generative MDL score. [54] compared several model selection criteria (unsupervised/supervised marginal likelihood, supervised prequential likelihood, cross validation) on a restricted subset of Bayesian

network structures. [40] presented an algorithm for discriminatively learning Bayesian networks that used the conditional likelihood of the class variable given the evidence variables as the model selection criterion.

In this chapter, I investigate score-based approaches for the problem of learning structures for Bayesian network classifiers. Instead of using the generative MDL/BIC and BDe scores to control the heuristic search process, I propose two novel discriminative model selection criteria, *Conditional Bayesian Information Criterion* (CBIC) and $Bias^2+Variance$ (BV), which aim to identify the model with the best generalization classification performance. This is the first main contribution of this chapter.

To evaluate the proposed discriminative criteria, I conduct a series of controlled experiments to compare them with the classical MDL/BIC and BDe criteria and the straightforward discriminative *Conditional Likelihood* (CL) criterion on the task of learning structures for Bayesian network classifiers. The experiments are designed to provide a comprehensive comparison between these model selection criteria across various situations: (1) using generative or discriminative methods for parameter estimation; (2) using the whole training set for both parameter estimation and criteria computation or dealing with these two tasks in a cross validation manner; (3) conducting experiments with a set of true data generation structures whose Markov Blanket complexities around the class variable have a range of values. Since most of the criteria considered are asymptotically correct, I only investigate the practically useful case of learning Bayesian network classifiers with small training sets. Since the goal is to better understand the quality of the criteria themselves, independent of the underlying search algorithm, I use the same heuristic search procedure for all of them. In particular, I follow a standard framework for evaluating model selection criteria [52, 92] that considers only a set of small models so that *each* model can be evaluated. This comprehensive empirical study constitutes the second main contribution of this chapter.

The rest of this chapter is structured as follows. Section 5.2 provides the framework for discriminative structure learning. Section 5.3 presents the proposed discriminative model selection criteria. The empirical study is reported in Section 5.4. This work was originally published in [41].

5.2 Bayesian Network Classifiers

Classification is one of the most important tasks addressed in machine learning. Solving a classification problem requires building a classification model over a set of variables

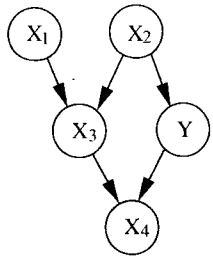


Figure 5.1: An example for Bayesian network classifier

$X_1, ..., X_n, Y$, where Y denotes the class variable, and then using the model to answer queries of the form what is value of Y given $\mathbf{X} = \mathbf{x}$? Because they encode uncertainty and provide a probabilistic foundation for modeling and inference, Bayesian networks are a natural model for answering such queries through sound probabilistic reasoning.

Recall that a Bayesian network encodes a joint distribution over a set of variables using an acyclic graph structure $\mathcal{G}$ that is associated with a set of parameters $\boldsymbol{\theta}$. When used as a classifier, the natural way to use a Bayesian network is to assign the most likely value of y given $\mathbf{x}$. Specifically, for a given structure $\mathcal{G}$ and parameters $\boldsymbol{\theta}$, the classification rule is given by

$$y^*_{(\mathcal{G},\boldsymbol{\theta})}(\mathbf{x}) = \operatorname*{argmax}_y P(y|\mathbf{x}, \mathcal{G}, \boldsymbol{\theta}) \tag{5.1}$$

To determine the conditional probabilities, and hence the classification, it turns out that not all the variables in the Bayesian network are relevant. In fact, only the variables that fall into the Markov Blanket of the class variable Y affect the inference (the concept of Markov Blanket is introduced in Definition 2.2). Essentially the Markov Blanket identifies the smallest set of variables that shield the target Y from the rest of the network. To clarify this, see Figure 5.1, which shows an example for Bayesian network classifier. In this example, the only variable that is not in the Markov Blanket of Y is X_1, and therefore this variable is irrelevant to the classification of Y.

A good Bayesian network classifier is one that produces accurate classifications. Typically, a Bayesian network classifier $(\mathcal{G}, \boldsymbol{\theta})$ is evaluated in terms of the misclassification error it obtains on unseen test data

$$err(\mathcal{G}, \boldsymbol{\theta}) = E_{P(\mathbf{x},y)}\left[1_{\left(y \neq y^*_{(\mathcal{G},\boldsymbol{\theta})}(\mathbf{x})\right)}\right] \tag{5.2}$$

Therefore the goal of learning is to obtain a Bayesian network that minimizes this score with respect to the true underlying distribution $P(\mathbf{x}, y)$. While one does not know this

distribution *a priori*, a set of test data $S = [\mathbf{x}^1 y^1; \ldots; \mathbf{x}^T y^T]$ can be drawn from $P(\mathbf{x}, y)$ to help evaluate Bayesian network models through the evaluation

$$err_S(\mathcal{G}, \theta) = \frac{1}{T} \sum_{i=1}^{T} 1_{\left(y^i \neq y^*_{\langle \mathcal{G}, \theta \rangle}(\mathbf{x}^i)\right)} \tag{5.3}$$

Note that although the goal of this chapter is to identify the optimal structure for a Bayesian network classifier given complete training data $D = [\mathbf{x}^1 y^1; \ldots; \mathbf{x}^N y^N]$, it first requires the parameters to be estimated, since the evaluation of each candidate structure is usually established on the parameters. Therefore, the parameter estimation strategy must also be considered. One strategy would be to use the standard generative maximum likelihood parameter estimation (ML). The other, however, would be to use discriminative parameter learning. In particular, one can use the maximum conditional likelihood parameter estimation (MCL) discussed in Section 2.2.1 of Chapter 2. In my implementation, I used the ELR algorithm proposed in [39] to perform discriminative parameter training. This ELR algorithm learns the parameters by maximizing the conditional likelihood of the training data using a simple gradient-ascent procedure.

5.3 Discriminative Model Selection Criteria

Given the difference between the goals of learning a generative Bayesian network model versus learning a discriminative Bayesian network classifier, I propose two new discriminative model selection criteria to guide the heuristic structure search—aiming to find the structure with the best generalization performance with respect to misclassification error.

The first criterion I propose is a discriminative variant of the standard BIC/MDL criterion. Note that the standard BIC score, see Equation (2.6), consists of two terms: a log likelihood term on the training data and a penalty term for model complexity. These two terms are used to balance the tradeoff between fit to the training data and model complexity. Note however, when using a Bayesian network for classification, one is concerned with the conditional likelihood of the target labels $\mathbf{y}$ given the $\mathbf{x}$ values, instead of the joint likelihood of the entire instances. Furthermore, many variables in the Bayesian network might not be relevant for classification if they do not belong to the Markov Blanket of Y, as discussed above. Therefore the standard model complexity term in the BIC score does not reflect the true complexity of the classifier. Taking these two observations into consideration, I propose a discriminative measure for scoring Bayesian network structures for classification, based on extending the standard BIC score. The new score I propose, *Conditional Bayesian*

Information Criterion (CBIC), is defined as follows

$$\text{CBIC}(\mathcal{G}, D) = \sum_{i=1}^{N} \log P(y^i|\mathbf{x}^i, \mathcal{G}, \boldsymbol{\theta}) - \frac{k_{MB}(\mathcal{G}, Y) \log N}{2} \tag{5.4}$$

where the first term is the conditional likelihood of labels on the training data; the second term measures the complexity of the conditional model where $k_{MB}(\mathcal{G}, Y)$ is defined as the number of free parameters in the substructure of $\mathcal{G}$ that falls into the Markov Blanket of the variable Y. By modifying BIC in this way to take into account only the relevant structural complexity of the classification model, the goal is to achieve a more accurate tradeoff between conditional data fitness and classifier complexity.

The second discriminative criterion I propose is motivated by a more general view of the classification task. Ripley [72] proves that the *expected mean-square-error* of a classifier corresponds to an additive combination of $Bias^2$ and *Variance*. Thus if one can measure both the bias and variance for all conditional distributions $P(y|\mathbf{x})$ encoded in a candidate Bayesian network $(\mathcal{G}, \boldsymbol{\theta})$, then the generalization classification performance of that candidate can be measured using a $Bias^2$+*Variance* (BV) criterion. Here, I define the BV criterion as follows

$$\text{BV}(\mathcal{G}, D) = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{P}(y^i|\mathbf{x}^i) - P(y^i|\mathbf{x}^i, \mathcal{G}, \boldsymbol{\theta}) \right)^2 + \widehat{Var}\left(P(y^i|\mathbf{x}^i, \mathcal{G}, \boldsymbol{\theta}) \right) \tag{5.5}$$

where, as before, $\boldsymbol{\theta}$ denotes the parameters trained on the training set D by either maximum likelihood estimation or maximum conditional likelihood estimation. The first term in the BV criterion is the square of bias for a conditional distribution $P(y^i|\mathbf{x}^i, \mathcal{G}, \boldsymbol{\theta})$ encoded in the candidate Bayesian network, where the mean for this conditional distribution is approximated using the empirical conditional distribution $\hat{P}(y^i|\mathbf{x}^i)$ on the training data. The empirical conditional distribution is defined as

$$\hat{P}(y|\mathbf{x}) = \begin{cases} \frac{\#_{\mathbf{x},y}}{\#_{\mathbf{x}}} & \text{if } \mathbf{x} = \mathbf{x}^i \text{ for some } i \in \{1 \ldots N\} \\ \text{undefined} & \text{otherwise} \end{cases} \tag{5.6}$$

where $\#_{\mathbf{x},y} = \sum_{i=1}^{N} 1_{(\mathbf{x}^i=\mathbf{x}, y^i=y)}$ and $\#_{\mathbf{x}} = \sum_{i=1}^{N} 1_{(\mathbf{x}^i=\mathbf{x})}$. The second term of the BV criterion measures the variance of the conditional distribution $P(y|\mathbf{x}, \mathcal{G}, \boldsymbol{\theta})$. Here, I adopt the variance estimate proposed in [93], which is derived from a Bayesian perspective

$$\widehat{Var}\left(P(y|\mathbf{x}, \mathcal{G}, \boldsymbol{\theta})\right) \tag{5.7}$$
$$= \sum_{j\mathbf{b}} \frac{1}{\#_{j\mathbf{b}} + \alpha_{j\mathbf{b}} + 1} \left[\begin{array}{l} \sum_a \frac{1}{\theta_{ja\mathbf{b}}} \left[P(a, \mathbf{b}, y|\mathbf{x}, \mathcal{G}, \boldsymbol{\theta}) - P(y|\mathbf{x}, \mathcal{G}, \boldsymbol{\theta}) P(a, \mathbf{b}|\mathbf{x}, \mathcal{G}, \boldsymbol{\theta})\right]^2 \\ - \left[P(\mathbf{b}, y|\mathbf{x}, \mathcal{G}, \boldsymbol{\theta}) - P(y|\mathbf{x}, \mathcal{G}, \boldsymbol{\theta}) P(\mathbf{b}|\mathbf{x}, \mathcal{G}, \boldsymbol{\theta})\right]^2 \end{array} \right]$$

where $\#_{j\mathbf{b}}$ and $\alpha_{j\mathbf{b}}$ denote the empirical and prior counts for $[\mathbf{x}_{\pi(j)} = \mathbf{b}]$ respectively, as introduced in Section 2.2.1 of Chapter 2.

Given these two discriminative structure scoring criteria, the main question is how they perform at learning Bayesian network classifiers. To answer this, I conducted a series of experiments comparing them to standard generative criteria and a straightforward discriminative *Conditional Likelihood* (CL) criterion

$$\mathrm{CL}(\mathcal{G}, D) = \sum_{i=1}^{N} \log P(y^i | \mathbf{x}^i, \mathcal{G}, \boldsymbol{\theta}) \tag{5.8}$$

5.4 Empirical Studies

This section reports on an empirical study that compares the two proposed discriminative model selection criteria to standard generative BIC/MDL and BDe criteria, and to a simple discriminative CL criterion, on the task of identifying the best Bayesian network model for classification. To attempt to provide a reasonably comprehensive study, I set up experiments across a range of contexts.

First, since the computation of model selection criteria requires parameter estimation, I conducted two types of comparisons, one using generative maximum likelihood parameter estimation (ML) and the other using discriminative maximum conditional likelihood estimation (MCL).

Second, there is an issue of how to use the training data for the combination of the parameter estimation and criterion computation. One approach is to use the entire training set for both tasks—I denote this approach as 1SS (1 single set). The other approach addresses these two tasks in a cross validation like manner, by splitting the training set into two subsets, and then using one subset for parameter estimation and the other subset for criterion computation. I denote this alternative approach as CV (cross validation). In my experiments, I used 5CV (5 fold cross validation) in particular. That is, the training set is first divided into 5 equal-sized subsets, then the following process is repeated for each of the 5 subsets: use the other 4 subsets for parameter estimation and then compute the criterion on the chosen subset. Finally, the candidate structure is scored with the average of the 5 scores computed.

Third, for each different combination of parameter estimation technique (ML vs MCL) and data using strategy (1SS vs 5CV), I study the performance of each criterion with respect to underlying true Bayesian network structures with various Markov Blanket complexities.

Procedure for generating sequence of structures:

Given a Bayesian network structure $\mathcal{G}^t$, used to generate training and test data, I generate a sequence of candidate structures with decreasing/increasing complexities as follows:

1. Starting from the original structure $\mathcal{G}^t$, sequentially delete one randomly selected edge from the Markov Blanket of the class variable, to generate a series of structures whose class variable has decreasing Markov Blanket size.
2. Starting from the original structure $\mathcal{G}^t$, sequentially add one random edge into the Markov Blanket of the class variable, while respecting to the acyclicity, to generate a series of structures whose class variable has increasing Markov Blanket size.

Figure 5.2: Candidate structure generation procedure

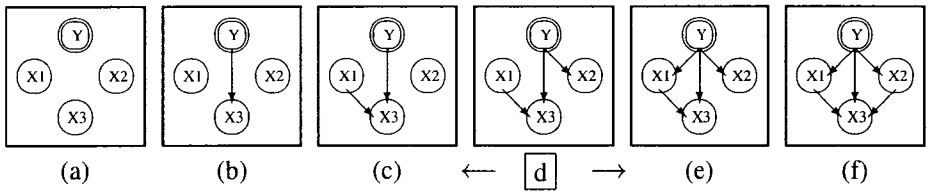


Figure 5.3: Sequence of structures; (d) is the original structure

Section 5.4.1 first describes the detailed experimental set up and evaluation method used. Then Section 5.4.2 presents the experimental comparison of the two proposed model selection criteria against the standard criteria in each particular context.

5.4.1 Experimental Setup

The goal of the experimental design is to attempt to investigate the performance of the various model selection criteria independent of the heuristic search procedure. Therefore, I used a set of candidate Bayesian network structures generated by perturbing a given target structure—the structure of the Bayesian network used to generate the training and test data—to approximate the structure search space. Details for the structure generation procedure are given in Figure 5.2. The procedure uses basic operators (adding/deleting an edge) to approximate the standard structure search process. Using this procedure, a sequence of candidate structures with a range of Markov Blanket complexities (with respect to the class variable) can be generated. Figure 5.3 shows an example of candidate structure sequence, where Y is the class variable, and structure in (d) is the starting point—the original data generation structure.

In particular, I run each experiment as follows. First, a Bayesian network structure $\mathcal{G}^t$ is constructed whose $k_{MB}(\mathcal{G}^t, Y)$ is within a specified range. Second, $\mathcal{G}^t$ is used to

generate the candidate structure sequence. Third, parameters θ^t are chosen randomly for the Bayesian network model $(\mathcal{G}^t, \theta^t)$, which is then used to generate the training set D and test set S. Finally, given a set of training data and an approximated structure space, each criterion c is used to select its best structure $\mathcal{G}^c$ from the candidate structure set.

One final issue to address is how to evaluate the performance of each model selection criterion. Here, I measure the performance of each criterion c by its *Relative Model Selection Error* (RMSE), which is defined as

$$\text{RMSE}(c) = \frac{err_S(\mathcal{G}^c, \theta^c)}{err_S(\mathcal{G}^*, \theta^*)} \tag{5.9}$$

where $err_S(\mathcal{G}^c, \theta^c)$ is the test misclassification error on S of the Bayesian network chosen by criterion c, while $(\mathcal{G}^*, \theta^*)$ is the Bayesian network chosen from the candidate structure sequence that has the smallest test error on S. Note that θ^c and θ^* are parameters trained on D for structure $\mathcal{G}^c$ and $\mathcal{G}^*$ respectively, according to the specific setting discussed at the beginning of the Section 5.4.

Since accurate structure selection is most challenging, and more relevant, when given limited training data, I focus on using small training sizes below.

5.4.2 Results

Now I present the results achieved under various experimental contexts. The results reported in this section are all obtained using training sets of size 50 and test sets of size 1000.

Using Maximum Likelihood Parameter Estimation

First, I consider the context of using generative maximum likelihood parameter estimation (ML) and 1SS manipulation of the training set—dealing with parameter estimation and criteria computation on the entire training set. To evaluate the different criteria with respect to Markov Blanket complexity, I randomly generated six groups of Bayesian network structures on 7 variables with varying Markov Blanket complexities with respect to the class variables. Each group includes 30 structures with Markov Blanket complexities on the same level. Each structure is used for data generation and structure search space construction as described in Section 5.4.1. Then each criterion c can be used to select the best structure from each structure sequence. The performance of each criterion is evaluated using the RMSE measure.

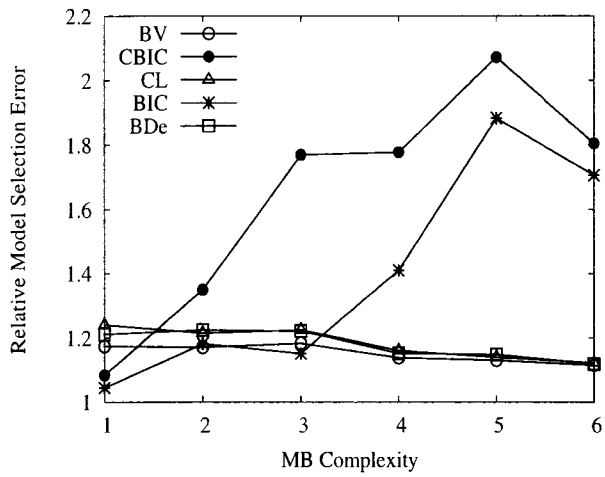


Figure 5.4: Comparison under context ML+1SS

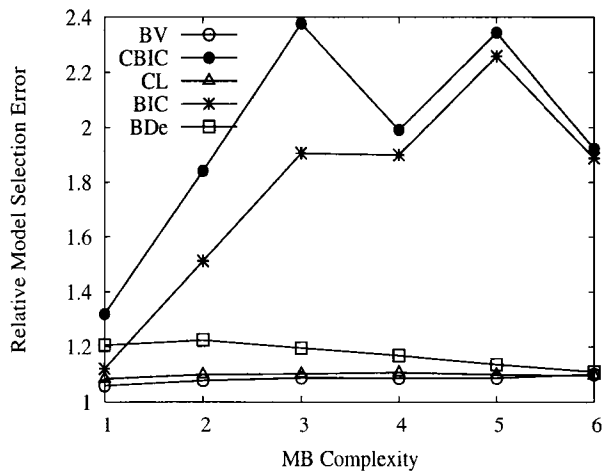


Figure 5.5: Comparison under context ML+5CV

Figure 5.4 shows the results for the ML+1SS set up. Here the y-axis indicates the six groups in the order of increasing Markov Blanket complexities regarding the data generation Bayesian networks. The figure shows that the proposed BV criterion performs the best in most cases, with the other discriminative criterion CL and the generative BDe criterion behaving similarly to BV. However, one can notice that CL and BDe demonstrate inferior performance when the Markov Blanket complexity is low, whereas their performance is similar to BV when the Markov Blanket complexity is high. This outcome reflects the fact that CL usually prefers complex structures, while BDe also tends to overfit for small training sets. The other two criteria, BIC and the proposed discriminative variant CBIC, however, both perform much worse across almost the entire range of Markov Blanket complexities, except the smallest complexity case. This suggests that BIC and CBIC overpunish complex models on small training sets, particularly CBIC which puts relatively less emphasis on the data likelihood term comparing to BIC.

Second, I repeated the previous experiments except that, instead of using 1SS, I used 5CV (5 fold cross validation) to manage the data for parameter estimation and criterion computation. Figure 5.5 shows that similar results are obtained to before, except now the BV and CL criteria overperform all the other criteria across the entire range of Markov Blanket complexities, while BV is slightly better than CL. This suggests the 5 fold cross validation technique effectively overcomes the overfitting problem of CL. BIC and CBIC still perform much worse. Note that the BDe criterion does not make sense in the 5CV setting, since it integrates parameter estimation and score computation together. Therefore I simply computed the BDe score on the training partition used for the criterion computation in this setting. This means that a smaller dataset is used for BDe computation than in the 1SS case, which explains why BDe has inferior performance to BV and CL in this case.

Using Maximum Conditional Likelihood Parameter Estimation

Since the ultimate goal in this chapter is learning a good Bayesian network classifier, it is natural to consider maximum conditional likelihood parameter estimation (MCL) instead of the generative parameter estimation technique used above. I repeated the previous two sets of experiments, using MCL instead of ML parameter estimation.

Figures 5.6 and 5.7 show the results for the two data manipulation strategies, 1SS and 5CV, respectively. Note that BDe criterion is not shown here since it does not have a conditional variant. Otherwise, the comparative results achieved here are quite similar to the ones reported above for ML parameter estimation. Once again, BIC and CBIC perform

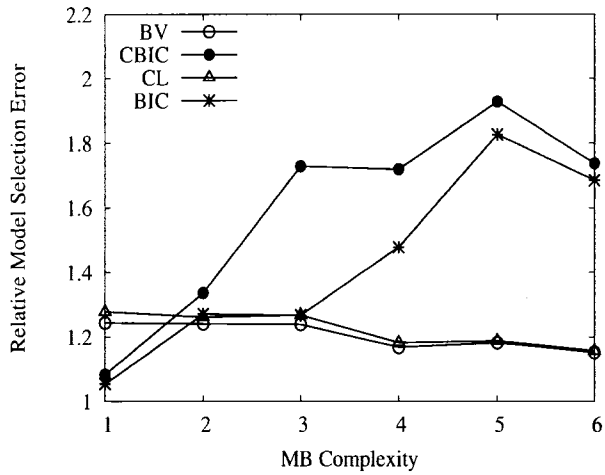


Figure 5.6: Comparison under context MCL+1SS

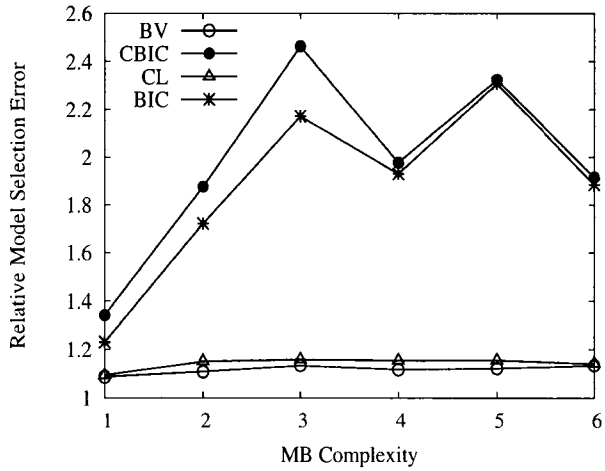


Figure 5.7: Comparison under context MCL+5CV

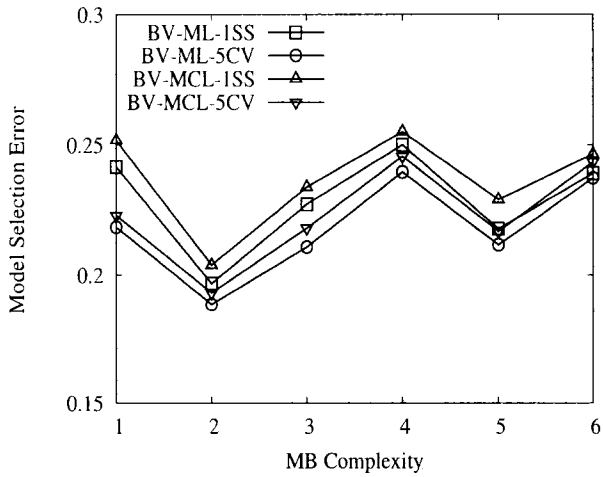


Figure 5.8: Comparison for BV's performance under the four contexts

much worse than BV and CL, except for the smallest complexity case in the 1SS context. In the 5CV context, however, both BV and CL criteria perform substantially better than BIC and CBIC across the entire range of Markov Blanket complexities. This is reasonable since MCL parameter estimation cannot overcome the over-penalization problem of BIC and CBIC. In both contexts, the proposed discriminative BV criterion performs best at all but the smallest complexity level.

Overall, these experimental results suggest that the discriminative BV criterion is a reasonable choice for learning the structure for Bayesian network classifiers using score-based structure search approach.

Comparing Parameter Estimation Regimes

The results presented so far provide the relative model selection error for each model selection criterion in fixed contexts. They suggest that the discriminative BV criterion is the best choice overall in each fixed context. These within-context comparisons, however, can not tell us under which specific context BV performs best.

I thus answer this question by comparing the BV criterion across the four contexts. I use the direct model selection error—test error $err_S(\mathcal{G}^{BV}, \boldsymbol{\theta}^{BV})$ as the performance measure where $(\mathcal{G}^{BV}, \boldsymbol{\theta}^{BV})$ denotes the model picked by criterion BV.

Figure 5.8 suggests that the performance of the BV criterion with respect to the Markov Blanket complexity is similar in all four contexts. Comparing the performance of BV with respect to each of the four contexts suggests an ordering ML+5CV $>$ MCL+5CV $>$ ML+1SS $>$ MCL+1SS, where "$>$" denotes "better than". Therefore, for BV, using

maximum conditional likelihood parameter estimation is not helpful, and in fact leads to worse performance than using maximum likelihood parameter estimation. However, 5CV is always helpful using both ML and MCL parameter estimation, relative to the 1SS data manipulation strategy. This reflects the ability of 5CV to help recognize the generalization performance of a candidate structure. Thus one can conclude that the BV criterion with ML parameter estimation and 5CV data manipulation is the most advantageous technique overall for score-based structure learning of Bayesian network classifiers.

5.5 Conclusion

In this chapter, I proposed two new discriminative model selection criteria, BV and CBIC, directed at the problem of learning structures for Bayesian network classifiers. CBIC is a discriminative variant of the standard BIC score, whereas BV is derived from a standard decomposition of the expected mean-square-error for classification. I conducted an empirical study that compared these two proposed criteria to the standard generative BIC and BDe criteria, and a simple discriminative CL criterion. This comparison was conducted in four contexts: ML+1SS, ML+5CV, MCL+1SS and MCL+5CV. The results suggest the BV criterion I proposed outperforms the other criteria across each context and most Markov Blanket complexities. A further study of BV across contexts reveals that ML+5CV is the best setting for BV. By contrast, the proposed CBIC suffers from the problem of over-penalization and demonstrate poor performance. I feel this investigation can provide a useful reference in the study of discriminative Bayesian network structure learning.

Chapter 6

Maximum Margin Bayesian Networks

6.1 Introduction

When training probability models for classification tasks, it is often recommended that the model parameters should be optimized under a discriminative training criterion such as conditional likelihood [33, 55, 56]. In this chapter, I investigate the problem of discriminative parameter learning for Bayesian network classifiers assuming given fixed structures. But instead of conditional likelihood, here I consider applying the maximum margin methodology to Bayesian networks to formulate a novel maximum margin parameter estimation approach, which can also be extended into the case of multiple class variables.

Maximum margin training is one of the most popular discriminative learning strategies available. Recently, it has been observed that Markov networks (undirected graphical models) can be efficiently trained to maximize the margin, even simultaneously, over a set of class variables [1, 88, 89, 91]. These training algorithms have adopted the Euclidean normalization constraint of support vector machines (SVMs), which can be accommodated in their frameworks because they rely on an undirected graphical model representation that allows a single arbitrary normalization. However, unlike Markov network models, Bayesian networks require additional normalization constraints to be satisfied; namely that the local clique potentials represent conditional probability distributions. These constraints are very different from the standard Euclidean normalization constraints of SVMs. Developing of maximum margin methodology for Bayesian networks under these local normalization constraints is much harder than for Markov networks. Nevertheless, Bayesian networks do not preclude the possibility of learning large margin classifiers. My goal in this chapter is to exploit the benefits of large margin training, while still being able to represent the learned

classifier as a Bayesian network. For the purpose of deriving the maximum margin training technique for Bayesian networks, I adopt the exponential Bayesian network representation introduced in (2.9), by substituting θ with ω using the logarithmic transformation (2.10).

There are several motivations for attempting to maintain a Bayesian network representation, even when performing large margin training. First, the classification model being learned could be a fragment of a much larger probabilistic causal model. In this case, maintaining a Bayesian network representation could allow one to integrate the learned model with a pre-existing background model without additional effort. Second, the normalization constraints asserted by a directed graphical structure capture nonparametric *causal* knowledge about the domain. Therefore, respecting these constraints allows one to exploit the advantages of Bayesian networks for capturing intuitive causal structure. Note that removing the normalization constraints would turn the Bayesian network into a Markov network, and this would unavoidably remove the causal knowledge that was originally encoded by the local normalization constraints.

To understand both the prospects and limitations of learning maximum margin Bayesian network classifiers, I proceed as follows. First, I investigate the notion of *classification margin* for Bayesian network classifiers in Section 6.2.1, and relate this to the common conditional likelihood criterion of graphical models. I then present a convex relaxation in Section 6.2.2 that can be used to derive an effective training algorithm in Section 6.3. Section 6.4 shows this algorithm solves a range of problems exactly and otherwise provides an effective heuristic for finding approximate solutions. In Section 6.5, I then present experimental results which show that the causal information in Bayesian networks can help achieve effective generalization performance when the directed graphical structure captures relevant causal knowledge. Finally, I extend the approach to deal with multiple class variables in Section 6.6 and present further experimental results in Section 6.7. This work was originally published in [45].

6.2 Maximum Margin Bayesian Networks

The goal of this section is to derive a convex maximum margin formulation for discriminative Bayesian network parameter training. Towards this goal, I first formulate a maximum margin training criterion within the Bayesian network framework, and then obtain a convex optimization formulation from it by relaxing the nonlinear equality constraints.

6.2.1 Maximum Margin Training Criteria

I initially assume there is a single class variable Y taking on values $y \in \{1..,V\}$. (I will extend this to multiple class variables in Section 6.6 below.) As introduced in Section 5.2, one usually makes predictions through conditional probabilities by $y^* = \text{argmax}_y P(y|\mathbf{x})$, while the conditional probabilities of y depend only on variables that are within the Markov Blanket of variable Y in the Bayesian network. Thus I will restrict my attention to the classification relevant subset of variables henceforth.

As discussed previously in this thesis, there are two parameter estimation methods that are often used to train Bayesian network parameters: a generative maximum likelihood (ML) parameter estimation and a discriminative maximum conditional likelihood (MCL) parameter estimation. Here, instead I investigate an alternative objective criterion based on the large margin criteria of SVMs. In particular, I adopt the multiclass margin definition of [22]. In the current context, given training data $D = [\mathbf{x}^1 y^1; \ldots; \mathbf{x}^N y^N]$, this objective can be cast maximizing the minimum conditional likelihood ratio (MCLR)

$$\text{MCLR}(\boldsymbol{\theta}) = \min_{i=1}^{N} \min_{y \neq y^i} \frac{P(y^i|\mathbf{x}^i, \boldsymbol{\theta})}{P(y|\mathbf{x}^i, \boldsymbol{\theta})}$$

which can be turned into a margin form by taking the logarithms of both sides

$$\log \text{MCLR}(\boldsymbol{\theta}) = \min_{i=1}^{N} \min_{y \neq y^i} \log P(\mathbf{x}^i, y^i|\boldsymbol{\theta}) - \log P(\mathbf{x}^i, y|\boldsymbol{\theta}) \tag{6.1}$$

Thus my goal is to find a set of parameters $\boldsymbol{\theta}$ that *maximizes* the minimum margin between the target class label against the best alternative under the probability model. (I introduce slack variables to obtain a soft margin version of the criterion later.)

To see the connection to SVMs more clearly, note that one can substitute the exponential form of $P(\mathbf{x}, y|\boldsymbol{\omega})$ given in (2.9) into the MCLR objective, to obtain

$$\begin{aligned} \log \text{MCLR}(\boldsymbol{\omega}) &= \min_{i=1}^{N} \min_{y \neq y^i} \left[\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, y)\right]^\top \boldsymbol{\omega} \\ &= \min_{i=1}^{N} \min_{y \neq y^i} \boldsymbol{\Delta}(i, y)\, \boldsymbol{\omega} \end{aligned} \tag{6.2}$$

where $\boldsymbol{\Delta}(i, y) = [\boldsymbol{\phi}(\mathbf{x}^i, y^i) - \boldsymbol{\phi}(\mathbf{x}^i, y)]^\top$, and $\boldsymbol{\phi}(\mathbf{x}, y)$ is defined in Section 2.3. Here the row vector $\boldsymbol{\Delta}(i, y)$ plays the role of the feature vector for training example i and class label y. Therefore I can write the entire set of feature vectors as a matrix Δ of size $(NV) \times |\boldsymbol{\phi}|$, where $|\boldsymbol{\phi}|$ denotes the number of features.

Thus, starting with the training objective (6.1), through a change of parameters, I am led to a training problem that can be cast as a conventional maximum margin problem

$$\max_{\boldsymbol{\omega}, \gamma} \gamma \qquad \text{subject to} \quad \Delta\boldsymbol{\omega} \geq \gamma\boldsymbol{\delta}, \quad \|\boldsymbol{\omega}\| \leq 1 \tag{6.3}$$

where $\delta_{(i,y)} = 1_{(y \neq y^i)}$, so δ is a vector of length NV. Note here I have added the normalization constraint $\|\boldsymbol{\omega}\| \leq 1$. Obviously some form of normalization is necessary to avoid making $\Delta\boldsymbol{\omega}$ large in a trivial manner just by making $\boldsymbol{\omega}$ large. Euclidean normalization happens to yield a weight vector that maximizes the Euclidean margin [80]. The resulting constrained optimization problem corresponds to the standard version of multiclass SVMs proposed in [22] (ignoring slacks), here expressed over features determined by the Bayesian network.

In fact, this connection between probabilistic and large margin classifiers is one of the main observations of [1, 88], which then proceed to use standard SVM training criteria over these features. Note however that the solution weight vector for (6.3) cannot be substituted into the Bayesian network representation, because it will not satisfy the proper normalization constraints (2.12); namely that $\sum_a e^{\omega_{ja\mathbf{b}}} = 1$ for all $j, \mathbf{b}$. The previous techniques of [1, 88] were able to proceed by using an *undirected* graphical model which could accommodate unnormalized weights in the potential function. However, for Bayesian networks this is not sufficient, and it is usually hard to represent the same classifier in the original Bayesian network structure.

The alternative approach I consider, therefore, is to maximize the same objective, but subject to constraints that preserve the representability as a Bayesian network

$$\max_{\boldsymbol{\omega},\gamma} \gamma \quad \text{subject to } \Delta\boldsymbol{\omega} \geq \gamma\boldsymbol{\delta}, \quad \sum_a e^{\omega_{ja\mathbf{b}}} = 1 \ \forall j, \mathbf{b} \tag{6.4}$$

Unfortunately, these natural constraints on $\boldsymbol{\omega}$ are nonlinear and this yields a difficult optimization problem. Attempts to reformulate the problem according to standard transformations also fail. For example, although using $\boldsymbol{\theta}$ parameters instead of $\boldsymbol{\omega}$ can change the local normalization constraints into linear constraints, it creates difficulty in the margin constraints $\Delta \ln(\boldsymbol{\theta}) \geq \gamma\boldsymbol{\delta}$. The standard trick of removing the normalization constraints via the alternative transformation (2.13), which uses the $\mathbf{w}$ parameters, also does not work in this case, since it creates terms of the form $\sum_{ja\mathbf{b}} \Delta_{(j,a,\mathbf{b})}(i,y) \left[w_{ja\mathbf{b}} - \log \sum_{a'} e^{w_{ja'\mathbf{b}}}\right]$, which cause the optimization to be neither convex nor concave in $\mathbf{w}$. Thus, if one hopes to solve the maximum margin Bayesian network training problem exactly, a more subtle approach is required.

6.2.2 Convex Relaxation

Although solving for the maximum margin Bayesian network parameters appears to be a hard problem, it is still possible to derive a practical training algorithm that solves the

problem for a range of graph topologies, and otherwise provides a useful foundation for approaches that seek local maxima. The main idea is to try to exploit convexity in the problem as much as possible, and identify situations where the solutions to a convex subproblem can be maintained.

First note that the objective and the margin constraints in (6.4) are linear in ω. Unfortunately, the normalization constraints are nonlinear equalities on ω, which eliminates the convexity of the problem. However, my basic observation is that the problem can be made convex simply by relaxing these equality constraints to inequality constraints, thus yielding a simple relaxation

$$\max_{\boldsymbol{\omega},\gamma} \gamma \quad \text{subject to } \Delta\boldsymbol{\omega} \geq \gamma\boldsymbol{\delta}, \quad \sum_{a} e^{\omega_{ja\mathbf{b}}} \leq 1 \ \ \forall j, \mathbf{b} \tag{6.5}$$

The solution to this problem will of course be subnormalized. The key fact about the relaxed problem (6.5), however, is that it is convex in ω and this will permit effective algorithmic approaches [8]. Note that the inequality form of the norm constraints in (6.3) and (6.5) is not vacuous: In either case, reducing the magnitude of the weights only has the effect of reducing the inner products in the margin constraints ($\boldsymbol{\Delta\omega}$), which can only yield a smaller margin γ. The maximization objective naturally forces the weight magnitudes overall to become as large as possible, subject to the normalization constraints.

It is interesting to compare the two convex optimization problems (6.3) and (6.5), which correspond to maximum margin Markov networks and Bayesian networks respectively. These problems have identical objectives and margin constraints on ω, but differ only in the normalization constraints—one global constraint for Markov networks versus multiple local constraints for Bayesian networks. The solutions to the two problems will obviously be different. Intuitively, the Bayesian network constraints might regularize the weights more comprehensively in the sense that each local CPT is constrained to have identical maximum influence, whereas a Markov network could concentrate its weight in a single local function.

Soft Margin Formulation

Before tackling real problems I need to introduce slack variables, since it is obviously not practical to use a hard margin formulation on real data. To this end, I consider the standard soft margin formulation of SVMs

$$\min_{\boldsymbol{\omega},\boldsymbol{\xi}} \frac{1}{2}\|\boldsymbol{\omega}\|^2 + C\boldsymbol{\xi}^\top \mathbf{1} \quad \text{subject to } \Delta\boldsymbol{\omega} \geq \boldsymbol{\delta} - S\boldsymbol{\xi} \tag{6.6}$$

where $\boldsymbol{\xi}$ are the slack variables; $\mathbf{1}$ denotes the vector of all 1 entries; S is an $(NV) \times N$ sparse matrix with nonzero entries $S((i,y),i) = 1$ (which enforces the constraint that $\boldsymbol{\Delta}(i,y)\boldsymbol{\omega} \geq \delta_{(i,y)} - \xi_i$ for all i, y); and C is a parameter that controls the slack effect [22]. Note that $\boldsymbol{\xi} \geq 0$ is already implied in (6.6), because $\boldsymbol{\Delta}(i,y^i) = \mathbf{0}$ and $\delta(i,y^i) = 0$ for all i. For my purpose, I need to state this objective explicitly in terms of the margin γ. It can be shown that (6.6) is equivalent to the following [58]

$$\min_{\boldsymbol{\omega},\gamma,\boldsymbol{\xi}} \frac{1}{2\gamma^2} + C\boldsymbol{\xi}^\top \mathbf{1} \qquad \text{subject to } \Delta\boldsymbol{\omega} \geq \gamma(\boldsymbol{\delta} - S\boldsymbol{\xi}), \quad \gamma \geq 0, \ \|\boldsymbol{\omega}\| \leq 1 \tag{6.7}$$

Thus, by replacing the Euclidean normalization constraint with the Bayesian network sub-normalization constraints, I obtain

$$\min_{\boldsymbol{\omega},\gamma,\boldsymbol{\xi}} \frac{1}{2\gamma^2} + C\boldsymbol{\xi}^\top \mathbf{1} \qquad \text{subject to } \Delta\boldsymbol{\omega} \geq \gamma(\boldsymbol{\delta} - S\boldsymbol{\xi}), \quad \gamma \geq 0, \ \textstyle\sum_a e^{\omega_{ja\mathbf{b}}} \leq 1 \ \ \forall j, \mathbf{b} \tag{6.8}$$

The two problems, (6.7) and (6.8), specify the soft margin formulations of maximum margin Markov networks and Bayesian networks respectively. Unfortunately, neither formulation is convex because the quadratic term $\gamma(\boldsymbol{\delta} - S\boldsymbol{\xi})$ is non-convex in the optimization variables γ and $\boldsymbol{\xi}$ [8]. For Markov networks one can simply convert (6.7) back to (6.6) and thus convexify the problem. It is of course no surprise that optimization problems can be converted between convex and non-convex formulations without affecting the optimal solution. For Bayesian networks I instead solve the following problem with alternative slack variables $\boldsymbol{\epsilon}$ and controlling parameter B

$$\min_{\boldsymbol{\omega},\gamma,\boldsymbol{\epsilon}} \frac{1}{2\gamma^2} + B\boldsymbol{\epsilon}^\top \mathbf{1} \qquad \text{subject to } \Delta\boldsymbol{\omega} \geq \gamma\boldsymbol{\delta} - S\boldsymbol{\epsilon}, \quad \gamma \geq 0, \ \sum_a e^{\omega_{ja\mathbf{b}}} \leq 1 \ \ \forall j, \mathbf{b} \tag{6.9}$$

This new formulation (6.9) is convex and equivalent to (6.8), and thus yields a convex version of the soft margin training problem for Bayesian networks.

Proposition 6.1 *Assuming $\gamma \geq 0$, $(\boldsymbol{\omega}, \gamma, \boldsymbol{\xi})$ is an optimal point for C in (6.8) if and only if $(\boldsymbol{\omega}, \gamma, \boldsymbol{\epsilon})$ is an optimal point for B in (6.9) with $\boldsymbol{\epsilon} = \gamma\boldsymbol{\xi}$ and $B = C/\gamma$.*

So if one chooses an optimal regularization parameter B for (6.9), then the optimal solution $(\boldsymbol{\omega}, \gamma)$ will be preserved, while the slacks $\boldsymbol{\xi}$ can be recovered by $\boldsymbol{\xi} = \boldsymbol{\epsilon}/\gamma$.

I now proceed to develop algorithmic approaches for solving the convex training problem (6.9), with the goal of ultimately comparing maximum margin Markov networks trained under (6.6) versus Bayesian networks trained under (6.9).

6.3 Training Algorithm

To solve (6.9) first consider the Lagrangian

$$\begin{aligned} & L(\boldsymbol{\omega}, \gamma, \boldsymbol{\epsilon}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \nu) \\ &= 1/(2\gamma^2) + B\boldsymbol{\epsilon}^\top \mathbf{1} + \boldsymbol{\eta}^\top(\gamma\boldsymbol{\delta} - S\boldsymbol{\epsilon} - \Delta\boldsymbol{\omega}) \\ &\quad + \sum_{j,\mathbf{b}} \lambda_{j\mathbf{b}} \left(\sum_a e^{\omega_{ja\mathbf{b}}} - 1 \right) - \nu\gamma \end{aligned} \tag{6.10}$$

The saddle point condition gives us an equivalent problem to (6.9)

$$\min_{\boldsymbol{\omega},\gamma,\boldsymbol{\epsilon}} \max_{\boldsymbol{\eta},\boldsymbol{\lambda},\nu} L(\boldsymbol{\omega}, \gamma, \boldsymbol{\epsilon}, \boldsymbol{\eta}, \boldsymbol{\lambda}, \nu) \quad \text{subject to } \boldsymbol{\eta}, \boldsymbol{\lambda}, \nu \geq 0 \tag{6.11}$$

Unfortunately, this Lagrangian is not nearly as convenient as the one for the SVM formulation (6.6), and a closed form solution for the dual is not readily obtainable in this case. For example, one cannot easily eliminate the primal variables from this problem: taking the partial derivative with respect to $\omega_{ja\mathbf{b}}$ yields

$$\frac{\partial L}{\partial \omega_{ja\mathbf{b}}} = \lambda_{j\mathbf{b}}\, e^{\omega_{ja\mathbf{b}}} - \boldsymbol{\eta}^\top \boldsymbol{\Delta}_{ja\mathbf{b}} \tag{6.12}$$

where $\boldsymbol{\Delta}_{ja\mathbf{b}}$ denotes the $ja\mathbf{b}$ column of Δ. The difficulty with (6.12) is that one cannot set this derivative to zero because $\boldsymbol{\eta}^\top \boldsymbol{\Delta}_{ja\mathbf{b}}$ can be negative ($\boldsymbol{\Delta}$ has negative entries). Nevertheless, the problem remains convex.

Rather than use a Lagrangian approach to solve this problem, I instead consider a standard barrier approach. In fact, barrier methods are among the most effective techniques for solving convex constrained optimization problems [8, 95]. In this approach one simply replaces the constraints with log barrier functions

$$\begin{aligned} \min_{\boldsymbol{\omega},\gamma,\boldsymbol{\xi}} \; & \frac{1}{2\gamma^2} + B\boldsymbol{\epsilon}^\top \mathbf{1} \\ & - \mu \sum_{(i,y)} \log\left(\boldsymbol{\Delta}(i,y)\boldsymbol{\omega} - \gamma\delta_{(i,y)} + \epsilon_i\right) \\ & - \mu \sum_{j,\mathbf{b}} \log\left(1 - \sum_a e^{\omega_{ja\mathbf{b}}}\right) \\ & - \mu \log(\gamma) \end{aligned} \tag{6.13}$$

In general, it can be shown that for convex inequality constraints, the resulting unconstrained objective (6.13) is also convex, while the solution to (6.13) converges to (6.9) as $\mu \to 0$ [8]. In the standard path following technique, an outer loop is used to solve a sequence of unconstrained optimization problems for a sequence of decreasing μ values,

where the optimal solution $(\omega, \gamma, \epsilon)$ obtained from one loop iteration is used as the starting point of the next iteration. The parameter μ is initially set to a reasonable value to ensure numerical stability, and then successively reduced to sharpen the barriers until a small value of μ is reached. For each fixed μ, the inner optimization problem is usually solved using a second order method to ensure fast convergence. Here, for the inner optimization loop, I implemented a Newton descent method based on computing the gradient and Hessian of (6.13) with respect to $(\omega, \gamma, \epsilon)$. I found that 7 outer iterations with $\mu^{(1)} = 1$, $\mu^{(k+1)} = \mu^{(k)}/10$, and fewer than 20 inner Newton iterations were required to obtain accurate solutions. In principle, the runtime of a barrier iteration method is not dramatically slower than solving a quadratic program [8]. However, my Matlab implementation is currently an order of magnitude slower than the quadratic program solver (CPLEX) I used for maximum margin Markov networks. The largest runtimes for my barrier training in the experiments below are a few minutes, versus a few seconds for CPLEX.

6.4 Bayesian Networks with Exact Solutions

Before presenting experiments, I first consider when the solutions to the relaxed problem (6.9) correspond to the solutions to the exact problem; i.e., satisfying local normalizations: $\sum_a e^{\omega_{jab}} = 1$, for all $j, \mathbf{b}$. The main concern is that the solutions obtained to (6.9) may not be representable in a Bayesian network because the parameters ω are subnormalized, not normalized. This leaves us with the question of determining when these subnormalized solutions can be equivalently converted into properly normalized Bayesian networks.

It turns out that a range of network topologies admit a simple procedure for renormalizing the local parameters so that they become proper CPTs, without affecting the conditional probability of y given $\mathbf{x}$. In fact, this observation has been previously made by [98, 99]. I present a simpler view here. In fact, it is easy to characterize a sufficient condition for an unnormalized Bayesian network classifier to be renormalized to preserve $P(y|\mathbf{x})$.

Proposition 6.2 *An unnormalized directed graphical model, defined on the Markov Blanket of the class variable Y, can be renormalized to preserve the decision function $P(y|\mathbf{x})$ if for each child variable of Y, its parents are fully connected.*

The intuition behind this result is fairly straightforward. Let local function $f(x_j, \mathbf{x}_{\pi(j)})$ denote the unnormalized parameter for configuration with child value x_j and parent values $\mathbf{x}_{\pi(j)}$. This local function can always be normalized by dividing a factor $\rho(\mathbf{x}_{\pi(j)}) = \sum_x f(x, \mathbf{x}_{\pi(j)})$ for each $\mathbf{x}_{\pi(j)}$. The renormalization does not affect the classification as

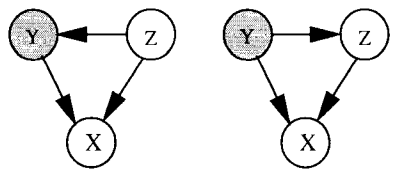


Figure 6.1: Bayesian networks that satisfy the renormalization condition

long as there is another local function $f(x_k, \mathbf{x}_{\pi(k)})$ where $\mathbf{x}_{\pi(j)} \subseteq \{x_k, \mathbf{x}_{\pi(k)}\}$ that can be multiplied by the same factor. Since the functions and variables follow an acyclic ordering in a Bayesian network, child variables can be sequentially renormalized bottom up without affecting previous normalizations. Finally, the renormalization procedure reaches the local function for Y variable which can be directly normalized without finding another local function to counterbalance the normalization factor. To clarify this, assume the adjusted local function for Y is $f'(y, \mathbf{x}_{\pi(Y)})$ after renormalizations of the functions associated with the children of Y. Then the prediction probability is written as

$$P(y|\mathbf{x}) = \frac{f'(y, \mathbf{x}_{\pi(Y)}) \prod_{j \in C(Y)} P(x_j|\mathbf{x}_{\pi(j)}, y)}{\sum_{y'} f'(y', \mathbf{x}_{\pi(Y)}) \prod_{j \in C(Y)} P(x_j|\mathbf{x}_{\pi(j)}, y')} \tag{6.14}$$

where $C(Y)$ represents the indices of the child variables of Y. Then renormalizing function $f'(y, \mathbf{x}_{\pi(Y)})$ to $P(y|\mathbf{x}_{\pi(Y)})$ only causes the same factor $\rho(\mathbf{x}_{\pi(Y)}) = \sum_y f'(y, \mathbf{x}_{\pi(j)})$ to be divided from both the numerator and the denominator in (6.14), and therefore does not affect the prediction function. Figure 6.1 shows two simple Bayesian network examples that satisfy the renormalization condition.

The renormalization strategy only fails if, at any stage, the parent variable set $\mathbf{x}_{\pi(j)}$ is not contained in another single local function, but is instead split between separate local functions, as in Figure 6.2. In this case, there would be no way to coordinate the compensation for $\rho(\mathbf{x}_{\pi(j)})$ (without adding a new local function over $\mathbf{x}_{\pi(j)}$). Thus, in the end, one is left with an intuitive sufficient condition for renormalizing a Bayesian network: any graph can be normalized without affecting $P(y|\mathbf{x})$ if the child variables of Y can be eliminated without adding any new edges. In these cases, one can recover a normalized model while maintaining the optimality of the solution to (6.9).

Note that the renormalization procedure can be applied to any set of parameters defining the decision rule $P(y|\mathbf{x})$ in a network structure satisfying Proposition 6.2, even if the parameters were produced by a Markov network training procedure. However, this does not imply that the resulting model $P(y|\mathbf{x})$ from Markov network training is optimal under the Bayesian network criterion (6.9).

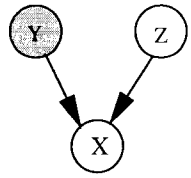


Figure 6.2: A Bayesian network that does not satisfy the renormalization condition

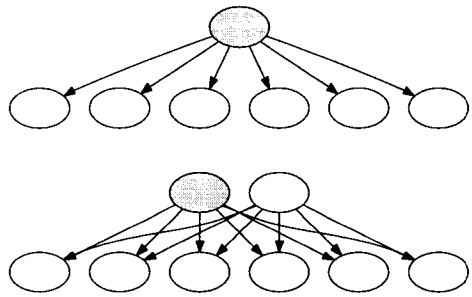


Figure 6.3: Two Bayesian networks where the class variable Y is shaded in each of them

6.5 Experimental Results

To evaluate the utility of learning maximum margin Bayesian networks, I conducted some simple experiments on both real and synthetic data sets. In the synthetic experiments, I first constructed the Bayesian network structure and parameters, and then used it to generate training and test data. The goal of the synthetic experiments is to run a controlled comparison of maximum margin Bayesian networks versus Markov networks in causal domains where one can obtain the correct Bayesian network structures. I experimented with several network topologies and parameterizations, and compared maximum margin Bayesian networks (MMBN) trained according to (6.9) against maximum margin Markov networks (MMMN) trained according to (6.6), and also against maximum conditional likelihood (MCL) trained with a gradient descent method. The results reported here are average misclassification errors on test data over 100 repetitions. For each method, on each model, the regularization parameters, B and C respectively, were first optimized on one pair of training and test sample sets and then fixed for the duration of the experiment.

The synthetic experiments were conducted on the networks shown in Figure 6.3. Here I fixed a network structure and then defined the generative model by selecting parameters from a skewed distribution. I used a parameter β to control the skewness of the conditional distributions of each child, where a value of $\beta = 1$ makes each child a deterministic function of the parents, and $\beta = 0.5$ gives each child a uniform distribution, rendering them effec-

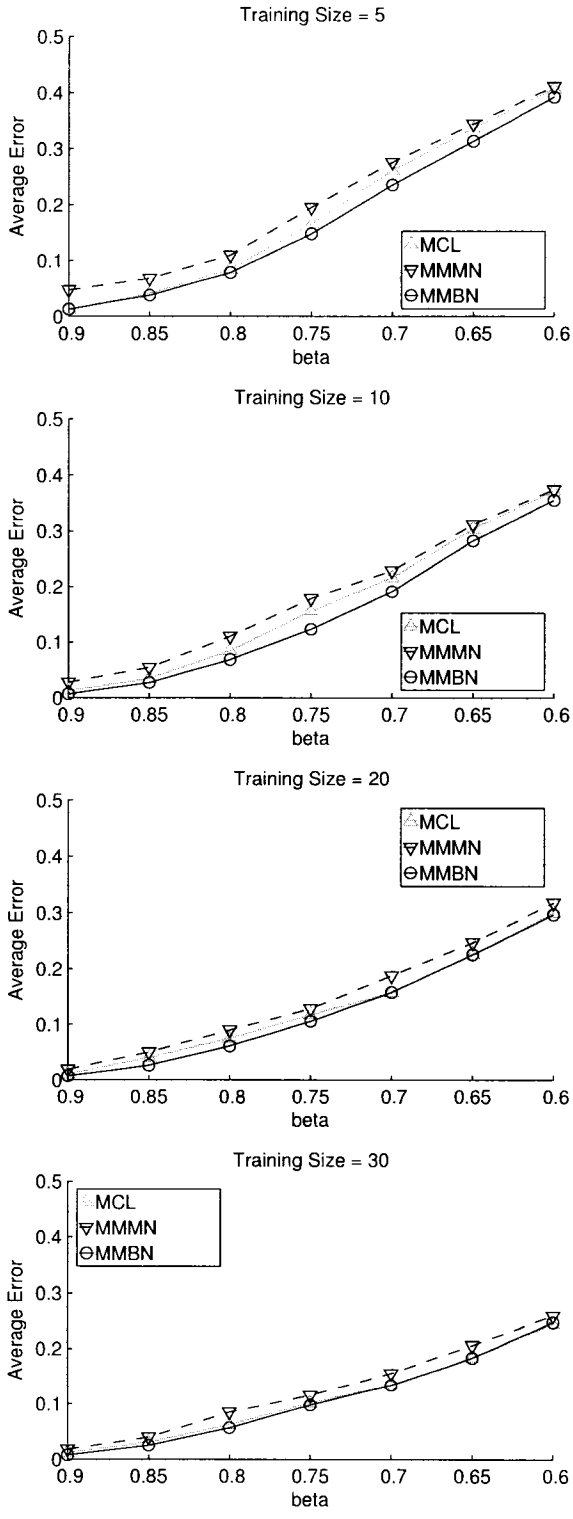


Figure 6.4: Average error results for Figure 6.3 (top)

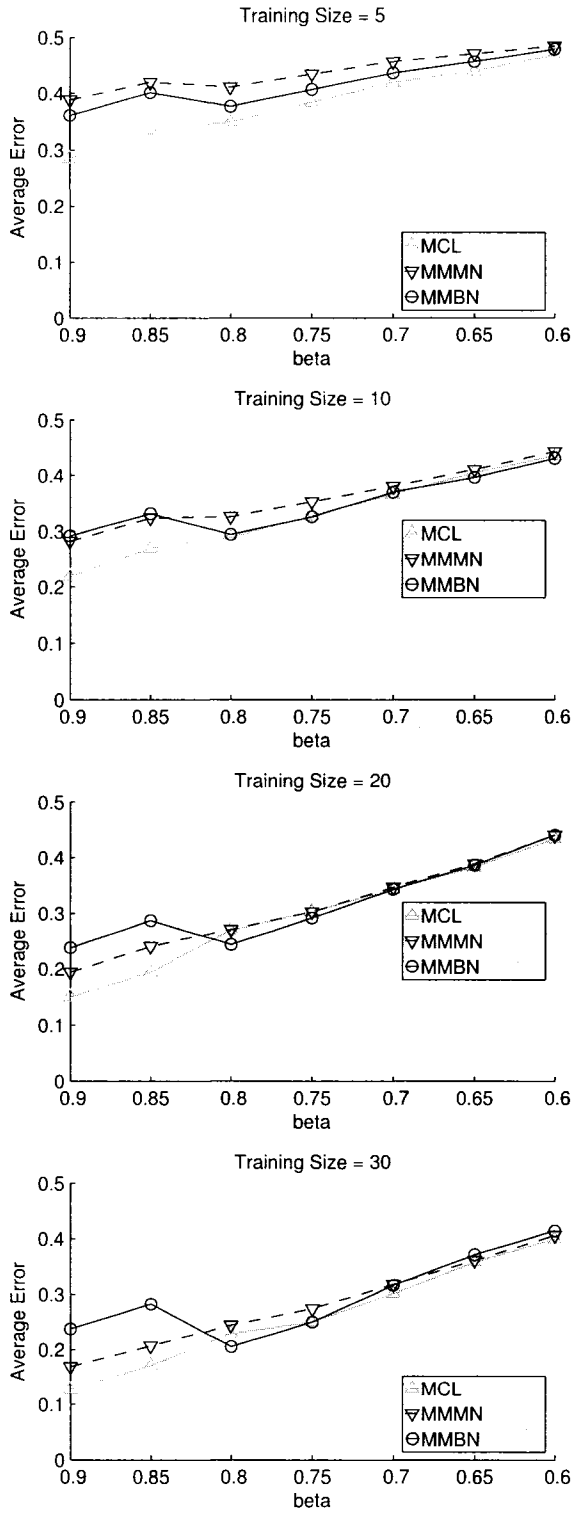


Figure 6.5: Average error results for Figure 6.3 (bottom)

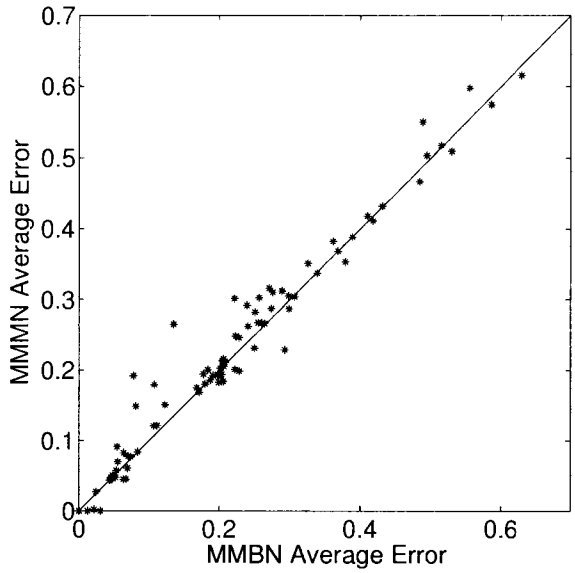


Figure 6.6: Average error comparison between MMBN and MMMN on UCI data sets

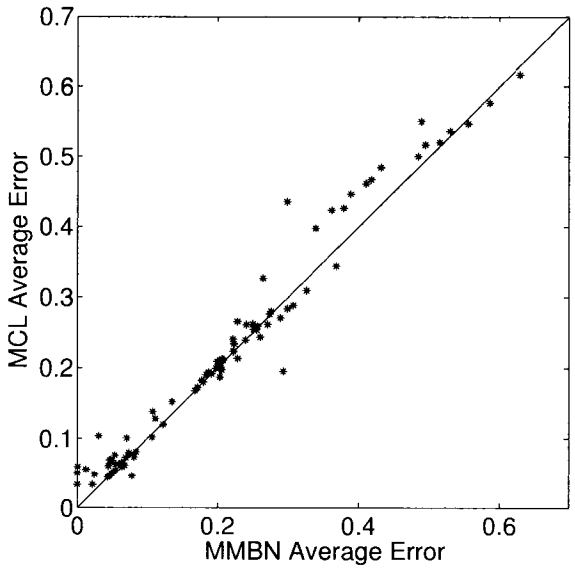


Figure 6.7: Average error comparison between MMBN and MCL on UCI data sets

tively independent of their parents. Figure 6.4 shows the comparison of the three strategies, MMBN, MMMN, and MCL for the first network topology shown in Figure 6.3. This network topology satisfies the condition of Proposition 6.2, and therefore MMBN computes a globally optimal solution in this case. Here one can see that for a range of generative models defined by β and several training sample sizes, MMBN demonstrates a systematic advantage over both MMMN and MCL, although MCL is clearly stronger than MMMN in this case. Figure 6.5 shows the same comparison using the second network topology from Figure 6.3. This network no longer satisfies the condition of Proposition 6.2, and therefore the training algorithm is no longer guaranteed to produce an optimal normalized solution (only an optimal subnormalized solution). Nevertheless, one can see that MMBN holds a slight advantage over MMMN in this case, while MCL is slightly better here. MMBN appears to have an advantage in cases where it is exact (for structures satisfying Proposition 6.2), but the advantage is diminished a lot in the subnormalized case.

For a real world comparison, I also experimented with real data from the UCI repository. Specifically, I used 17 data sets: Australian*, Breast, Chess, Cleve, Diabetes*, Flare*, Glass, Glass2*, Heart, Hepatitis*, Iris*, Lymphography, Mofn*, Pima*, Vehicle*, Vote, and Waveform. For each data set, I formulated a Bayesian network topology that was intended to capture the causal structure of the domain, but in this case there was no guarantee that the presumed structure was correct. The network structures I used were automatically generated using the "PowerConstructor" technique discussed in [14]. These networks are much larger and cannot be easily visualized here. Nevertheless, in 9 of the 17 cases the network topologies satisfied the condition of Proposition 6.2 (marked * above). For each data set I considered 5 different training sample sizes, N =10, 20, 30, 40, 50. For each N, I run experiments on the random training/test splits. The results are averages over 5 repetitions with disjoint training sets when it was possible. Interestingly, Figure 6.6 shows that MMBN obtains an overall advantage over MMMN. Moreover, MMBN also shows a slight overall advantage over MCL on these data sets; see Figure 6.7.

6.6 Multivariate Extension

In this section, I consider extending the proposed maximum margin Bayesian networks to deal with structured classification problems with multiple connected class variable, following the key extension of [1, 19, 88]. In this setting, one observes training data $D = [\mathbf{x}^1\mathbf{y}^1, \ldots, \mathbf{x}^N\mathbf{y}^N]$ as before, however, now the targets $\mathbf{y}^i$ are *vectors* of values for cor-

related class variables. The first main issue is to adapt the training criterion (6.9) to this multivariate prediction case. Following [88], I scale the margin between a target class vector $\mathbf{y}^i$ and an alternative vector $\mathbf{y}$ proportional to the number of misclassifications, which is to set

$$\delta_{(i,\mathbf{y})} = \sum_k 1_{(y^i_k \neq y_k)} \tag{6.15}$$

This immediately yields multivariate versions of the training problems (6.6) and (6.9).

The primary difficulty in dealing with the multivariate form of these problems is coping with the exponential number of constraints in $\boldsymbol{\Delta}\boldsymbol{\omega} \geq \gamma\boldsymbol{\delta} - S\boldsymbol{\epsilon}$. That is, one now has to assert $\boldsymbol{\Delta}(i,\mathbf{y})\boldsymbol{\omega} \geq \gamma\boldsymbol{\delta}_{(i,\mathbf{y})} - \epsilon_i$ for all training examples i, over all possible label vectors $\mathbf{y}$. Such a constraint set is too large to handle explicitly, and an approach must be developed for handling them implicitly.

One of the key results in [88] is showing that, for maximum margin Markov networks (6.6), the constrained optimization problem can be factored and re-expressed in terms of "marginal" Lagrange multipliers $\alpha_{(i,\mathbf{y}_{ja\mathbf{b}})} = \sum_{\mathbf{y}\setminus\mathbf{y}_{ja\mathbf{b}}} \mu_{(i,\mathbf{y})}$, where $\mathbf{y}_{ja\mathbf{b}}$ denotes the subconfiguration of $\mathbf{y}$ that matches the local function j on pattern $a\mathbf{b}$. This allows a compact reformulation of an equivalent convex problem that can be solved efficiently as a compact quadratic program [88]. Unfortunately, this approach does not work readily in my current case because the Lagrangian (6.10) does not permit a simple closed form expression of the dual. Thus I have to follow a log-barrier approach to solve the problem (6.13). However, a direct factorization approach is not readily available for reducing the exponential sum in $\sum_{(i,\mathbf{y})} \log\left(\boldsymbol{\Delta}(i,\mathbf{y})\boldsymbol{\omega} - \gamma\delta_{(i,\mathbf{y})} + \epsilon_i\right)$. Nevertheless the constraint generation strategy of [1] can be usefully applied in this case.

To solve (6.9) in the multivariate case I implemented a cutting plane method, where initially only a small subset of constraints in $\boldsymbol{\Delta}\boldsymbol{\omega} \geq \gamma\boldsymbol{\delta} - S\boldsymbol{\epsilon}$ were considered. Given a current set of constraints, a solution $(\boldsymbol{\omega}, \gamma, \boldsymbol{\epsilon})$ was computed using the barrier method outlined above. Then for each training example $(\mathbf{x}^i, \mathbf{y}^i)$ one new labeling $\mathbf{y}$ was generated to maximize the degree of constraint violation

$$\begin{aligned} & \operatorname*{argmax}_{\mathbf{y}} \ \gamma\delta_{(i,\mathbf{y})} - \epsilon_i - \boldsymbol{\Delta}(i,\mathbf{y})\boldsymbol{\omega} \\ = \ & \operatorname*{argmax}_{\mathbf{y}} \ \exp\left(\gamma\delta_{(i,\mathbf{y})} + \boldsymbol{\phi}(\mathbf{x}^i,\mathbf{y})^\top\boldsymbol{\omega}\right) \end{aligned} \tag{6.16}$$

This is in fact an inference problem that can be solved by conventional methods. For example, if $\mathbf{Y}$ forms a Markov chain, then the most violated constraint can be generated by a Viterbi algorithm run on the probability model defined by (6.16).

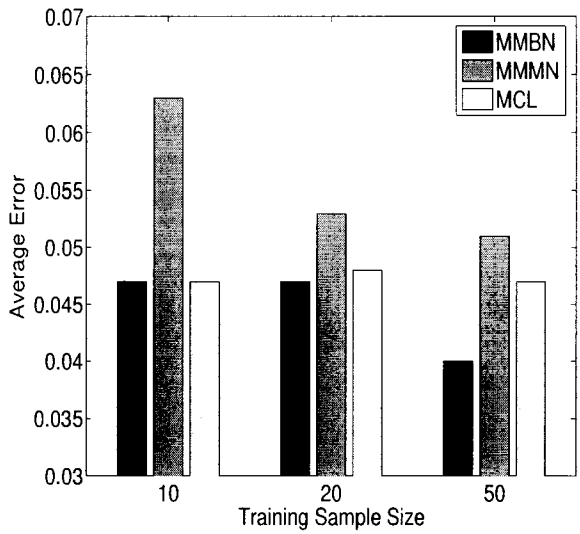


Figure 6.8: Average error results for MMBN and MMMN on synthetic networks with multiple class variables

Once the new constraints have been generated, they are added to the problem and the solution $(\omega, \gamma, \epsilon)$ is re-computed using the barrier method. In my experiments I found this constraint generation scheme was quite effective, requiring at most 10 to 50 generation iterations before solving the problem.

6.7 Multivariate Experimental Results

I implemented this approach and tested it on both synthetic and real data using HMM models for classification, where the class variables $\mathbf{Y}$ play the role of the hidden state sequence, and the input variables $\mathbf{X}$ play the role of the observations. Generally I considered models of the form depicted in Figure 6.9, where each Y variable has multiple (disjoint) X-children. In my synthetic experiment, I sampled $(\mathbf{x}, \mathbf{y})$ from a sequence of length 5 (5 Y variables with 4 X-children each, for a total of 20 X variables). I then used a generative model based on the same skewed parameters used in the synthetic single class variable experiments above; here with $\beta = 0.85$. The results reported here were averages on 20 repetitions and for 3 different training sizes (10, 20 and 50) respectively. Figure 6.8 shows that MMBN again outperforms MMMN and MCL in controlled experiments where the correct Bayesian network structure is known.

I also conducted an experiment on a protein secondary structure database [23]. Here the goal is to predict the sequence of secondary structure labels given an observed amino acid sequence. Figure 6.9 shows the prediction model I used. Basically, the secondary structure

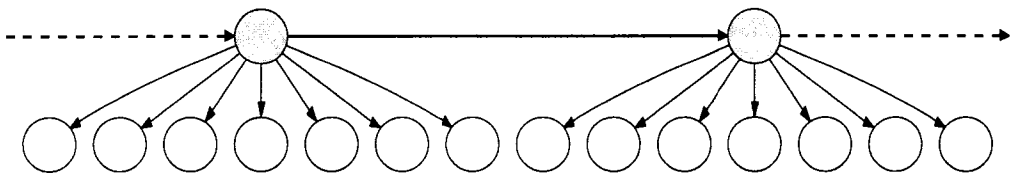


Figure 6.9: Structure of the protein secondary structure prediction model

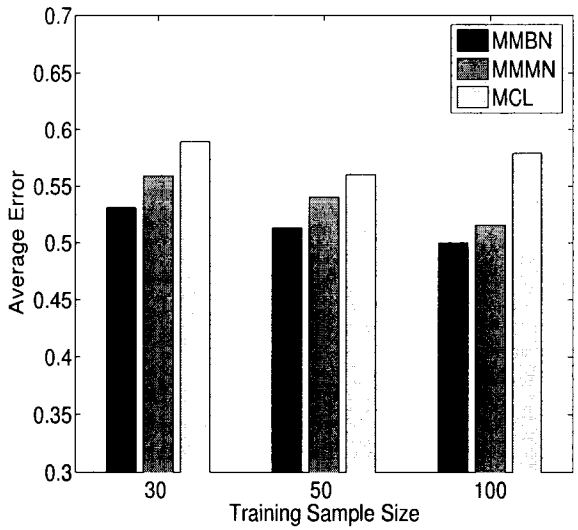


Figure 6.10: Average error results for MMBN and MMMN on protein secondary structure prediction

tag $\mathbf{y}_k$ for a location k in the amino acid sequence is predicted based on a sliding window of 7 amino acid observations, as well as the neighboring secondary structure tags. I trained on a subset of the data and tested on 1000 sampled subsequences disjoint from the training data. The experiment was repeated 20 times to obtain the average for each different training size of 30, 50 and 100. Figure 6.10 shows that MMBN performs better than both MMMN and MCL on this data set.

6.8 Conclusion

In this chapter, I have investigated the interesting issue for learning Bayesian network classifiers: whether a Bayesian network representation can be combined with discriminative training based on the maximum margin criterion of SVMs. I have found that training Bayesian networks under the maximum margin criterion is a nontrivial computational problem—harder than the standard quadratic program of SVM training. Nevertheless, I still developed a reasonable training algorithm that optimizes the margin exactly in special

cases, and provides a reasonable heuristic in general cases. I also extended the proposed maximum margin Bayesian network method to address the structured classification problems with multiple class variables. My experimental results for both single class variable and multivariate classifications show that there might be an advantage to respecting the causal model constraints embodied by a Bayesian network, if indeed these constraints were present during the data generation. In this sense, maximum margin Bayesian networks offer a new way to add prior knowledge to SVMs.

Chapter 7

Parameter Estimation with Hidden Variables

7.1 Introduction

The previous chapters of this thesis investigated Bayesian network learning assuming that the given training data was complete. In this chapter, I consider the problem of learning Bayesian network parameters in the presence of hidden variables. Hidden variables are variables whose values are not observed in the training data and yet remain part of the joint probability model we are estimating. The existence of hidden variables usually occurs for two practical reasons. One is when some variables encode derived or predicted concepts that are not readily observed in real world data, yet nevertheless remain important. For example, a disease classification variable is often important to infer in a medical Bayesian network model, but is not directly observable. The second reason is when auxiliary variables are deliberately introduced into the model to simplify the explanation of the observed data. For example, hidden variables can be introduced to reduce the complexity of the structure and therefore simplify the learning process for a given set of training data [32].

Learning Bayesian networks in the presence of hidden variables has been widely studied [5, 30, 32, 35, 76]. So far, the most common approach considered in this scenario is to adopt *expectation maximization* (EM) algorithm. Few algorithms are better known in machine learning and statistics than EM. One reason is that EM solves a common problem–learning from incomplete data–that occurs in almost every area of applied statistics. Equally well known to the algorithm itself, however, is the fact that EM suffers from shortcomings. Here it is important to distinguish between the EM algorithm (essentially a coordinate descent procedure [68]) and the objective it optimizes. Only one problem is due to the algorithm itself: since it is a simple coordinate descent, EM suffers from slow (linear) con-

vergence and therefore can require a large number of iterations to reach a solution. Standard optimization algorithms such as quasi-Newton methods can, in principle, require exponentially fewer iterations to achieve the same accuracy (once close enough to a well behaved solution) [8, 70]. Nevertheless, EM converges quickly in many circumstances [77, 87]. The main problems attributed to EM are not problems with the algorithm itself, but instead are properties of the objective it optimizes. In particular, the standard objective and its variants tackled by EM are not convex in any standard probability model. Non-convexity immediately creates the risk of local minima, which unfortunately is not just a theoretical concern: EM often does not produce very good results in practice, and can sometimes fail to improve significantly upon initial parameter settings [65]. For example, the field of unsupervised grammar induction [53] has been unsuccessful in its attempts to use EM for decades and is still unable to infer useful syntactic models of natural language from raw unlabeled text.

In this chapter, I present a novel convex approach to EM training that addresses the problem of learning Bayesian network parameters with hidden variables. This approach is based on formulating a convex relaxation of a particular variant of the EM algorithm—Viterbi EM. As in previous chapters, the goal of this convexification is to remove local minima from the training objective in an attempt to overcome one of the main shortcomings of EM algorithms. However, achieving an effective convex relaxation is nontrivial due to some technical barriers in this case. After introducing the standard variants of EM training in Section 7.2, I first show in Section 7.3 that *any* convex relaxation of the standard EM must produce trivial results if it maintains any dependence on the values of hidden variables. Although this result suggests that any convex relaxation of EM cannot succeed, I subsequently show in Section 7.4.1 that the problem can be overcome by deriving an equivalent parameter estimation formulation in terms of the equivalence relations over the values of the hidden variables, rather than the missing values themselves. Based on this new formulation, I then formulate a convex Viterbi EM formulation for Bayesian network parameter estimation in Section 7.4.2. The main technical contribution of this chapter is a reformulation of standard estimation principles for exponential conditional models in terms of equivalence relations on variable values, rather than the variable values themselves. Although this chapter only focuses on the hidden variable case, the technique I developed remains extendable to the general missing value case. This work has been accepted for publication [43].

7.2 EM Variants

Before proceeding, it is important to first clarify the precise EM variant that I address in this chapter. In fact, there are many EM variants that optimize different criteria. Let $\mathbf{z} = (\mathbf{x}, \mathbf{y})$ denote a complete observation, where $\mathbf{x}$ refers to the observed part of the data and $\mathbf{y}$ refers to the unobserved part; and let $\mathbf{w}$ refer to the parameters of the underlying probability model, $P(\mathbf{x}, \mathbf{y}|\mathbf{w})$. *Joint* and *conditional* EM algorithms are naive "self-supervised" training procedures that alternate between optimizing the values of the missing variables and optimizing the parameters of the model

$$\text{joint EM:} \qquad \begin{aligned} \mathbf{y}^{(k+1)} &= \arg\max_{\mathbf{y}} P(\mathbf{x}, \mathbf{y}|\mathbf{w}^{(k)}) \\ \mathbf{w}^{(k+1)} &= \arg\max_{\mathbf{w}} P(\mathbf{x}, \mathbf{y}^{(k+1)}|\mathbf{w}) \end{aligned} \tag{7.1}$$

$$\text{conditional EM:} \qquad \begin{aligned} \mathbf{y}^{(k+1)} &= \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}, \mathbf{w}^{(k)}) \\ \mathbf{w}^{(k+1)} &= \arg\max_{\mathbf{w}} P(\mathbf{y}^{(k+1)}|\mathbf{x}, \mathbf{w}) \end{aligned} \tag{7.2}$$

The joint version of EM (7.1) is also often referred to as *Viterbi* EM, and is sometimes proposed as a computationally convenient approximation to the standard EM algorithm (2.2) [79]. Both joint EM and conditional EM are clearly coordinate descent procedures that make monotonic progress in their objectives, $P(\mathbf{x}, \mathbf{y}|\mathbf{w})$ and $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$. Moreover, the criteria being optimized are in fact well motivated objectives for unsupervised training. The primary problem with these iterations is not that they optimize approximate or unjustified criteria, but rather that they rapidly get stuck in poor local maxima due to the extreme updates made on $\mathbf{y}$.

By far, the more common form of EM, contributing the very name expectation–maximization, is given by the familiar update (2.2). Although it is not immediately obvious what the standard EM iteration (2.2) optimizes, it has long been known that it monotonically improves the *marginal* likelihood $P(\mathbf{x}|\mathbf{w})$ [26]. Therefore, I refer to the standard EM algorithm as *marginal* EM. [68] later showed that the E-step could be generalized to

$$\max_{\mathbf{q_y}} \sum_{\mathbf{y}} \mathbf{q_y} \log \left(P(\mathbf{x}, \mathbf{y}|\mathbf{w}^{(k)}) / \mathbf{q_y} \right)$$

Due to the softer $\mathbf{q_y}$ update, the standard EM update does not converge as rapidly to a local maximum as the joint and conditional variants; however, as a result, it tends to find better local maxima. Marginal EM has subsequently become the dominant form of EM

algorithm in the literature. Nevertheless, none of the training criteria are jointly convex in the optimization variables, thus these iterations are only guaranteed to find local maxima.

Independent of the updates, the three training criteria are not equivalent nor equally well motivated. In fact, for most applications one is more interested in acquiring an accurate conditional $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$, rather than optimizing the marginal $P(\mathbf{x}|\mathbf{w})$. Of the three training criteria therefore (joint, conditional and marginal), marginal likelihood appears to be the least relevant to learning predictive models. In this chapter, I will focus on maximizing *joint* likelihood, since it incorporates aspects of both marginal and conditional training. I will also primarily consider the hidden variable case and assume a fixed set of random variables $Y_1, ..., Y_\ell$ is always unobserved, and a fixed set of variables $X_{\ell+1}, ..., X_n$ is always observed.

7.3 A Cautionary Result for Convexifying EM

As stated, I will focus on convexifying the *joint EM* objective. Assume we are given training data $[\mathbf{z}^1; ...; \mathbf{z}^N]$, where $\mathbf{z}^i = (\mathbf{x}^i, \mathbf{y}^i)$ such that $\mathbf{x}^i$ denotes the observed components and and $\mathbf{y}^i$ denotes the unobserved components. The goal in this chapter will be to develop a jointly convex relaxation to the minimization problem

$$\min_{\mathbf{y}} \min_{\mathbf{w}} -\sum_i \log P(\mathbf{x}^i, \mathbf{y}^i|\mathbf{w}) \tag{7.3}$$

where $\mathbf{y}$ takes only discrete assignments.

Thus one obvious issue one must face is to relax the discrete constraints on the assignment $\mathbf{y}$. However, the challenge is deeper than this. The difficulty dwells in the complete symmetry property that holds between configurations of values for the hidden variables. In particular, for any optimal solution $(\mathbf{y}, \mathbf{w})$ there must be at least another, equivalent solution $(\mathbf{y}', \mathbf{w}')$, corresponding to a permutation of the hidden variable values, that achieves the same objective value. For example, suppose there is only one hidden variable with domain $\{1, -1\}$. Then one can assign $\mathbf{y}'$ to be a permutation of $\mathbf{y}$ by replacing all the 1s in $\mathbf{y}$ with -1 and all the -1s with 1, and correspondingly assign $\mathbf{w}'$ to be a rearrangement of the weights in $\mathbf{w}$, without affecting the optimality of the solution. Unfortunately, this form of solution symmetry has devastating consequences for any convex relaxation. Assume one attempts to use any jointly convex relaxation $f(\mathbf{q}_\mathbf{y}, \mathbf{w})$ of the standard joint likelihood objective (7.3), where the the missing variable assignment $\mathbf{y}$ has been relaxed into a continuous probabilistic assignment $\mathbf{q}_\mathbf{y}$ (like standard EM).

Lemma 7.1 *If f is strictly convex and invariant to permutations of unobserved variable values, then the global minimum of f, $(\mathbf{q}_\mathbf{y}^*, \mathbf{w}^*)$, must assign equal probabilities to the different* $\mathbf{y}$ *configurations; that is, $\mathbf{q}_\mathbf{y}^*$ must be uniform.*

Proof: Assume $(\mathbf{q_y}, \mathbf{w})$ is a global minimum of f but $\mathbf{q_y}$ is not uniform. Then there must be some permutation of the hidden values, Π, such that the alternative $(\mathbf{q}'_\mathbf{y}, \mathbf{w}') = (\Pi(\mathbf{q_y}), \Pi(\mathbf{w}))$ satisfies $\mathbf{q}'_\mathbf{y} \neq \mathbf{q_y}$. By the permutation invariance of f, this implies $f(\mathbf{q_y}, \mathbf{w}) = f(\mathbf{q}'_\mathbf{y}, \mathbf{w}')$. Then by the strict convexity of f, one has $f\left(\alpha(\mathbf{q_y}, \mathbf{w}) + (1-\alpha)(\mathbf{q}'_\mathbf{y}, \mathbf{w}')\right) < \alpha f(\mathbf{q_y}, \mathbf{w}) + (1-\alpha) f(\mathbf{q}'_\mathbf{y}, \mathbf{w}') = f(\mathbf{q_y}, \mathbf{w})$, for $0 < \alpha < 1$, contradicting the global optimality of $f(\mathbf{q_y}, \mathbf{w})$. ∎

Therefore, any convex relaxation of (7.3) that uses a distribution $\mathbf{q_y}$ over hidden values and does not make arbitrary distinctions can never do anything but produce a uniform distribution over the hidden variable values. This trivialization is perhaps the main reason why standard EM objectives have not been previously convexified. (Note that standard coordinate descent algorithms simply break the symmetry arbitrarily and descend into some local solution.) This negative result seems to imply that *no* useful convex relaxation of EM is possible in the hidden variable case. However, my key observation is that a convex relaxation expressed in terms of an equivalence relation over the hidden values can avoid this symmetry breaking problem. In particular, equivalence relations exactly collapse the unresolvable symmetries in this context, while still representing useful structure over the hidden assignments. Representations based on equivalence relations are a useful tool for unsupervised learning that has largely been overlooked in the current literatures. My goal in this chapter, therefore, is to reformulate the joint EM training objective to use only equivalence relations on hidden variable values, and subsequently develop a relaxed convex formulation.

7.4 Convex EM

In this section, I will derive a convex relaxation of the joint EM to address the parameter estimation issue for Bayesian networks with hidden variables. In this context, I assume a Bayesian network defines the joint probability distribution over a set of random variables $\mathbf{Z} = (\mathbf{X}, \mathbf{Y})$, where $\mathbf{X}$ denotes the observed variables and $\mathbf{Y}$ denotes the hidden variables. Given a set of training data with only $\mathbf{X}$ observed $[\mathbf{x}^1; ...; \mathbf{x}^N]$, my goal is to train the Bayesian network parameters to maximize the joint data likelihood (7.3).

The derivation in this chapter is based on the exponential representation (2.14) introduced in Chapter 2. In particular, using this representation, the joint likelihood objective

$\sum_i \log P(\mathbf{x}^i, \mathbf{y}^i)$ can be written as

$$\sum_i \log P(\mathbf{z}^i|\mathbf{w}) = \sum_j \sum_i \mathbf{w}_j^\top \boldsymbol{\phi}_j(z_j^i, \mathbf{z}_{\pi(j)}^i) - A(\mathbf{w}_j, \mathbf{z}_{\pi(j)}^i) \tag{7.4}$$

where $\boldsymbol{\phi}_j(z_j^i, \mathbf{z}_{\pi(j)}^i)$ denotes a vector of features evaluated on the value of the child variable Z_j and its parent variables for instance i, such that $\boldsymbol{\phi}_j(z_j^i, \mathbf{z}_{\pi(j)}^i) = (\ldots 1_{(z_j^i=a, \mathbf{z}_{\pi(j)}^i=\mathbf{b})} \ldots)^\top$. A regularized version of the joint EM optimization problem (7.3) can then be formulated as

$$\begin{aligned} &\min_{\mathbf{y}} \min_{\mathbf{w}} \sum_j \left(\sum_i A(\mathbf{w}_j, \mathbf{z}_{\pi(j)}^i) - \mathbf{w}_j^\top \boldsymbol{\phi}_j(z_j^i, \mathbf{z}_{\pi(j)}^i) \right) + \frac{\beta}{2} \mathbf{w}_j^\top \mathbf{w}_j \\ = &\min_{\mathbf{y}} \sum_j \min_{\mathbf{w}_j} \left(\sum_i A(\mathbf{w}_j, \mathbf{z}_{\pi(j)}^i) - \mathbf{w}_j^\top \boldsymbol{\phi}_j(z_j^i, \mathbf{z}_{\pi(j)}^i) \right) + \frac{\beta}{2} \mathbf{w}_j^\top \mathbf{w}_j \end{aligned} \tag{7.5}$$

since the regularized likelihood decomposes into an independent sum over the local parameters $\mathbf{w}_j$ for each local variable Z_j.

As discussed in the previous section, a direct convexification of (7.5) based only on relaxing the discrete assignment $\mathbf{y}$ can only obtain vacuous uniform solutions. Instead one must derive an equivalent formulation in terms of equivalence relations over the assignments to hidden variables, instead of the hidden variable assignments themselves. This requires a fundamental reformulation of the optimization problem (7.5).

To derive the reformulation, initially it will be easier to work with an individual term in (7.5) corresponding to an arbitrary local variable Z_j

$$\min_{\mathbf{w}_j} \left(\sum_i A(\mathbf{w}_j, \mathbf{z}_{\pi(j)}^i) - \mathbf{w}_j^\top \boldsymbol{\phi}_j(z_j^i, \mathbf{z}_{\pi(j)}^i) \right) + \frac{\beta}{2} \mathbf{w}_j^\top \mathbf{w}_j \tag{7.6}$$

Note that this objective still depends on $\mathbf{y}$ since $\mathbf{z}^i = (\mathbf{x}^i, \mathbf{y}^i)$. Also this regularized form of the objective corresponds to the maximum a posteriori (MAP) parameter estimation with Gaussian priors on the parameters. Given complete data, solving this local parameter estimation amounts to solving a logistic regression problem. In fact, (7.6) was the same form as the regularized logistic regression form (3.3) in Chapter 3.

7.4.1 Logistic Regression on Equivalence Relations

A key subproblem is reformulating each of the local logistic regression problems (7.6) to drop the dependence on hidden variable assignments $\mathbf{y}$, and instead develop a form that can be expressed strictly in terms of equivalence relations over hidden variable assignments.

To simplify the notation, consider one of the local logistic regression problems (7.6) and drop the subscript j. To further simplify the notation, I also use a matrix notation to

rewrite the the jth local parameter optimization problem as follows

$$\min_{W} \Big(\sum_{i} A(W, \Phi_{i:}) \Big) - \text{tr}(\Phi W Y^{\top}) + \frac{\beta}{2} \text{tr}(W^{\top} W) \tag{7.7}$$

where $W \in I\!R^{F \times V}$, $\Phi \in \{0,1\}^{N \times F}$, and $Y \in \{0,1\}^{N \times V}$, such that N is the number of training instances, V is the number of possible values for the child variable, and F is the number of possible configurations for the parent variables. Here, the notation $\Phi_{i:}$ denotes the ith row vector in a matrix Φ such that $\Phi_{if} = \phi_f(z^i, \mathbf{z}^i_\pi)$ for the ith training instance and fth feature. The log normalization factor is given by

$$A(W, \Phi_{i:}) = \log \sum_{a} \exp\left(\Phi_{i:} W \mathbf{1}_a\right) \tag{7.8}$$

where $\mathbf{1}_a$ denotes a sparse vector with a single 1 in position a. To explain this notation, note that Y and Φ are indicator matrices that have a single 1 in each row, where Y indicates the value of the child variable, and Φ indicates the specific configuration of the parent values, respectively; i.e. $Y\mathbf{1} = \mathbf{1}$ and $\Phi\mathbf{1} = \mathbf{1}$, where $\mathbf{1}$ denotes the vector of all 1s. This matrix notation greatly streamlines the presentation below.

The first step in reformulating (7.7) in terms of equivalence relations is to derive its dual. The derivation follows the same step as deriving the dual (3.10) of the logistic regression (3.3) in Chapter 3.

Lemma 7.2 *An equivalent optimization problem to (7.7) is*

$$\begin{aligned} \max_{\Theta} \quad & -tr(\Theta \log \Theta^{\top}) - \frac{1}{2\beta} tr\Big((Y - \Theta)^{\top} \Phi \Phi^{\top} (Y - \Theta)\Big) \\ \textit{subject to} \quad & \Theta \geq 0, \;\; \Theta\mathbf{1} = \mathbf{1} \end{aligned} \tag{7.9}$$

Proof: The proof follows the same argument given in Chapter 3. I outline the steps here to make the derivation clear in the new notation. The dual optimization problem is derived by first considering the Fenchel conjugate function of $A(W, \Phi_{i:})$, given by

$$A^*(U_i, \Phi_{i:}) = \sup_{W} \; \text{tr}(U_i^{\top} W) - A(W, \Phi_{i:}) \tag{7.10}$$

A more convenient representation of the conjugate function can be obtained by solving the optimization (7.10). From (7.8), it is not hard to show that

$$\begin{aligned} \nabla_W A(W, \Phi_{i:}) &= \mathrm{E}_a \left[\Phi_{i:}^{\top} \mathbf{1}_a^{\top} \right] \\ &= \Phi_{i:}^{\top} \Theta_{i:} \end{aligned}$$

for some row vector $\Theta_{i:}$ such that $\Theta_{i:} \geq 0$ and $\Theta_{i:}\mathbf{1} = 1$. Setting the derivation of the left side of (7.10) with respect to W to zero then yields

$$U_i = \Phi_{i:}^\top \Theta_{i:} \tag{7.11}$$

Thus, from [96, Theorem 2] as in (3.5), one obtains

$$A^*(U_i, \Phi_{i:}) = \begin{cases} \Theta_{i:} \log \Theta_{i:}^\top & \text{if } U_i \in \mathcal{M}_{\Phi_{i:}} \\ \infty & \text{otherwise} \end{cases}$$

where $\Theta_{i:}$ is given by (7.11). Since $A(W, \Phi_{i:})$ is a closed convex function, it follows [6, Theorem 4.2.1] that the conjugate of the conjugate is the original function, and therefore

$$A(W, \Phi_{i:}) = \sup_{\Theta_{i:} \in \mathcal{S}_i} \operatorname{tr}(\Theta_{i:}^\top \Phi_{i:} W) - \Theta_{i:} \log \Theta_{i:}^\top \tag{7.12}$$

where $\mathcal{S}_i$ denotes the probability simplex

$$\mathcal{S}_i = \{\Theta_{i:} : \Theta_{i:} \geq 0, \Theta_{i:}\mathbf{1} = 1\}$$

Substituting (7.12) into (7.7) yields

$$\begin{aligned}
& \inf_W \Big(\sum_i A(W, \Phi_{i:}) \Big) - \operatorname{tr}(\Phi W Y^\top) + \frac{\beta}{2} \operatorname{tr}(W^\top W) \\
= & \inf_W \Big(\sum_i \sup_{\Theta_{i:} \in \mathcal{S}_i} \operatorname{tr}(\Theta_{i:}^\top \Phi_{i:} W) - \Theta_{i:} \log \Theta_{i:}^\top \Big) - \operatorname{tr}(\Phi W Y^\top) + \frac{\beta}{2} \operatorname{tr}(W^\top W) \\
= & \inf_W \sup_{\Theta \in \mathcal{S}} -\operatorname{tr}(\Theta \log \Theta^\top) + \operatorname{tr}(\Phi W \Theta^\top) - \operatorname{tr}(\Phi W Y^\top) + \frac{\beta}{2} \operatorname{tr}(W^\top W) \\
= & \inf_W \sup_{\Theta \in \mathcal{S}} G(W, \Theta)
\end{aligned}$$

where

$$G(W, \Theta) = -\operatorname{tr}(\Theta \log \Theta^\top) - \operatorname{tr}((Y - \Theta)^\top \Phi W) + \frac{\beta}{2} \operatorname{tr}(W^\top W)$$

and $\mathcal{S} = \prod_i \mathcal{S}_i$. It is easy to verify that $\mathcal{S}$ is closed and bounded and subsequently that G satisfies the conditions of Theorem 3.1 (Strong Minmax Property) in Chapter 3. Therefore, it follows that

$$\begin{aligned}
& \inf_W \sup_{\Theta \in \mathcal{S}} G(W, \Theta) \\
= & \sup_{\Theta \in \mathcal{S}} \inf_W -\operatorname{tr}(\Theta \log \Theta^\top) - \operatorname{tr}((Y - \Theta)^\top \Phi W) + \frac{\beta}{2} \operatorname{tr}(W^\top W)
\end{aligned} \tag{7.13}$$

The inner optimization can be solved by setting

$$W = \frac{1}{\beta} \Phi^\top (Y - \Theta) \tag{7.14}$$

Substituting this into (7.13) yields the result. ∎

Interestingly, deriving the dual has already achieved part of the desired result: the parent configurations now only enter the problem through the kernel matrix $K = \Phi\Phi^\top$. For Bayesian networks this kernel matrix is in fact an equivalence relation between parent configurations. To see this, note that Φ is a $\{0,1\}$ indicator matrix with a single 1 in each row, implying that $K_{i\ell} = 1$ if $\Phi_{i:} = \Phi_{\ell:}$, and $K_{i\ell} = 0$ otherwise. It is then straightforward to verify that K satisfies the reflexivity, symmetry and transitivity properties, and therefore encodes an equivalence relation. But more importantly, K can be re-expressed as a function of the individual equivalence relations on each of the parent variables. Let $Y^p \in \{0,1\}^{N \times V_p}$ indicate the values of a parent variable Z_p for the training instances. That is, $Y^p_{i:}$ is a $1 \times V_p$ sparse row vector with a single 1 indicating the value of variable Z_p in instance i. Then $M^p = Y^p Y^{p\top}$ defines an equivalence relation over the assignments to variable Z_p, since $M^p_{i\ell} = 1$ if $Y^p_{i:} = Y^p_{\ell:}$ and $M^p_{i\ell} = 0$ otherwise. It is not hard to see that the equivalence relation over complete parent configurations, $K = \Phi\Phi^\top$, is equal to the componentwise (Hadamard) product of the individual equivalence relations for each parent variable. That is, $K = \Phi\Phi^\top = M^1 \circ M^2 \circ \cdots \circ M^p$, since $K_{i\ell} = 1$ iff $M^1_{i\ell} = 1$ and $M^2_{i\ell} = 1$ and ... $M^p_{i\ell} = 1$.

Unfortunately, the dual problem (7.9) is still expressed in terms of the indicator matrix Y over the child variable values. The training problem still has to be reformulated in terms of the equivalence relation matrix $M = YY^\top$. Towards this goal, consider an alternative dual parameterization $\Omega \in \mathbb{R}^{N \times N}$ such that $\Omega \geq 0$, $\Omega\mathbf{1} = \mathbf{1}$, and

$$\Omega Y = \Theta$$

Note that $\Theta \in \mathbb{R}^{N \times V}$, for $V < N$, and therefore Ω is larger than Θ. Nevertheless, if Y is full rank (V), then in fact the two dual parameterizations, Θ and Ω, are equivalent. To see this, note that if $\Omega \geq 0$ and $\Omega\mathbf{1} = \mathbf{1}$, then setting $\Theta = \Omega Y$ implies $\Theta \geq 0$ and $\Theta\mathbf{1} = \mathbf{1}$, since $Y \in \{0,1\}^{N \times V}$ and $Y\mathbf{1} = \mathbf{1}$. Similarly, since Y is full rank, for every Θ such that $\Theta \geq 0$ and $\Theta\mathbf{1} = \mathbf{1}$, there exists some Ω such that $\Omega \geq 0$ and $\Omega\mathbf{1} = \mathbf{1}$ and $\Omega Y = \Theta$. If Y is not full rank, there must be some child value that never occurs in the training set. Then the number of effective values for child variable can be reduced while Y is simultaneously reduced until Y becomes full rank again without affecting the objective (7.7). Therefore one can relate the primal parameters to this larger set of dual parameters by replacing (7.14) with the relation

$$W = \frac{1}{\beta}\Phi^\top(I - \Omega)Y \tag{7.15}$$

Even though Ω is larger than Θ, they can only express the same realizable set of parameters W through the equations (7.14) and (7.15). To simplify the notation, let

$$Q = I - \Omega$$

Therefore $Q \leq I$, $Q\mathbf{1} = \mathbf{0}$, and

$$W = \frac{1}{\beta}\Phi^\top QY \tag{7.16}$$

By first deriving the reformulation (7.9) given in Lemma 7.2 and then making the substitution (7.16), one can show that an equivalent optimization problem to (7.7) is given by

$$\begin{aligned} \min_{Q} \quad & \Big(\sum_i A(Q, \Phi_{i:})\Big) - \frac{1}{\beta}\mathrm{tr}(KQM) + \frac{1}{2\beta}\mathrm{tr}(Q^\top KQM) \\ \text{subject to} \quad & Q \leq I, \;\; Q\mathbf{1} = \mathbf{0} \end{aligned} \tag{7.17}$$

where $K = \Phi\Phi^\top$ and $M = YY^\top$ are equivalence relations on the parent configurations and child values respectively. The formulation (7.17) is now almost completely expressed in terms of equivalence relations over the data, except for one subtle problem: the log normalization factor

$$A(Q, \Phi_{i:}) = \log \sum_a \exp\left(\frac{1}{\beta}\Phi_{i:}\Phi^\top QY\mathbf{1}_a\right)$$

still directly depends on the label indicator matrix Y. The next key technical lemma is that this log normalization factor can be re-expressed to depend on the equivalence relation matrix M alone.

Lemma 7.3

$$A(Q, \Phi_{i:}) = \log \sum_\ell \exp\left(\frac{1}{\beta}K_{i:}QM_{:\ell} - \log \mathbf{1}^\top M_{:\ell}\right) \tag{7.18}$$

Proof: The main observation I exploit is that an equivalence relation over value indicators, defined by $M = YY^\top$, must consist of columns that are copied from Y. That is, for all ℓ, $M_{:\ell} = Y_{:a}$ where a is the child variable value for instance ℓ. Therefore $\mathbf{1}^\top M_{:\ell}$ is the number of instances that have the same child value as instance ℓ. Let $y(\ell)$ denote the child variable value for instance ℓ. One then can derive

$$
\begin{aligned}
A(Q, \Phi_{i:}) &= \log \sum_a \exp\left(\frac{1}{\beta}\Phi_{i:}\Phi^\top Q Y \mathbf{1}_a\right) \\
&= \log \sum_a \exp\left(\frac{1}{\beta}K_{i:}QY_{:a}\right) \\
&= \log \sum_a \sum_{\ell:y(\ell)=a} \frac{1}{\mathbf{1}^\top M_{:\ell}} \exp\left(\frac{1}{\beta}K_{i:}QM_{:\ell}\right) \\
&= \log \sum_\ell \frac{1}{\mathbf{1}^\top M_{:\ell}} \exp\left(\frac{1}{\beta}K_{i:}QM_{:\ell}\right) \\
&= \log \sum_\ell \exp\left(\frac{1}{\beta}K_{i:}QM_{:\ell} - \log \mathbf{1}^\top M_{:\ell}\right) \blacksquare
\end{aligned}
$$

Substituting (7.18) into (7.17) successfully achieves the goal of reformulating the original problem (7.7) strictly in terms of equivalence relations over the hidden variable values. This reformation gives an equivalent result to the original training problem (7.7), but by eliminating Y, it is no longer subject to the triviality outcome established by Lemma 7.1 in Section 7.3. However, the final goal of effectively convexifying the joint EM has not been accomplished yet. Note it is easy to see the objective of (7.17) is not jointly convex in M and Q (or K and Q), which means (7.17) is non-convex in M (or K). Nevertheless, a convex form in M (or K) can be obtained by taking the dual of (7.17).

To derive the dual, first consider the Fenchel conjugate of $A(Q, \Phi_{i:})$.

Lemma 7.4

$$
A(Q, \Phi_{i:}) = \max_{\Lambda_{i:} \geq 0,\ \Lambda_{i:}\mathbf{1}=\mathbf{1}} \quad \frac{1}{\beta}K_{i:}QM\Lambda_{i:}^\top - \Lambda_{i:}\log\Lambda_{i:}^\top - \Lambda_{i:}\log(M\mathbf{1}) \tag{7.19}
$$

Proof: Let $\Upsilon_{i\ell} = \frac{1}{\beta}K_{i:}QM_{:\ell} - \log \mathbf{1}^\top M_{:\ell}$, thus $A(Q, \Phi_{i:}) = \log\sum_\ell \exp(\Upsilon_{i\ell})$ by Lemma 7.3. Then the Fenchel conjugate of $A(Q, \Phi_{i:})$ is defined by

$$
A^*(U_i, \Phi_{i:}) = \sup_Q \ \mathrm{tr}(U_i^\top Q) - A(Q, \Phi_{i:}) \tag{7.20}
$$

The maximum of the left side of (7.20) is achieved by setting its gradient with respect to Q to zero, yielding

$$
U_i \ - \ \sum_\ell \frac{\exp(\Upsilon_{i\ell})}{\sum_{\ell'}\exp(\Upsilon_{i\ell'})}\frac{1}{\beta}K_{i:}^\top M_{:\ell}^\top = 0
$$

which implies

$$
U_i \ = \ \sum_\ell \Lambda_{i\ell}\frac{1}{\beta}K_{i:}^\top M_{:\ell}^\top \tag{7.21}
$$

where $\Lambda_{i\ell} = \frac{\exp(\Upsilon_{i\ell})}{\sum_{\ell'}\exp(\Upsilon_{i\ell'})}$, and $\Lambda_{i:} \geq 0, \Lambda_{i:}\mathbf{1} = 1$. Thus $\sum_{\ell'}\exp(\Upsilon_{i\ell'}) = \frac{\exp(\Upsilon_{i\ell})}{\Lambda_{i\ell}}$ and $A(Q,\Phi_{i:}) = \Upsilon_{i\ell} - \log\Lambda_{i\ell}$ for all ℓ. Substituting (7.21) back into (7.20), one obtains $\mathrm{tr}(U_i^\top Q) = \sum_\ell \Lambda_{i\ell}\frac{1}{\beta}K_{i:}QM_{:\ell}$ and therefore

$$\begin{aligned} A^*(U_i,\Phi_{i:}) &= \sum_\ell \Lambda_{i\ell}\left(\frac{1}{\beta}K_{i:}QM_{:\ell} - A(Q,\Phi_{i:})\right) \\ &= \sum_\ell \Lambda_{i\ell}\left(\frac{1}{\beta}K_{i:}QM_{:\ell} - \Upsilon_{i\ell} + \log\Lambda_{i\ell}\right) \\ &= \Lambda_{i:}\log(M\mathbf{1}) + \Lambda_{i:}\log\Lambda_{i:}^\top \end{aligned}$$

Since $A(Q,\Phi_{i:})$ is a closed convex function, it follows [6] that the conjugate of the conjugate of A is the original function, hence

$$\begin{aligned} A(Q,\Phi_{i:}) &= \max_{\Lambda_{i:}\geq 0,\Lambda_{i:}\mathbf{1}=1} \mathrm{tr}(U_i^\top Q) - A^*(U_i,\Phi_{i:}) \\ &= \max_{\Lambda_{i:}\geq 0,\Lambda_{i:}\mathbf{1}=1} \frac{1}{\beta}K_{i:}QM\Lambda_{i:}^\top - \Lambda_{i:}\log\Lambda_{i:}^\top - \Lambda_{i:}\log(M\mathbf{1}) \end{aligned}$$

∎

This Lemma then allows me to derive the main result of this section.

Theorem 7.1 *An equivalent optimization problem to (7.7) is*

$$\max_{\Lambda\geq 0,\Lambda\mathbf{1}=\mathbf{1}} -tr(\Lambda\log\Lambda^\top) - \mathbf{1}^\top\Lambda\log(M\mathbf{1}) - \frac{1}{2\beta}tr((I-\Lambda)^\top K(I-\Lambda)M) \qquad (7.22)$$

where $K = M^1 \circ \cdots \circ M^p$ for parent variables $Z_1, ..., Z_p$.

Proof: It has already been established above that (7.17) is equivalent to (7.7). By substituting (7.19) into (7.17), one then obtains that (7.17) is equivalent to

$$\begin{aligned} &\min_{Q\leq I, Q\mathbf{1}=\mathbf{0}} \left(\sum_i \max_{\Lambda_{i:}\geq 0,\Lambda_{i:}\mathbf{1}=1} \frac{1}{\beta}K_{i:}QM\Lambda_{i:}^\top - \Lambda_{i:}\log\Lambda_{i:}^\top - \Lambda_{i:}\log(M\mathbf{1})\right) \\ &\qquad -\frac{1}{\beta}\mathrm{tr}(KQM) + \frac{1}{2\beta}\mathrm{tr}(Q^\top KQM) \\ = &\min_{Q\leq I, Q\mathbf{1}=\mathbf{0}} \max_{\Lambda\geq 0,\Lambda\mathbf{1}=\mathbf{1}} G(Q,\Lambda) \end{aligned}$$

where

$$\begin{aligned} G(Q,\Lambda) &= -\mathrm{tr}(\Lambda\log\Lambda^\top) - \mathbf{1}^\top\Lambda\log(M\mathbf{1}) \\ &\quad -\frac{1}{\beta}\mathrm{tr}((I-\Lambda)^\top KQM) + \frac{1}{2\beta}\mathrm{tr}(Q^\top KQM) \end{aligned} \qquad (7.23)$$

Here the feasible regions are closed and bounded, and also that G satisfies the conditions of Theorem 3.1 (Strong Minmax Property). Therefore, it follows that

$$\begin{aligned} &\min_{Q\leq I, Q\mathbf{1}=\mathbf{0}} \max_{\Lambda\geq 0,\Lambda\mathbf{1}=\mathbf{1}} G(Q,\Lambda) \\ = &\max_{\Lambda\geq 0,\Lambda\mathbf{1}=\mathbf{1}} \min_{Q\leq I, Q\mathbf{1}=\mathbf{0}} G(Q,\Lambda) \end{aligned} \qquad (7.24)$$

Taking the gradient of G defined in (7.23) with respect to Q yields

$$\nabla_Q G(Q, \Lambda) = \frac{1}{\beta} K[Q - (I - \Lambda)]M$$

Note that setting $Q = I - \Lambda$ forces the gradient to zero and also satisfies the constraints over Q, since $\Lambda \geq 0$ and $\Lambda \mathbf{1} = \mathbf{1}$. Therefore $Q = I - \Lambda$ is a solution to the inner minimization in (7.24). Making this substitution in (7.24) yields (7.22). (Note Λ happens to be same as the dual parameter Ω introduced before.) ∎

Theorem 7.1 gives my key result. The dual formulation (7.22) is equivalent to the original logistic regression problem (7.7). However, it is now expressed strictly in terms of equivalence relations over the parent configurations (K) and child values (M). That is, the value indicators, Φ and Y, have been successfully eliminated from the formulation. Furthermore, (7.22) is a pointwise maximum function of M (or K) and is therefore convex in M (or K, but not both jointly; see below) [8], which will allow me to derive a convenient convex relaxation of the joint EM training problem below.

7.4.2 Convex Relaxation of EM

By exploiting Theorem 7.1, I can now re-express the regularized form of joint EM objective (7.5) strictly in terms of equivalence relations over the hidden variable values

$$\min_{\{Y^h\}} \sum_j \min_{\mathbf{w}_j} \sum_i A(\mathbf{w}_j, \mathbf{z}^i_{\pi(j)}) - \mathbf{w}_j^\top \boldsymbol{\phi}_j(z^i_j, \mathbf{z}^i_{\pi(j)}) + \frac{\beta}{2}\mathbf{w}_j^\top \mathbf{w}_j \tag{7.25}$$

$$= \min_{\{M^h\}} \sum_j \max_{\Lambda_j} -\text{tr}(\Lambda_j \log \Lambda_j^\top) - \mathbf{1}^\top \Lambda_j \log(M^j \mathbf{1}) \tag{7.26}$$

$$-\frac{1}{2\beta}\text{tr}\left((I - \Lambda_j)^\top K_j (I - \Lambda_j) M^j\right)$$

$$\text{subject to } \Lambda_j \geq 0, \Lambda_j \mathbf{1} = \mathbf{1}, \text{for all } j$$

$$M^h = Y^h Y^{h\top}, Y^h \in \{0,1\}^{N \times V_h}, Y^h \mathbf{1} = \mathbf{1}, \text{for all } h \tag{7.27}$$

where h ranges over the hidden variables, and $K_j = M^{j_1} \circ \cdots \circ M^{j_p}$ for the parent variables $Z_{j_1}, ..., Z_{j_p}$ of Z_j.

Note that (7.5) is equivalent to (7.25), hence equivalent to (7.26); that is, no approximation has been introduced to this point. The objective of (7.26) is concave in each Λ_j and convex in each M^h individually (a maximum of convex functions is convex [8]). Therefore, (7.26) appears as though it might admit an efficient algorithmic solution. However, one difficulty in solving the resulting optimization problem is the constraints imposed in (7.27). These constraints are not convex. Therefore, to obtain a convex formulation some

form of relaxation appears to be required. One natural convex relaxation is suggested by the following.

Lemma 7.5 *The constraints (7.27) are equivalent to:* $M^h \in \{0,1\}^{N\times N}$, *diag*$(M^h) = \mathbf{1}$, $M^h = M^{h\top}$, $M^h \succeq 0$, *rank*$(M^h) = V_h$.

Thus a natural convex relaxation of (7.27) can be obtained by relaxing the discreteness constraint and dropping the non-convex rank constraint, yielding

$$M^h \in [0,1]^{N\times N}, \text{diag}(M^h) = \mathbf{1}, M^h = M^{h\top}, M^h \succeq 0 \tag{7.28}$$

Optimizing the exact objective in (7.26) subject to the relaxed convex constraints (7.28) provides the foundation for convexifying joint EM. Note that since (7.26) and (7.28) are expressed solely in terms of equivalence relations, and do not depend on the specific values of hidden variables in any way, this formulation is not subject to the triviality result of Lemma 7.1.

However, there are still some details left to consider. First, if there is only a single hidden variable then (7.26) is convex with respect to the single matrix variable M^h. This result immediately provides a convex joint EM training algorithm for various applications, such as naive Bayes for classification, for example. Second, if there are multiple hidden variables that are separated from each other (none are neighbors, nor share a common child) then the formulation (7.26) remains convex and can be directly applied. On the other hand, if hidden variables are connected in any way, either by sharing a parent-child relationship or having a common child, then (7.26) is no longer jointly convex because the trace term is no longer linear in the matrix variables $\{M^h\}$. In this case, one can restore convexity by further relaxing the problem. To illustrate, if there are multiple hidden parents $Z_{p_1}, ..., Z_{p_k}$ for a given child, then the combined equivalence relation $M^{p_1} \circ \cdots \circ M^{p_k}$ is a Hadamard product of the individual matrices. A convex formulation can be recovered by introducing an auxiliary matrix variable $\tilde{M}$ to replace $M^{p_1} \circ \cdots \circ M^{p_k}$ in (7.26) and adding the set of linear constraints $\tilde{M}_{i\ell} \leq M^p_{i\ell}$ for $p \in \{p_1, ..., p_k\}$, $\tilde{M}_{i\ell} \geq M^{p_1}_{i\ell} + \cdots + M^{p_k}_{i\ell} - k + 1$ to approximate the componentwise 'and'. However, when a child variable and one of its parent variables are both hidden, a more complex relaxation has to be developed (I leave this as part of future work). To conduct experiments, I implemented a barrier approach for minimizing (7.26) subject to (7.28) based on BFGS (Broyden-Fletcher-Goldfarb-Shanno) optimization method [70]. See [8] for a detailed description of the standard barrier optimization, which is also described in Chapter 6 of this thesis. For the problem here, I used log barriers for the linear inequality constraints and log-determinant barriers for the semidefinite constraints.

Recovering the Model Parameters Once the relaxed equivalence relation matrices $\{M^h\}$ have been obtained, the parameters of the underlying probability model need to be recovered. At an optimal solution to (7.26), one not only obtains $\{M^h\}$, but also the associated set of dual parameters $\{\Lambda_j\}$. Therefore, one can recover the primal parameters W_j from the dual parameters Λ_j using the relationship $W_j = \frac{1}{\beta}\Phi_j^\top(I - \Lambda_j)Y^j$ established above, which only requires availability of a label assignment matrix Y^j. For observed variables, Y^j is known, and therefore the model parameters can be immediately recovered. For hidden variables, we first need to compute a rank V_h factorization of M^h. Let $S = T\Sigma^{1/2}$ where T and Σ are the top V_h eigenvector and eigenvalue matrices of the centered matrix HM^hH. One typical way to recover $\hat{Y}^h$ from S is to run k-means on the rows of S and construct the indicator matrix. A more elegant approach would be to use a randomized rounding scheme [37], which also produces a deterministic $\hat{Y}^h$, but provides some guarantees about how well $\hat{Y}^h\hat{Y}^{h\top}$ approximates M^h.

7.5 Experimental Results

An important question to ask is whether the relaxed, convex objective (7.26) is in fact over-relaxed, and whether important structure in the original objective (7.25) has been lost as a result. To investigate this question, I conducted a set of experiments to evaluate the proposed convex approach compared to the standard joint EM algorithm, and to supervised training on fully observed data. Whenever possible, I also compared with the golden standard true model. My experiments are conducted using both synthetic Bayesian networks and real world networks, while evaluating the trained models by the logloss (negative loglikelihood) they produced on the fully observed training data and testing data. All the results reported in this chapter are averaged over 10 repetitions. The test size for the experiments is 1000, and the training size is 100. For a fair comparison, I used 10 random restarts for the standard joint EM algorithm to help avoid poor local optima.

For the synthetic experiments, I constructed three Bayesian networks, shown in Figure 7.1: BN1 is a three layer network with 9 variables, where the two nodes in the middle layer are picked as hidden variables; BN2 is a network with 6 variables and 6 edges, where a node with 2 parents and 2 children is picked as hidden variable; BN3 is a naive Bayes model with 8 variables, where the parent node is selected as the hidden variable. The parameters are generated in a discriminative way to produce models with apparent causal relations between the connected variables. I then conducted experiments on these three synthetic

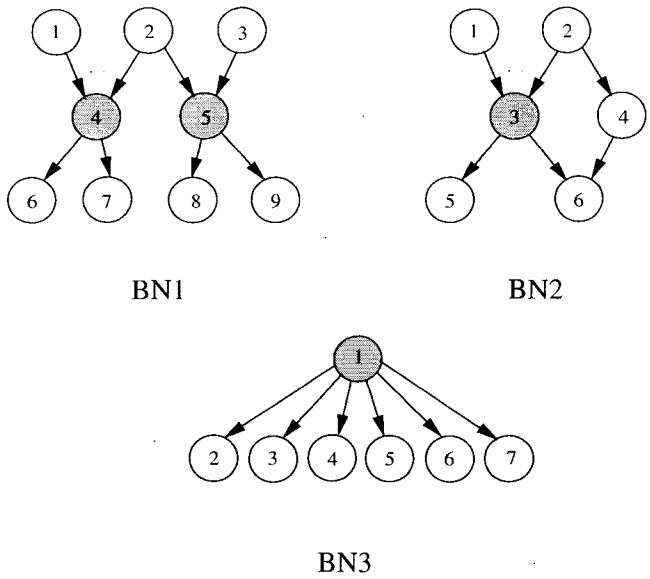


Figure 7.1: Synthetic Bayesian networks 1–3

Table 7.1: Experimental results on synthetic Bayesian networks

	Golden Standard Train Logloss	Fully Supervised Train Logloss	Joint EM Train Logloss	Convex EM Train Logloss
BN1	7.52 ± 0.06	7.23 ± 0.06	11.29 ± 0.44	8.96 ± 0.24
BN2	4.34 ± 0.04	4.24 ± 0.04	6.06 ± 0.20	5.23 ± 0.18
BN3	5.09 ± 0.02	4.93 ± 0.02	7.81 ± 0.35	6.23 ± 0.18
	Test Logloss	Test Logloss	Test Logloss	Test Logloss
BN1	7.48 ± 0.01	7.90 ± 0.04	11.73 ± 0.38	9.16 ± 0.21
BN2	4.37 ± 0.01	4.50 ± 0.03	6.48 ± 0.23	5.48 ± 0.19
BN3	5.10 ± 0.01	5.32 ± 0.05	8.18 ± 0.33	6.41 ± 0.14

networks. The results are reported in Table 7.1. One can see that the convex relaxation was successful at preserving structure in the EM objective, and in fact, generally performed much better than the standard joint EM algorithm, particularly in the case (BN1) where there was two hidden variables. Not surprisingly, supervised training on the complete data performed better than the EM methods, but generally demonstrated a larger gap between training and test losses than the EM methods.

In addition to these three synthetic Bayesian networks, I also ran experiments using some UCI data sets. Here I used naive Bayes as the model structure, and set the class variable to be hidden. The UCI results are reported in Table 7.2. The results I obtained in this case are mixed: the convex EM algorithm performed better than the joint EM on four data sets—Australian, Diabetes, Flare and Pima—while worse on the other four data sets—Breast, Cleve, Crx and Heart. One has to admit that it is possible for the joint EM to converge to a better solution in some cases. Further investigation needs to be conducted with respect to what aspects of the problem are responsible for the weak approximation given by the convex EM in such cases.

Finally, I conducted additional experiments on three real world Bayesian networks: Alarm, Cancer and Asian (downloaded from http://www.norsys.com/networklibrary.html). I selected one well connected node from each model to serve as the hidden variable, and generated data by sampling from the models. Table 7.3 shows the experimental results for these three Bayesian networks. Here one can see that the convex EM relaxation performed well on the Cancer and Alarm networks, though the advantage is very small for the Alarm network. Since I only selected one hidden variable from the 37 variables in Alarm, it is understandable that any potential advantage for the convex approach might not be large. Much weaker results are obtained on the Asian network however. The reason remains to be further investigated.

7.6 Conclusion

In this chapter, I have presented a novel convex relaxation of the standard joint EM (Viterbi EM) algorithm for Bayesian network parameter estimation in the presence of hidden variables. This convex EM approach was facilitated by a novel reformulation of logistic regression that refers only to equivalence relation information on the hidden variable values, and thereby allows one to avoid the symmetry breaking problem that blocks naive convexification strategies from working. Experimental results, that compared this convex EM

Table 7.2: Experimental results on UCI data sets

	Fully Supervised Train Logloss	Joint EM Train Logloss	Convex EM Train Logloss
Australian	10.32 ± 0.07	12.83 ± 0.21	11.92 ± 0.23
Breast	4.70 ± 0.10	4.86 ± 0.13	6.06 ± 0.28
Cleve	8.17 ± 0.08	8.64 ± 0.14	9.03 ± 0.21
Crx	11.35 ± 0.07	13.35 ± 0.40	13.45 ± 0.19
Diabetes	5.23 ± 0.04	6.70 ± 0.27	6.51 ± 0.35
Flare	5.96 ± 0.06	11.79 ± 0.26	7.36 ± 0.37
Heart	8.11 ± 0.05	8.56 ± 0.11	8.93 ± 0.15
Pima	5.07 ± 0.03	6.74 ± 0.34	5.81 ± 0.07
	Test Logloss	Test Logloss	Test Logloss
Australian	11.05 ± 0.04	13.57 ± 0.09	12.34 ± 0.22
Breast	4.92 ± 0.03	5.02 ± 0.04	6.30 ± 0.27
Cleve	8.51 ± 0.05	9.05 ± 0.14	9.15 ± 0.14
Crx	12.18 ± 0.05	13.72 ± 0.29	13.91 ± 0.23
Diabetes	5.53 ± 0.04	7.07 ± 0.23	6.50 ± 0.28
Flare	6.46 ± 0.04	12.11 ± 0.20	7.82 ± 0.44
Heart	8.48 ± 0.03	8.91 ± 0.07	9.09 ± 0.14
Pima	5.32 ± 0.03	6.93 ± 0.21	6.03 ± 0.09

Table 7.3: Experimental results on real-world Bayesian networks

	Golden Standard Train Logloss	Fully Supervised Train Logloss	Joint EM Train Logloss	Convex EM Train Logloss
Cancer	2.23 ± 0.05	2.18 ± 0.05	3.90 ± 0.31	2.98 ± 0.19
Alarm	11.14 ± 0.18	10.23 ± 0.16	11.94 ± 0.32	11.74 ± 0.25
Asian	2.24 ± 0.06	2.17 ± 0.05	2.21 ± 0.05	2.70 ± 0.14
	Test Logloss	Test Logloss	Test Logloss	Test Logloss
Cancer	2.24 ± 0.01	2.31 ± 0.02	3.94 ± 0.29	3.06 ± 0.16
Alarm	10.93 ± 0.06	12.30 ± 0.06	13.75 ± 0.17	13.62 ± 0.20
Asian	2.22 ± 0.01	2.33 ± 0.02	2.36 ± 0.03	2.78 ± 0.12

relaxation to the standard joint EM algorithm, on both synthetic and real world Bayesian networks, show this novel convex technique can perform more effectively than joint EM in some circumstances. However, some weak results also existed, suggesting weaker approximation qualities in those cases. The reason remains to be investigated.

Chapter 8

Conclusions

In this thesis, I have investigated a few challenging problems for learning Bayesian networks from data. Specifically, I presented five pieces of work motivated toward achieving better data modeling and pattern classification with Bayesian networks under various contexts: generative modeling, discriminative classification, and learning with hidden variables. In particular, I have exploited several convex optimization techniques to address the Bayesian network learning issues by first formulating a natural optimization problem and then relaxing it to a convex form whenever it is possible.

First, I presented a novel convex relaxation for generative Bayesian network structure learning. This approach simultaneously searches over variable orders, structure and parameters in a joint convex optimization by introducing a set of auxiliary feature selection variables. Compared to standard score-based heuristic search methods, which suffer from local optima, this convex approach suggests a new class of algorithms for learning Bayesian networks that ultimately might lead to guaranteed approximation quality. Beyond achieving approximation guarantees and algorithmic improvements, other significant directions for future research include considering the problem of structure learning in the presence of missing data or hidden variables, and attempting to extend the current analysis to BDe scores.

Second, following the idea of selection variable controlled structure learning, I then presented a globally regularized risk minimization method for inferring gene regulatory network structure from time series expression data. This method formulates structure inference as a feature selection problem. Exploiting the assumption that genes with similar expression patterns are likely to be co-regulated, the proposed approach learns the regulatory relationships using global feature selection to encourage genes with similar expression patterns to share regulators, while local feature selection is controlled by L1 regularization,

which allows individual differences. This framework also provides an opportunity to incorporate additional background knowledge, which can be considered in the future. Given that this approach appears to be an effective feature selection strategy, one future research direction is to extend this technique to other bioinformatics problems.

Third, considering that Bayesian networks have been popularly applied as classification tools, I presented two new discriminative model selection criteria (BV and CBIC) to guide the structure learning for Bayesian network classifiers with the goal of identifying the structure with the best classification performance. I conducted a comprehensive empirical study to compare the proposed discriminative criteria with standard criteria in various contexts. The proposed BV criterion turns out to perform best across most contexts. This work provides a useful reference for studying the discriminative Bayesian network structure learning problem in the future.

Fourth, I presented a discriminative maximum margin approach for Bayesian network classifier parameter estimation. This approach extends the most popular maximum margin criterion of SVMs to the classification setting using Bayesian networks. Although within this framework, maximum margin training is a hard computational problem, I still devised a reasonable convex relaxation to solve it more efficiently. The empirical study suggests that maximum margin Bayesian networks can be more effective for classification than maximum margin Markov networks, when the Bayesian network structure encodes the causal information in the underlying domain. In this sense, maximum margin Bayesian networks offer a new way to add prior knowledge to SVMs. The main directions for future research are to improve the training procedure and explore the possibility of using kernels in the local feature representation.

Finally, instead of considering complete training data, I presented a novel convex Viterbi EM algorithm for parameter estimation with hidden variables. To illustrate the challenges involved in effectively convexifying Viterbi EM, I showed that naive convexifications can only lead to vacuous results, due to the symmetry property of the configurations for hidden variables. Thus, I reformulated the Viterbi EM optimization problem in terms of equivalence relations over the hidden variable values instead of the hidden values themselves, which allows one to avoid the symmetry breaking problem that blocks naive convexification strategies from working. I then relaxed the objective to obtain a convex optimization problem. My preliminary results suggest this convex relaxation of EM obtains reasonable results in comparison to the standard Viterbi EM algorithm. So far, this work has not addressed the complicated case where the child variable and at least one parent variable are

both hidden. Deriving a reasonable convex relaxation in such case remains part of future work. Another further research direction is to investigate the approximation quality of the convex relaxation. Extending the current work to deal with Gaussian mixture models is also one part of the future research.

Most of the work in this thesis focused on discrete data. Therefore extending the proposed techniques to continuous data is a general direction for future research.

Overall, this thesis provides a broad study on learning Bayesian networks for generative data modeling and discriminative data classification. It presented novel Bayesian network learning approaches, often by exploring convex optimization techniques. This thesis enriches the literature on Bayesian network learning, and also provides some useful tools for application fields such as bioinformatics.

Bibliography

[1] Y. Altun, I. Tsochantaridis, and T. Hofmann. Hidden Markov support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[2] A. Bernard and A. Hartemink. Informative structure priors: Joint learning of dynamic regulatory networks from multiple types of data. *Pacific Symposium on Biocomputing*, pages 459–470, 2005.

[3] J. Bernardo and A. Smith. *Bayesian Theory*. Wiley, 1994.

[4] D. Bertsekas. *Nonlinear Optimization*. Athena Scientific, 1995.

[5] J. Binder, D. Koller, S. Russell, and K. Kanazawa. Adaptive probabilistic networks with hidden variables. *Machine Learning*, 29:213–244, 1997.

[6] J. Borwein and A. Lewis. *Convex Analysis and Nonlinear Optimization: Theory and Examples*. Springer, 2000.

[7] C. Boutilier, N. Friedman, M. Goldszmidt, and D. Koller. Context-specific independence in Bayesian networks. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, 1996.

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge U. Press, 2004.

[9] H. Bozdogan. Model selection and Akaike's Information Criterion (AIC): The general theory and its analytical extensions. *Psychometrica*, 52, 1987.

[10] W. Buntine. Theory refinement on Bayesian networks. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, 1991.

[11] K. Chen, T. Wang, H. Tseng, C. Huang, and C. Kao. A stochastic differential equation model for quantifying transcriptional regulatory network in Saccharomyces cerevisiae. *Bioinformatics*, 21:2883–2890, 2005.

[12] X. Chen, G. Anantha, and X. Wang. An effective structure learning method for constructing gene networks. *Bioinformatics*, 22:1367–1374, 2006.

[13] J. Cheng, D. Bell, and W. Liu. An algorithm for Bayesian belief network construction from data. In *Artificial Intelligence and Statistics*, 1997.

[14] J. Cheng and R. Greiner. Comparing Bayesian network classifiers. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

[15] M. Chickering. Learning Bayesian networks is NP-complete. In D. Fisher and H. Lenz, editors, *Learning from Data: Artificial Intelligence and Statistics*, volume 5. Springer, 1996.

[16] M. Chickering, C. Meek, and D. Heckerman. Large-sample learning of Bayesian networks is NP-hard. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, 2003.

[17] R. Cho, M. Campbell, E. Winzeler, L. Steinmetz, A. Conway, L. Wodicka, T. Wolfsberg, A. Gabrielian, D. Landsman, D. Lockhart, and R. Davis. A genome-wide transcriptional analysis of the mitotic cell cycle. *Molecular Cell*, 2:65–73, 1998.

[18] C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory*, 14:462–467, 1968.

[19] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2002.

[20] G. Cooper and E. Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9, 1992.

[21] T. Cover and J. Thomas. *Elements of Information Theory*. Wiley, 1991.

[22] K. Crammer and Y. Singer. On the algorithmic interpretation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2, 2001.

[23] J. Cuff and G. Barton. Evaluation and improvement of multiple sequence methods for protein secondary structure prediction. *Proteins: Structure, Function and Genetics*, 34:508–519, 1999.

[24] M. De Hoon, S. Imoto, K. Kobayashi, N. Ogasawara, and S. Miyano. Inferring gene regulatory networks from time-ordered gene expression data of bacillussubtilis using differential equations. *Pacific Symposium on Biocomputing*, pages 17–28, 2003.

[25] H. De Jong, J. Gouze, C. Hernandez, M. Page, T. Sari, and J. Geiselmann. Qualitative simulation of genetic regulatory networks using piecewise-linear models. *Bulletin Mathematical Biology*, 66:301–340, 2004.

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

[27] P. D'Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Linear modeling of mRNA expression levels during CNS development and injury. *Pacific Symposium on Biocomputing*, pages 41–52, 1999.

[28] P. D'Haeseleer, X. Wen, S. Fuhrman, and R. Somogyi. Genetic network inference: From co-expression clustering to reverse engineering. *Bioinformatics*, 16:707–726, 2000.

[29] R. Duda and P. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

[30] G. Elidan and N. Friedman. Learning hidden variable networks: The information bottleneck approach. *Journal of Machine Learning Research*, 6, 2005.

[31] G. Elidan, M. Ninio, N. Friedman, and D. Schuurmans. Data perturbation for escaping local maxima in learning. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, 2002.

[32] N. Friedman. Learning belief networks in the presence of missing values and hidden variables. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

[33] N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine Learning*, 29:131–163, 1997.

[34] N. Friedman and M. Goldszmidt. Learning Bayesian networks with local structure. In M. Jordan, editor, *Learning in Graphical Models*, pages 421–459. MIT Press, 1999.

[35] D. Geiger, D. Heckerman, and C. Meek. Asymptotic model selection for directed networks with hidden variables. Technical Report MSR-TR-96-07, Microsoft Research, 1996.

[36] Z. Ghahramani. *Graphical Models: Parameter Learning*. The Handbook of Brain Theory and Neural Networks (2nd edition), 2002.

[37] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[38] A. Goldenberg and A. Moore. Tractable learning of large Bayes net structure from sparse data. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[39] R. Greiner and W. Zhou. Structural extension to logistic regression: Discriminant parameter learning of belief net classifiers. In *Proceedings of the Eighteenth Annual National Conference on Artificial Intelligence*, 2002.

[40] D. Grossman and P. Domingos. Learning Bayesian network classifiers by maximizing conditional likelihood. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[41] Y. Guo and R. Greiner. Discriminative model selection for belief net structures. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 2005.

[42] Y. Guo and D. Schuurmans. Convex structure learning for Bayesian networks: Polynomial feature selection and approximate ordering. In *Proceedings of the Twenty-Second Conference on Uncertainty in Artificial Intelligence*, 2006.

[43] Y. Guo and D. Schuurmans. Convex relaxations of latent variable training. In *Advances in Neural Information Processing Systems 20*, 2007.

[44] Y. Guo and D. Schuurmans. Learning gene regulatory networks via globally regularized risk minimization. In *Proceedings of the Fifth Annual RECOMB Satellite Workshop on Comparative Genomics*, 2007.

[45] Y. Guo, D. Wilkinson, and D. Schuurmans. Maximum margin Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 2005.

[46] A. Hartemink, D. Gifford, T. Jaakkola, and R. Young. Using graphical models and genomic expression data to statistically validate models of genetic regulatory networks. *Pacific Symposium on Biocomputing*, pages 422–433, 2001.

[47] D. Heckerman. A tutorial on learning with Bayesian networks. In M. Jordan, editor, *Learning in Graphical Models*, pages 301–354. MIT Press, 1999.

[48] D. Heckerman, D. Geiger, and M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 9, 1995.

[49] I. Inza, P. Larranaga, J. Lozano, and J. Pena. Special issue of machine learning journal: Probabilistic graphical models for classification. *Machine Learning*, 59, 2005.

[50] V. Iyer, C. Horak, C. Scafe, D. Botstein, M. Snyder, and P. Brown. Genomic binding sites of the yeast cell-cycle transcription factors SBF and MBF. *Nature*, 409:533–8, 2001.

[51] M. Jordan. An introduction to probabilistic graphical models. Textbook in preparation, 2003.

[52] M. Kearns, Y. Mansour, A. Ng, and D. Ron. An experimental and theoretical comparison of model selection method. *Machine Learning*, 27, 1997.

[53] D. Klein and C. Manning. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the Forty-Second Annual Meeting of the Association for Computational Linguistics*, 2004.

[54] P. Kontkanen, P. Myllymäki, T. Silander, and H. Tirri. On supervised selection of Bayesian networks. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, 1999.

[55] J. Lafferty, Y. Liu, and X. Zhu. Kernel conditional random fields: Representation, clique selection, and semi-supervised learning. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[56] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 2001.

[57] W. Lam and F. Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10(4), 1994.

[58] G. Lanckriet, N. Cristianini, P. Bartlett, L. Ghaoui, and M. Jordan. Learning the kernel matrix with semidefinite programming. *Journal of Machine Learning Research*, 5, 2004.

[59] P. Langley and S. Sage. Induction of selective Bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 1994.

[60] P. Larranaga, C. Kuijpers, R. Murga, and Y. Yurramendi. Learning Bayesian network structures by searching for the best ordering with genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 26(4):487–493, 1996.

[61] S. Lauritzen. The EM algorithm for graphical association models with missing data. *Computational Statistics and Data Analysis*, 19(2):191–201, 1995.

[62] S. Lauritzen. *Graphical Models*. Clarendon Press, Oxford, 1996.

[63] F. Li and Y. Yang. Recovering genetic regulatory networks from micro-array data and location analysis data. *Genome Informatics*, 15:131–140, 2004.

[64] D. Margaritis. *Learning Bayesian Network Model Structure from Data*. PhD thesis, CMU, CS, 2003.

[65] B. Merialdo. Tagging English text with a probabilistic model. *Computational Linguistics*, 20(2):155–171, 1994.

[66] A. Moore and W. Wong. Optimal reinsertion: A new search operator for accelerated and more accurate Bayesian network structure learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[67] K. Murphy. A brief introduction to graphical models and Bayesian networks. http://www.ai.mit.edu/~murphyk/Bayes/bnintro.html, 1998.

[68] R. Neal and G. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, 1999.

[69] A. Ng. Feature selection, L1 vs L2 regularization, and rotational invariance. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[70] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

[71] J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[72] B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University, 1996.

[73] J. Rissanen. Stochastic complexity. *Journal of the Royal Statistical Society, Series B*, 49, 1987.

[74] R. Rockafellar. *Convex Analysis*. Princeton Univ. Press, 1970.

[75] T. Roos, H. Wettig, P. Grünwald, P. Myllymäki, and H. Tirri. On discriminative Bayesian network classifiers and logistic regression. *Machine Learning*, 2004.

[76] S. Russell, J. Binder, D. Koller, and K. Kanazawa. Local learning in probabilistic networks with hidden variables. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 1995.

[77] R. Salakhutdinov, S. Roweis, and Z. Ghahramani. Optimization with EM and expectation-conjugate-gradient. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

[78] J. Salojarvi, K. Puolamaki, and S. Kaski. Expectation maximization algorithms for conditional likelihoods. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 2005.

[79] T. Sato and Y. Kameya. A Viterbi-like algorithm and EM learning for statistical abduction. In *Notes for UAI-00 Workshop on Fusion of Domain Knowledge with Data for Decision Support*, 2000.

[80] B. Schoelkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[81] G. Schwartz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[82] E. Segal, D. Pe'er, A. Regev, D. Koller, and N. Friedman. Learning module networks. *Journal of Machine Learning Research*, 6:557–588, 2005.

[83] B. Shen, X. Su, R. Greiner, P. Musilek, and C. Cheng. Discriminative parameter learning of general Bayesian network classifiers. In *Proceedings of the Fifteenth IEEE International Conference on Tools with Artificial Intelligence*, 2003.

[84] I. Simon, J. Barnett, N. Hannett, C. Harbison, N. Rinaldi, T. Volkert, J. Wyrick, J. Zeitlinger, D. Gifford, T. Jaakkola, and R. Young. Serial regulation of transcriptional regulators in the yeast cell cycle. *Cell*, 106:697–708, 2001.

[85] P. Simon, L. Kevin, and T. James. Grafting: Fast, incremental feature selection by gradient descent in function space. *Journal of Machine Learning Research*, 3:1333–1356, 2003.

[86] P. Spirtes, C. Glymour, and R. Scheines. *Causation, Prediction and Search*. MIT Press, 2000.

[87] N. Srebro, G. Shakhnarovich, and S. Roweis. An investigation of computational and informational limits in gaussian mixture clustering. In *Proceedings of the Twenty-Third International Conference on Machine Learning*, 2006.

[88] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *Advances in Neural Information Processing Systems 16*, 2003.

[89] B. Taskar, D. Klein, M. Collins, D. Koller, and C. Manning. Max-margin parsing. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2004.

[90] M. Teyssier and D. Koller. Ordering-based search: A simple and effective algorithm for learning Bayesian networks. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, 2005.

[91] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the Twenty-First International Conference on Machine Learning*, 2004.

[92] T. Van Allen and R. Greiner. Model selection criteria for learning belief nets. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 2000.

[93] T. Van Allen, R. Greiner, and P. Hooper. Bayesian error-bars for belief net inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, 2001.

[94] E. van Someren, L. Wessels, and M. Reinders. Linear modeling of genetic networks from experimental data. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, pages 355–366, 2000.

[95] R. Vanderbei. *Linear Programming: Foundations and Extensions*. Kluwer, 1996.

[96] M. Wainwright and M. Jordan. Graphical models, exponential families, and variational inference. Technical Report TR-649, UC Berkeley, Dept. Statistics, 2003.

[97] S. Wang. Reconstructing genetic networks from time ordered gene expression data using Bayesian method with global search algorithm. *Journal of Bioinformatics and Computational Biology*, 2:441–458, 2004.

[98] H. Wettig, P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri. On supervised learning of Bayesian network parameters. Technical Report HIIT 2002-1, Helsinki Institute for Information Technology, 2002.

[99] H. Wettig, P. Grünwald, T. Roos, P. Myllymäki, and H. Tirri. When discriminative learning of Bayesian network parameters is easy. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, 2003.

[100] J. Yu, V. Smith, P. Wang, A. Hartemink, and E. Jarvis. Advances to Bayesian network inference for generating causal networks from observational biological data. *Bioinformatics*, 20:3594–3603, 2004.

[101] M. Zou and S. Conzen. A new dynamic Bayesian network (DBN) approach for identifying gene regulatory networks from time course microarray data. *Bioinformatics*, 21:71–79, 2005.

Appendix A

Feature Generation Example

Here I use a simple example to illustrate the feature generation procedure showed in Figure 3.1. Consider the feature generation problem for variable X_3 with $\{X_1, X_2\}$ as the potential parent set. Assume that the value domains for these three variables are all $\{1, 2\}$, and the relevant columns (corresponding to the three variables) of training data D for this problem is given by

$$D_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 2 \end{bmatrix}$$

The augmented matrix $\tilde{D}_3$ can then be obtained by duplicating the rows of the first two columns of D_3 and setting the third column values by enumerating the possible values of X_3—each copy of the data taking a different value

$$\tilde{D}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \\ 1 & 2 & 2 \end{bmatrix}$$

Then, the features are generated as follows.

First, $\Phi^{(0)}$ is constructed to include all the singleton features, i.e., $\Phi^{(0)} = \{\phi(x_3 = 1), \phi(x_3 = 2)\}$. The response matrix for $\Phi^{(0)}$ on $\tilde{D}_3$ is given by

$$I(\Phi^{(0)}) = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 1 \end{bmatrix}$$

where $\text{rank}(I(\Phi^{(0)})) = 2$.

Next, consider features with one parent variable involved. First, let $\Phi^{(1)} = \emptyset$. Then consider the one-step extended feature set for each feature in $\Phi^{(0)}$. For $\phi(x_3 = 1)$, its one-step extended feature set is $\Psi = \{\phi(x_1 = 1, x_3 = 1), \phi(x_1 = 2, x_3 = 1), \phi(x_2 = 1, x_3 =$

1), $\phi(x_2 = 2, x_3 = 1)\}$ with response matrix

$$I(\Psi) = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Since rank$(I(\Phi^{(0)} \cup \Phi^{(1)} \cup \Psi)) = 3 >$ rank$(I(\Phi^{(0)} \cup \Phi^{(1)})) = 2$, the extension feature set Ψ would be generated: $\Phi^{(1)} = \Phi^{(1)} \cup \Psi$. Similarly, the one-step extension feature set for $\phi(x_3 = 2)$ is $\Psi = \{\phi(x_1 = 1, x_3 = 2), \phi(x_1 = 2, x_3 = 2), \phi(x_2 = 1, x_3 = 2), \phi(x_2 = 2, x_3 = 2)\}$ with response matrix

$$I(\Psi) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

Since now rank$(I(\Phi^{(0)} \cup \Phi^{(1)} \cup \Psi)) = 4 >$ rank$(I(\Phi^{(0)} \cup \Phi^{(1)})) = 3$, this new set Ψ would be generated as well: $\Phi^{(1)} = \Phi^{(1)} \cup \Psi$.

To this point, the rank of the response matrix has reached the maximum number, 4—the number of rows of $\tilde{D}_3$. Thus the feature generation process can be stopped, without considering further extensions.