University of Alberta

# Discriminative Learning of Bayesian Net Parameters

By

Wei Zhou ©

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of Master of Science.

Department of Computing Science

Edmonton, Alberta

Spring 2002

0-612-69787-8

University of Alberta

Library Release Form

Name of Author: Wei Zhou

Title of Thesis: Discriminative Learning of Bayesian Net Parameters

Degree: Master of Science

Year this Degree Granted: 2002

Wei Zhou

12911-147 Ave.
Edmonton, AB T6V 1E3
Canada

Oct. 31, 2001

University of Alberta

Faculty of Graduate Studies and Research

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research for acceptance, a thesis entitled Discriminative Learning of Bayesian Net Parameters submitted by Wei Zhou in partial fulfillment of the requirements for the degree of Master of Science.

Russell Greiner

Russ Greiner

[illegible]

Peter Hooper

[illegible]

Terry Caelli

Date: Oct 26 2001

Dedicated to

my wife, my parents, my grand parents, and my sister

## ABSTRACT

Like most other generative models, Bayesian networks are commonly learned using generative approaches. In particular, the maximum likelihood approach is often used to produce structures and to estimate the parameters of Bayesian nets. Besides their important application in modeling, Bayesian nets also always had important applications on specialized tasks such as diagnosis or classification. It is therefore reasonable to use discriminative approaches that directly maximize Bayesian nets' performance on these tasks.

In this dissertation, we describe new discriminative approaches, with a focus on maximum conditional likelihood estimation. We compare this approach with the commonly used maximum likelihood approach. We provide empirical evidence to show that the discriminative approaches indeed perform better than the generative approach. In addition, the empirical results show that the discriminative approaches perform well on many real world problems.

## ACKNOWLEGEMENT

The person I wish to thank the most is my supervisor, Russ Greiner, who has been a great source of inspiration, knowledge, encouragement, and support, who has helped me to improve so much in the past few years, and secured my interest in artificial intelligence.

Thanks to Peter Hooper and Terry Caelli for being on my thesis committee.

Thanks to Syncrude and NSERC for their generous funding.

Thanks to Andrew Liu, the most wonderful instructor, for reviving my interest in mathematics.

Thanks to my wife, my parents, my grand parents, and my sister for their endless love.

# Table of Contents

**CHAPTER 1 INTRODUCTION ........ 1**

1.1 OVERVIEW ........ 2

**CHAPTER 2 BACKGROUND ........ 4**

2.1 LEARNING ........ 4

*2.1.1 The Components of Learning ........ 5*

*2.1.2 Generative vs. Discriminative ........ 6*

*2.1.3 Unsupervised vs. Supervised ........ 7*

2.2 BAYESIAN NETWORKS ........ 7

*2.2.1 The Advantages of Bayesian Nets ........ 7*

*2.2.2 Bayesian Nets Defined ........ 8*

2.3 LEARNING BAYESIAN NETWORK ........ 9

**CHAPTER 3 LEARNING BAYESIAN NET PARAMETERS . 10**

3.1 EVALUATION CRITERIA ........ 10

*3.1.1 Maximum Likelihood ........ 10*

*3.1.2 Maximum Conditional Likelihood ........ 11*

*3.1.3 Minimum Classification Error ........ 12*

*3.1.4 Minimum Square Error ........ 13*

3.2 LEARNING ALGORITHM ........ 14

*3.2.1 Complexity ........ 14*

*3.2.2 Gradient Method ........ 17*

*3.2.3 Expectation Maximization ........ 23*

*3.2.4 Inference ........ 24*

**CHAPTER 4 EXAMPLES AND EMPIRICAL RESULTS ........ 25**

4.1 ARTIFICIAL NETWORK ........ 25

*4.1.1 Computational Efficiency ........ 25*

*4.1.2 Accuracy with Incorrect Structure ........ 26*

4.2 REAL WORLD DATA ........ 29

*4.2.1 Naïve Bayes ........ 31*

*4.2.2 Tree Augmented Naïve Bayes ........ 37*

*4.2.3 Incomplete Data ........ 41*

4.3 SUMMARY ........ 48

*4.3.1 Computational Efficiency ........ 49*

**CHAPTER 5 DISCUSSIONS AND FUTURE WORK ........ 51**

5.1 RELATED WORK ........ 51

5.2 IS DISCRIMINATIVE APPROACH ALWAYS BETTER? ........ 52

*5.2.1 Bayesian Net for Avoiding Over-Fitting* ........ *52*
5.3 FEATURE SELECTION ........ 53
5.4 MODEL IMPROVEMENT ........ 54
5.5 STRUCTURE LEARNING ........ 54
5.6 COMPUTATIONAL EFFICIENCY ........ 55
5.7 APPROXIMATE INFERENCE ........ 56
**CHAPTER 6 CONCLUSION ........ 58**
**BIBLIOGRAPHY ........ 59**

# List of Tables

Table 2-1 The components of learning ........ 5
Table 3-1 Queries used to prove MCL estimation of BN parameter is NP-hard ........ 15
Table 4-1 Description of datasets used in the experiments ........ 31
Table 4-2 Classification error of six classifiers over 25 datasets for complete data ........ 34
Table 4-3 Classification error of six classifiers over 25 datasets for incomplete data ........ 43

# List of Figures

Figure 2.1 The learning process ........ 5
Figure 2.2 a simple Bayesian network ........ 8
Figure 3.1 Belief Net structure for any SAT problem (Cooper 1990) ........ 15
Figure 4.1 An example to illustrate the efficiency of MCL estimation ........ 25
Figure 4.2 An example to show the effect of incorrect structure ........ 26
Figure 4.3 Comparing MCL and ML on incorrect structure using LL score 27
Figure 4.4 Comparing MCL and ML on incorrect structure using LCL score ........ 28
Figure 4.5 Comparing MCL and ML on incorrect structure using CE score 28
Figure 4.6 Comparing MCL and ML on incorrect structure using MSE score ........ 29
Figure 4.7 CLGD and LM on CHESS dataset ........ 32
Figure 4.8 Comparing CLGD and LM on UCI datasets using naive Bayes structure ........ 33
Figure 4.9 Comparing NB+CLGD with C4.5 ........ 35
Figure 4.10 Comparing NB+CLGD and SNB+LM ........ 36
Figure 4.11 TAN structure (Friedman, Geiger et al. 1997) ........ 37
Figure 4.12 Comparing NB+CLGD and TAN+LM ........ 38
Figure 4.13 Comparing TAN+CLGD with NB+CLGD ........ 39
Figure 4.14 Comparing TAN+CLGD with TAN+LM ........ 40
Figure 4.15 Comparing TAN+CLGD with C4.5 ........ 40
Figure 4.16 Comparing TAN+CLGD with SNB+LM ........ 41
Figure 4.17 Comparing NB+CLGD with NB+LGD ........ 44
Figure 4.18 NB+CLGD with NB+LEM ........ 44
Figure 4.19 Comparing TAN+CLGD with TAN+LEM, ........ 47
Figure 4.20 Comparing TAN+CLGD with TAN+LGD ........ 47
Figure 4.21 The effect of different missing amount on letter dataset ........ 48
Figure 5.1 MCL structure for complete data ........ 55

# List of Symbols

| Symbol | Meaning |
|---|---|
| $B$ | Bayesian net |
| $\Theta$ | parameters of a Bayesian net |
| $B_\Theta$ | Bayesian net with parameter $\Theta$ |
| $\mathbf{D}$ | data |
| $\mathbf{c}$ | a case contained in data $\mathbf{D}$, either a tuple $\mathbf{t}$ or a query $\langle q \mid \mathbf{e} \rangle$ |
| $\mathbf{t}$ | tuple |
| capital letter such as $X$ | random variable |
| small letter such as $x$ | a particular assignment of random variable $X$ |
| bold letter such as $\mathbf{e}$ | a set of variables or assignments |
| $V_X$ | the set of all possible values of X |
| $\Pi_X$, $\pi_x$ | parent variables and parent configuration of $X$ |
| $\theta_{x\|\pi}$ | a BN parameter corresponding to $p(x\| \pi_x)$ |
| $Q$, $q$ | query variable and assignment |
| $\mathbf{E}$, $\mathbf{e}$ | evidence variables and assignments |
| $p$ | probability |
| $p_B$, $p_\Theta$ | probability given by Bayesian net $B$ or parameters $\Theta$ |

# List of Abbreviations

| Abbreviation | Meaning |
|---|---|
| BN | Bayesian Net, Section 2.2 |
| CPT | Conditional Probability Table, Section 2.2.2 |

**Scores**

| | |
|---|---|
| LL | Log Likelihood, Equation (3.2) |
| LCL | Log Conditional Likelihood, Equation (3.4) |
| CE | Classification Error, Equation (3.6) |
| SE | Square Error, Equation (3.8) |

**Criteria**

| | |
|---|---|
| ML | Maximum Likelihood, Section 3.1.1 |
| MCL | Maximum Conditional Likelihood, Section 3.1.2 |
| MCE | Minimum Classification Error, Section 3.1.3 |
| MSE | Minimum Square Error, Section 3.1.4 |

**Algorithms, Section** 3.2

| | |
|---|---|
| LM | Likelihood Maximization (frequency based estimation) |
| LEM | Likelihood Expectation Maximization (EM) |
| LGD | Likelihood Gradient Descent |
| CLGD | Conditional Likelihood Gradient Descent |
| DEPB | methods for Discriminative Estimation of Parameters for BNs, including MCL, MCE, MSE approaches, Section 3.2.2 |

# Chapter 1

# Introduction

Imagine that a medical doctor is going to build a cancer diagnostic system using the data he collected from the patients over the years. He wishes to use this system to accurately diagnosis the presence of cancer for future patients, after entering their signs and symptoms (e.g. weight, age, gender, smoking history…) into the system. He may choose one of the following two approaches for building this diagnosis system.

1. *Generative*: Build a general model that closely resembles reality, that is, matches all existing data, including both patients' information and the actual presence of cancer.

2. *Discriminative*: Build a specialized system that focuses on correctly predicting the presence of cancer given the patients' information, not caring about the connections among various signs and symptoms provided by the patients.

Although the same data is used by both approaches, the two approaches see the data differently. The first approach treats the presence of cancer as an ordinary variable, the same as all other information provided about patients. The second approach distinguishes the presence/absence of cancer from the other patient information; the cancer property is the objective of the prediction, called the *query variable*; and patients' information is evidence used to make this prediction, called *evidence variables*.

Which approach should the doctor choose? Let us look at some characteristics of the two approaches.

If we are successful in building a perfect model of the real world using the generative approach, the result should work well on cancer diagnosis, as well as many other tasks, such as representing the connection between weight and age. Because the generative approach is "all-encompassing", it naturally requires a great deal of work. However, it has some important advantages. The first advantage of the generative approach is its intuitiveness; we can easily understand the model, as it resembles reality, e.g., each individual component in the model such as how age affects weight, corresponds to the real world phenomenon, and can be verified independently. The second advantage is that the generative model is modular and easily extendable; we can build up on to it later when we need to diagnose say fever, because much of the needed the connections among the variables are already correctly represented.

As noted above, a generative approach requires much work to build a good model of the real world, a model that correctly represents connections among all variables, including the link between age and weight. This is often too expensive for many real world problems. If we insist on building the generative model with limited resources, e.g., with limited computation time, the model quality suffers. With limited resources, the doctor may happily trade off the intuitiveness and modularity of a generative model for the efficiency and accuracy that are provided by the discriminative approach. The discriminative approach does not care about the connections among evidence variables, but instead focuses on the objective of diagnosis, predicting the query variable given evidence. As opposed to the generative approach, the results of the discriminative approach are not intuitive and not easily extendable. Nevertheless, if intuitiveness and extensibleness are not an issue, discriminative approaches are usually more practical than generative approaches.

There is in fact a combined approach – first quickly building a crude generative model using the generative approach, and then refining the model on our specific task – cancer diagnosis – using the discriminative approach. This combined approach usually both offers the advantages provided by either pure generative or discriminative approach, and also eliminates most of their problems.

## 1.1 Overview

The concept of generative and discriminative approaches is quite general and applies to many other problems. In this dissertation, we look at how they apply to the parameter estimation of Bayesian networks (BNs). Like most other generative models, Bayesian nets are typically constructed using a generative approach; here, using the maximum likelihood approach that maximizes the model's chance of producing the observed data (Heckerman 1995).

Although typically constructed using a generative approach, Bayesian nets are often used in specialized tasks such as diagnosis, where the objective is to infer the result from the given evidence. Surprisingly, even trivial Bayesian nets like naïve Bayes (Langley, Iba and Thompson 1992) with strong independence assumptions among the evidence variables, have been found to perform extremely well on many classification tasks, making it one of the most widely used classifiers. Since discriminative approaches improve a BN's performance on these tasks directly, it is therefore reasonable to learn Bayesian nets in a discriminative manner, which may result in even better classification performance.

In our research, we look at ways to learn Bayesian nets discriminatively, seeking BNs that perform well on specified predictions or answering queries. In particular, we developed methods to discriminatively estimating the parameters of Bayesian nets. Our empirical results indicated that discriminative estimation is indeed useful – it consistently improves the performance of Bayesian nets on real world classification tasks.

The following is the outline of the remaining chapters.

In Chapter 2, we give a short review of machine learning and Bayesian networks.

In Chapter 3, we describe various ways of estimating the parameters of Bayesian networks. In particular, we discuss four criteria – likelihood, conditional likelihood, classification error, and square error – for evaluating and learning the parameters, and we discuss methods to optimize these criteria.

In Chapter 4, we provide examples to illustrate the characteristics of generative and discriminative estimation of Bayesian net parameters; we also provide empirical results comparing the classification performance of maximum likelihood estimation and maximum conditional likelihood estimation on real world data.

In Chapter 5, we relate this work to existing results, and then discuss the possible extensions and future work.

In Chapter 6, we summarize and conclude.

## Thesis

The central thesis of this work is that *discriminative estimation of BN parameters is useful. Comparing to the typically used maximum likelihood estimation, its resulting BNs often produce more accurate predications.*

# Chapter 2

# Background

## 2.1 Learning

*Learn – To gain knowledge, comprehension, or mastery of through experience*
*– The American Heritage® Dictionary of the English Language*

Humans are undoubtedly the most potent learners in the world. Not coincidentally, humans are also the most intelligent life forms on earth. Apparently, nature believed in learning and gave us human this powerful ability, which made us the dominant species on earth, and eventually allowed us to ponder today whether we can again use this powerful method to make other intelligent machines like ourselves... Let us not be too ambitious now, as this ultimate goal of artificial intelligence is still far out of our reach. However, the importance of learning is evident for human, as well as for machines – even still at this infant stage, machine learning has already had many great successes and its applications are growing faster and faster. Its development is fuelled by the exponential increase in computational power and pushed by the ever-increasing demands in industry (Mitchell 1997).

Machine learning allows complicated systems to be constructed automatically, and can save the human the work of specifying every detail for the system. In fact, it is often impossible for a human to specify these details for the following reasons:

1. The source data may contain thousands or millions of records and variables; this is too much for an ordinary human to handle. The system may be too complicated, often difficult to understand, let alone construct. On the other hand, a computer can quickly go through a much larger amount of data than humans, and can build complex models accurately.

2. Many details of the environment may not be known at the time of the creation, or may change in a dynamic environment. For example, the consumer's behaviour may change over time. While it is difficult for a human to keep updating the system to accommodate for new data, it is easy for a machine learner to adapt to the changes.

## 2.1.1 The Components of Learning

The learning process involves the basic components shown in Table 2-1. Figure 2.1 illustrates the learning process.

| Gneral learning | Machine learning |
|---|---|
| Learner | Learning algorithm |
| Knowledge | Model |
| Experience | Data |
| Reward/Punishment | Performance evaluation |

Table 2-1 The components of learning

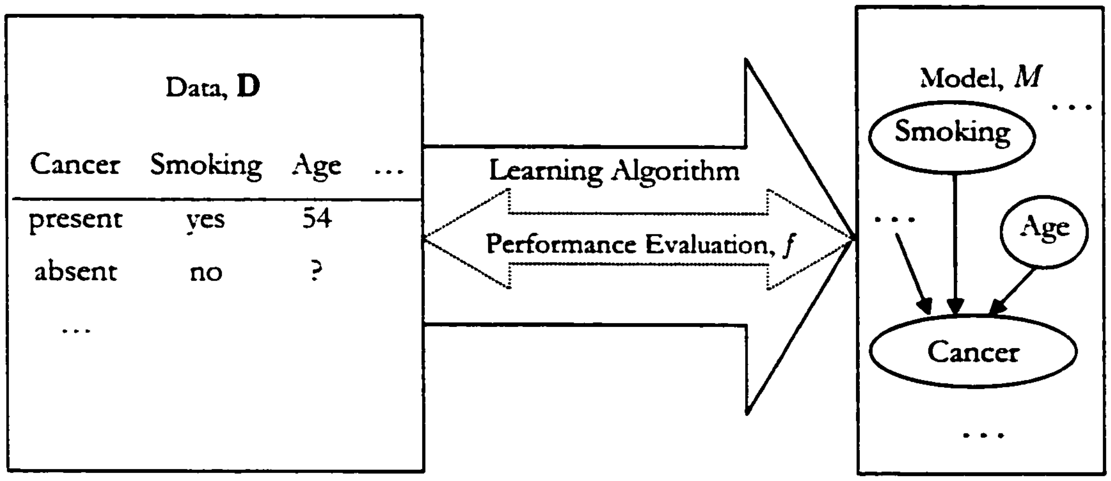


Figure 2.1 The learning process

### Learning Algorithm

A learning algorithm defines the actual learning process. It species how to build an optimal model $M$ from data $\mathbf{D}$ according to some evaluation function $f(M,\mathbf{D})$ i.e., it tries to find a model $M^{*} = \arg\max_{M} f(M,\mathbf{D})$. The effectiveness of a learning algorithm is determined by its efficiency on finding optimal $M^*$ and the $f$-quality of its results.

### Model

Models are often used by learning algorithms to represent knowledge. A good model allows the knowledge to be stored and used efficiently. Bayesian nets, neural nets, rule sets, and decision trees are each models widely used in machine learning (Mitchell 1997). Our research focuses on Bayesian nets (BNs), which will be explained in Section 2.2.

**Data**

The data **D** is the input of the learning process. It is sampled from the original environment – the ***true distribution*** of the data. For example, if 10% of the patients in the population have cancer, we expect 10% of the sample data to have *Cancer=present*.

Each data point is described in terms of a set of variables **X**, e.g., in the example shown in Figure 2.1, **X** contains Cancer, Smoking and Age. The variable can be either ***discrete***, as in Cancer, or ***continuous***, as in Age. We only look at discrete variables in this dissertation. If necessary, continuous variables are made discrete by using discretization procedures. The set of all possible values of variable $X$ is denoted by $V_X$.

The data is divided into ***cases***, also called ***records*** or ***instances***, i.e., $\mathbf{D} = \{\mathbf{c}_1,\ldots,\mathbf{c}_n\}$, e.g., each **c** corresponds to a patient. Each case is an independent sample from the original environment. It contains the values of variables $\mathbf{c} = \{x_1,\ldots x_n\}$, e.g., $x_1$ might be *Cancer=present*, $x_2$ might be *Smoking=yes*. The assigned value $X=x$ is also called an ***observation***. The data collection process is often imperfect. It may introduce noise into the data. It may omit the values of some variables; when this happens, we say the data is ***incomplete***. For example, the data in Figure 2.1 is incomplete because *Age* is unknown in the second case.

**Performance Evaluation**

Before any learning can occur, the objective of the learning must be defined. For example, when learning a diagnostic system, we may wish to have a system that is likely to provide correct diagnostic results for all patients. An evaluation function is used by the learner to determine how close it is to this objective. The evaluation function is important as it guides the learning process. Different evaluation functions usually correspond to different learning objectives, and often produce different results.

### 2.1.2 Generative vs. Discriminative

The previous chapter mentioned two types of learning objectives, generative and discriminative, and noted that generative learning tries to build a model that closely resembles the original distribution, while discriminative learning tries to optimize the model's performance on specific tasks such as diagnosis.

### 2.1.3 Unsupervised vs. Supervised

We noted that the same data could be used for both generative and discriminative approaches. However, the data is treated differently. The generative approach usually treats all variables equally and maximizes the joint likelihood of all observations – this corresponds to *unsupervised* learning, and we call each case a *tuple*, denoted by $\mathbf{t}$. On the other hand, the discriminative approach distinguishes the query variables $\mathbf{Q}$ from the evidence variables $\mathbf{E}$ and maximizes the performance on answering $\mathbf{Q}$ given $\mathbf{E}$ – this corresponds to *supervised* learning, and we call each case a *query*, denoted by $\langle \mathbf{q} | \mathbf{e} \rangle$. For simplicity, we assume only one query variable $Q$ in $\mathbf{X}$, and $Q$ is the same variable for all cases, although our work is directly applicable to multiple query variables, where a case may contain more than one query variables $\mathbf{Q}$.

## 2.2 Bayesian Networks

Bayesian Networks (Pearl 1988) have became popular tools for many real world applications, including modeling, diagnosis and classification (Spiegelhalter, Dawid, Lauritzen and Cowell 1993; Friedman et al. 1997). BNs have several strengths that make them particularly attractive.

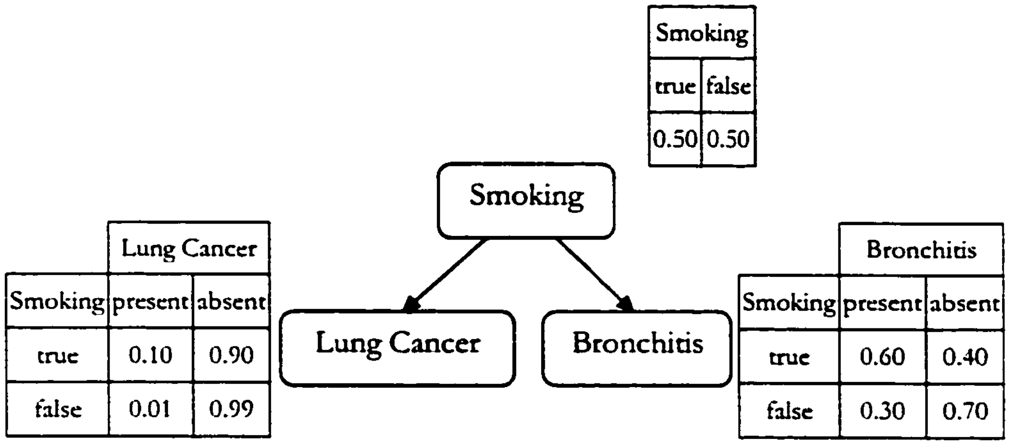
### 2.2.1 The Advantages of Bayesian Nets

As a statistical model, its properties and functionalities are intuitive and have strong theoretical justifications. We can use many existing results developed for general statistical models. Bayesian nets naturally handle uncertainties. This is an important feature, as uncertainty always exists in real world. A BN model can thus provide realistic results (Pearl 1988). BNs also handle missing data. A BN always produce reasonable inference results given any amount of evidence; the results become more accurate as the number of evidence increases.

As a graphical model, its semantics is easily understood by human. For this reason, it is easy to construct and modify by humans. The "localness" or modularity of the structure also allows efficient inference.

### 2.2.2 Bayesian Nets Defined

A Bayesian network $B=\langle G, \Theta\rangle$ consists of a graphical structure $G$ and parameters $\Theta$. A simple BN is depicted in Figure 2.2.



| Smoking | |
|---|---|
| true | false |
| 0.50 | 0.50 |

| | Lung Cancer | |
|---|---|---|
| Smoking | present | absent |
| true | 0.10 | 0.90 |
| false | 0.01 | 0.99 |

| | Bronchitis | |
|---|---|---|
| Smoking | present | absent |
| true | 0.60 | 0.40 |
| false | 0.30 | 0.70 |

Figure 2.2 a simple Bayesian network

#### Structure

The structure $G=\langle\mathbf{V},\mathbf{A}\rangle$ is a directed acyclic graph, with each node $V\in\mathbf{V}$ representing a random variable $X$, and each arc $A\in\mathbf{A}$, $A\equiv\langle Y,X\rangle$ representing the claim that variable $Y$ *causes* variable $X$, and $Y$ is a parent of $X$. The set of all parents of X is denoted by $\Pi_X$. The variables $X$ and $\Pi_X$, form the *family* of $X$. For example, to represent Smoking causes Lung Cancer, we add an arc from Smoking to Lung Cancer as shown in Figure 2.2. We see this graphical structure is intuitive; which allows it to be easily constructed by human experts. This makes BN a reasonable model for encoding expert knowledge (Pearl 1988; Jensen 1996).

#### Parameters

While the graphical structure of a BN merely indicates whether a variable directly causes another, the parameters $\Theta$ provides more information on these causal connections, by specifying the *conditional distribution* $p(X|\Pi_X)$ for each family. With discrete variables, the conditional distribution can be specified in tables, as shown in Figure 2.2. For example, we see the probability of *Lung Cancer=present* given *Smoking=true*, *p(Lung Cancer=present| Smoking=true)* is 0.10. These tables are called ***conditional probability tables*** (CPTs). The size of a BN is measured by the total number of parameters, also known as CPT entries.

A BN efficiently represent the full joint distribution of all its variables. The joint probability of any complete tuple **t** can be calculated as

$$p(\mathbf{t}) = \prod_{x \in \mathbf{t}} p(x \mid \pi_x) = \prod_{x \in \mathbf{t}} \theta_{x|\pi_x} \tag{2.1}$$

where $x$ is a variable assignment in **t**; $\pi_x$ is the parents' configuration of $x$ from **t**, $\theta_{x|\pi_x}$ is the parameter that corresponds to variable assignment $x$ and its parents' configuration $\pi_x$. For example,

*p(Smoking=true, Lung Cancer=present, Bronchitis=false)*
*= p(Smoking=true) * p(Lung Cancer=present| Smoking=true)*
** p(Bronchitis=false| Smoking=true)*
*=0.50*0.10*0.40*
*=0.02*

## 2.3 Learning Bayesian Network

As mentioned above, a BN consists of two parts – the graphical structure and parameters. In this dissertation, we focus on learning parameters and assume the structure is given. A justification for this is that human experts are better at constructing the graphical structure, and worse at specifying the numerical parameters. In addition, learning the structure is often expensive for machine learning (Chickering, Geiger and Heckerman 1994). With extensive prior knowledge, human experts can build the structure more easily. A typical way of constructing a BN model is to ask the human expert to construct a structure, and then use the machine learner to learn the parameters.

In the next chapter, we will discuss various ways to learn BN parameters, including both generative approaches and discriminative approaches.

# Chapter 3

# Learning Bayesian Net Parameters

Given a Bayesian network structure and data **D** that consists of independent cases, each case **c** may be considered as either a tuple **t**, or a query $\langle q|\mathbf{e}\rangle$, our learning task is to find the parameters $\Theta^* = \arg\max_\Theta f(\Theta, \mathbf{D})$, where $f(\Theta,\mathbf{D})$ evaluates parameters $\Theta$ on **D** according to some criterion.

## 3.1 Evaluation Criteria

As discussed previously, the evaluation criterion corresponds to the learning objective. We first review the commonly used maximum likelihood criterion that corresponds to building a good generative model; we will then look at three criteria for discriminatively learning BN parameters – conditional likelihood, classification error, and square error.

### 3.1.1 Maximum Likelihood

Maximum likelihood (ML) estimation finds parameters that maximize the likelihood of all observed tuples

$$L(\Theta,\mathbf{D}) = p_\Theta(\mathbf{D}) = \prod_{\mathbf{t}\in\mathbf{D}} p_\Theta(\mathbf{t}) \qquad (3.1)$$

or equivalently, maximize the log likelihood (LL) of the tuples,

$$LL(\Theta,\mathbf{D}) = \log p_\Theta(\mathbf{D}) = \sum_{\mathbf{t}\in\mathbf{D}} \log p_\Theta(\mathbf{t}) \qquad (3.2)$$

where the function $p_\Theta$ corresponds to the probability computed from BN inference using parameters $\Theta$. Note that $p_\Theta(\mathbf{t})$ is the joint probability of all observations in the tuple **t**. Thus, this criterion can be more accurately referred to as *maximum joint likelihood*, which can help in distinguishing it from *maximum conditional likelihood*, which will be discussed next. However, maximum joint likelihood is typically just referred to as maximum likelihood.

In practice, the log likelihood is often used in to simplify computation, as the summation in (3.2) is often easier to manipulate than the product in (3.1).

Note we sometimes use average negative log likelihood $-LL/|\mathbf{D}|$ for convenience, as the negation makes this score always positive, and the averaging scales it to a common unit measure that allows easy comparison.

### 3.1.2 Maximum Conditional Likelihood

While likelihood evaluates the quality of a generative model and it is widely used for many estimation tasks, a good likelihood does not always lead to good performance on answering queries. As noted by (Friedman et al. 1997), $\log p(\mathbf{t}) = \log p(q,\mathbf{e}) = \log p(q|\mathbf{e}) + \log p(\mathbf{e})$. Only the first conditional term $p(q|\mathbf{e})$ corresponds to the performance of predicting $q$ given $\mathbf{e}$. However, the joint probability of the evidence $p(\mathbf{e})$ may dominate $p(q|\mathbf{e})$, especially when the evidence contains many variables and are not closely related to the query variable. Thus, the conditional term $p(q|\mathbf{e})$ may easily become an insignificant contribution to the joint probability. Hence, the learner that optimize $p(q,\mathbf{e})$ may not produce a good $p(q|\mathbf{e})$ score, and so may result in poor performance of predicting $q$ given $\mathbf{e}$.

There is another closely related problem. It is well known that the maximum likelihood estimation requires correct assumptions on the underlying distribution – in our case, a correct BN structure. However, we rarely have a perfect BN structure because it is difficult to obtain using either machine learning or human experts.

With machine learning, a learner has to consider an exponential number of possible structures. In addition, most of the heuristic scores used to reduce the search space are problematic and can discard correct structures. For example, MDL score favours simple structures, but correct structures are not necessarily simple (Cohen and Stewart 1994; Friedman et al. 1997; Van Allen and Greiner 2000).

With human expert construction of BN structures, the prior knowledge of the expert is often much more extensive than the heuristic score, which can help to greatly simplify the construction of the structure. However, it is well known that humans are more prone to making mistakes of adding redundant arcs or missing required arcs (Jensen 1996). In addition, good experts do not always exist.

These issues motivate us to investigate alternative criteria for learning BNs. To maximize the probability of correctly predicting all query values given evidence, we maximize the conditional likelihood (CL)

$$CL(\Theta,\mathbf{D}) = \prod_{(q|\mathbf{e})\in\mathbf{D}} p_\Theta(q \mid \mathbf{e}) \tag{3.3}$$

or equivalently maximize the log conditional likelihood (LCL)

$$LCL(\Theta,\mathbf{D}) = \sum_{(q|\mathbf{e})\in\mathbf{D}} \log p_\Theta(q \mid \mathbf{e}) \tag{3.4}$$

The value $\Theta$ that maximizes this score is the maximum conditional likelihood (MCL) estimate.

MCL estimation does not care about $p(\mathbf{e})$, i.e., correctly representing the joint distribution of evidence. By focusing on $p(q|\mathbf{e})$, we will see that it can work well even with incorrect structures, i.e., it is more robust than ML estimation.

In summary, MCL approach can produce better prediction accuracy for queries by directly maximizing the conditional likelihood, and it does not require good structures that may be too expensive to obtain. We next look at two other discriminative.

### 3.1.3 Minimum Classification Error

Classification is the task of predicting the class, that is, the value $q$ of the query variable Q, given evidence **E=e**, also called attributes or features. To use a BN as a classifier, we first use the BN to infer the conditional probability $p_\Theta(q|\mathbf{e})$, we then predict the class value using the optimal classification strategy

$$class_\Theta(\mathbf{e}) = \arg\max_{q'\in V_Q} p_\Theta(q' \mid \mathbf{e}) \tag{3.5}$$

Classification error (CE) is simply the number of incorrect predictions

$$CE(\Theta,\mathbf{D}) = \sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} \left\| class_{\Theta}(\mathbf{e}) \neq q \right\| \tag{3.6}$$

where $\|true\| = 1$, and $\|false\| = 0$.

However, CE based on the optimal classification strategy (3.5) is not a smooth function and is not differentiable; thus, it is difficult to minimize directly. We therefore approximate CE based on an alternative classification function *class'(***e***)* that randomly assigns class according to $p(q|\mathbf{e})$ (Hooper 2001). The expected error AE based on this classification function is

$$\begin{aligned} AE(\Theta,\mathbf{D}) &= \sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} E_{\Theta}\left( \left\| class'_{\Theta}(\mathbf{e}) \neq q \right\| \right) \\ &= \sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} \sum_{q' \in V_Q} p_{\Theta}(q' \mid \mathbf{e}) \left\| q' \neq q \right\| \\ &= \sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} 1 - p_{\Theta}(q \mid \mathbf{e}) \end{aligned} \tag{3.7}$$

There is an alternative interpretation for AE. Recall that MCL approach maximizes the conditional probability $p(q|\mathbf{e})$. We can view (3.7) as simply trying to make $p(q|\mathbf{e})$ as close to 1 as possible, i.e. minimize the difference between the predicted conditional probability and the outcome. Therefore, we also call (3.7) the absolute probability error.

### 3.1.4 Minimum Square Error

Square error (SE) is similar to AE, except it evaluates the square of the difference between the predicted conditional probability and the outcome. SE is commonly in function approximation to measure the discrepancy between approximation and the true value.

$$SE(\Theta,\mathbf{D}) = \sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} \left(1 - p_{\Theta}(q \mid \mathbf{e})\right)^2 \tag{3.8}$$

The four discriminative evaluation functions – LCL, CE, AE, and SE – share some similarities. A BN with perfect score (that is, $LCL=0$, $CE=0$, $AE = 0$, or $SE=0$) on any of these evaluation functions produces perfect scores on all these discriminative evaluation functions. This is easily seen by noting that the perfect scores are only achieved if all conditional probabilities $p(q|\mathbf{e})=1$.

## 3.2 Learning Algorithm

### 3.2.1 Complexity

Unfortunately, it is intractable to find the parameters that optimize the above discriminative evaluation functions.

We first show that it is NP-hard to find maximum conditional likelihood estimation of the BN parameters.

**Theorem 3.1**

MCL estimation of BN parameters is NP-hard.

**Proof (this proof is developed by Russ Greiner)**

We reduce 3SAT to our task, using a construction similar to the one in (Cooper 1990). Given any 3-CNF formula $\varphi=\wedge_i C_i$, where $C_i=\vee_j \pm_{ij} X_{ij}$, we construct the network shown in Figure 3.1, with one node for each variable $X_i$ and one for each clause $C_j$, with an arc from $X_i$ to $C_j$ whenever $C_j$ involves $X_i$ — e.g. If $C_1 = X_1 \vee \neg X_2 \vee X_3$, and $C_2 = \neg X_1 \vee X_3 \vee \neg X_4$, then there are links to $C_1$ from $X_1$, $X_2$, $X_3$, and to $C_2$ from $X_1$, $X_3$, $X_4$. In addition, we include $K-1$ other Boolean nodes, $D_2,\ldots,D_{K-1}$, $A$, where $D_j$ is the child of $D_{j-1}$ and $C_j$, where $D_1$ is identified with $C_1$, and $A$ is used for $D_K$.

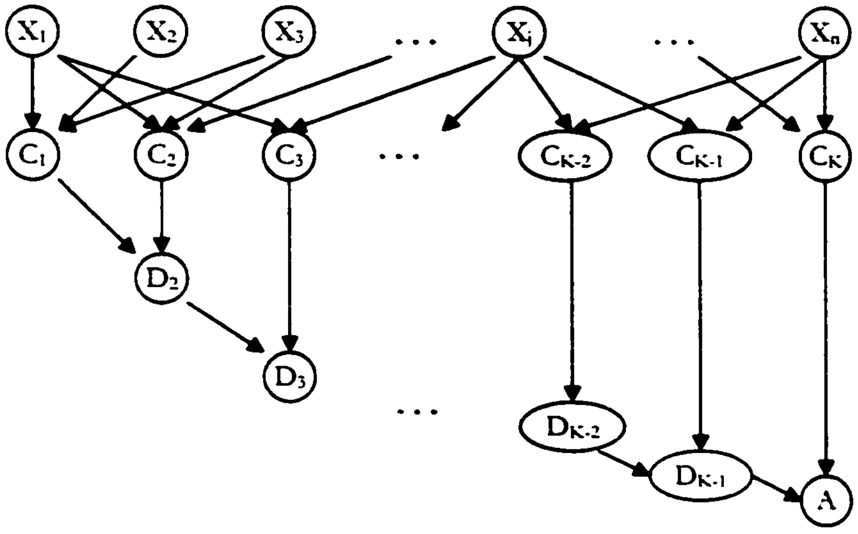


Figure 3.1 Belief Net structure for any SAT problem (Cooper 1990)

Here, we intend each $C_i$ to be true if the assignment to the associated variables $X_{i1}$, $X_{i2}$, $X_{i3}$ satisfies $C_i$; and A corresponds to the conjunction of those variables. We do this using (all-but-the-final) instances in Table 3-1. There is one such instance for each clause, with exactly the assignment (of the 3 relevant variables) that falsifies this clause. Hence, the first line corresponds to $C_1 = X_1 \vee \neg X_2 \vee X_3$.

| $X_1$ | $X_2$ | $X_3$ | $X_4$ | ... | $X_n$ | A |
|---|---|---|---|---|---|---|
| 0 | 1 | 0 | | | | 0 |
| 0 | | 0 | 1 | | | 0 |
| ⋮ | | | | | | ⋮ |
| 0 | | 1 | | 1 | | 0 |
| | | | | | | 1 |

Table 3-1 Queries used to prove MCL estimation of BN parameter is NP-hard

We now prove, in particular, that there is a set of parameters for the structure in Figure 3.1, producing 0 LCL score over the queries in Table 3-1, if and only if there is a satisfying assignment for the associated φ formula.

*There exists a satisfying assignment for the formula $\varphi \Rightarrow$ there exists a set of parameters for the structure in Figure 3.1, producing 0 LCL score over the queries in Table 3-1:*

We just set each $C_i$ to be the disjunction of the associated $X_{i1}$, $X_{i2}$, $X_{i3}$ variables (its parents), with the appropriate sense. E.g., using $C_1 = X_1 \vee \neg X_2 \vee X_3$, then $C_1$'s CPT would have

| $x_1$ | $x_2$ | $x_3$ | $p(C_1=1 \mid X_1=x_1, X_2=x_2, X_3=x_3)$ |
|---|---|---|---|
| 0 | 0 | 0 | 1.0 |
| 0 | 0 | 1 | 1.0 |
| 0 | 1 | 0 | 0.0 |
| 0 | 1 | 1 | 1.0 |
| 1 | 0 | 0 | 1.0 |
| 1 | 0 | 1 | 1.0 |
| 1 | 1 | 0 | 1.0 |
| 1 | 1 | 1 | 1.0 |

Similarly set the CPT for the $D_j$ to correspond to the conjunction of its 2 parents $D_j = D_{j-1} \wedge C_j$; e.g..,

| $d_4$ | $c_5$ | $p(D_5=1 \mid D_4=d_4, C_5=c_5)$ |
|---|---|---|
| 0 | 0 | 0.0 |
| 0 | 1 | 0.0 |
| 1 | 0 | 0.0 |
| 1 | 1 | 1.0 |

Finally, set $X_i$ to correspond to the satisfying assignment; i.e. if $X_1 = 1$, then $p(x_1)=1.0$; and if $X_4 = 0$, then $p(x_4)=0.0$. Note that these CPT values satisfy all $k+1$ of the labelled instances.

*No satisfying assignment for the formula $\varphi \Rightarrow$ there does not exist a set of parameters for the structure in Figure 3.1 that produce 0 LCL score over the queries in Table 3-1:*

Here, we assume there is no satisfying assignment. Towards a contradiction, we can assume that there is a 0 LCL set of CPT entries. This means, in particular, that $p(a|x_{i1}, x_{i2}, x_{i3}) = 0$, where $x_{i1}$, $x_{i2}$, $x_{i3}$ correspond to the assignment that violates the ith constraint. (E.g., for $C_1 = X_1 \vee \neg X_2 \vee X_3$, this would be $X_1=0$, $X_2=1$, $X_3=0$.)

Now consider the final labelled instance, $p(a)$. As there is no satisfying assignment, we know that each assignment $\mathbf{x}$ violates at least one constraint. For notation, let $\gamma^x$ refer to one of these violations (say the one with the smallest index). So if $x = \langle 0,1,0, \ldots \rangle$, then $\gamma^{\langle 0,1,0,\ldots\rangle} = \langle X_1=0,\ X_2=1,\ X_3=0 \rangle$ corresponds to the violation of the first constraint $C_1$. We also let $\beta^x$ refer to the rest of the assignment.

Now observe

$$\begin{aligned} p(a) &= \sum_{\mathbf{x}} p(a,\mathbf{x}) \\ &= \sum_{\mathbf{x}} p(a \mid \gamma^{\mathbf{x}}) p(\gamma^{\mathbf{x}}) p(\beta^{\mathbf{x}} \mid a, \gamma^{\mathbf{x}}) \\ &= \sum_{\mathbf{x}} 0 \cdot p(\gamma^{\mathbf{x}}) p(\beta^{\mathbf{x}} \mid a, \gamma^{\mathbf{x}}) \\ &= 0 \end{aligned} \tag{3.9}$$

This shows that the final instance will be mislabelled. This proves that there can be no set of CPT values that produce 0 LCL-score when there are no satisfying assignments. ■

**Corollary 3.2**

MCE estimation and MSE estimation of BN parameters are NP-hard

**Proof**

Since a set of BN parameters produces a 0 LCL score if and only if it produces a 0 CE score, and the proof of Theorem 3.1 shows that the 3SAT problem can be reduced to the problem of finding parameters that producing 0 LCL score, therefore, the 3SAT problem can also be reduced to the problem of finding parameters that provides 0 CE score, which shows that MCE estimation of BN parameters is NP-hard. The argument for the NP-hardness of MSE estimation is exactly the same. ■

### 3.2.2 Gradient Method

Because it is intractable to find the exact optimum for the evaluation functions discussed above, we use ***conjugate gradient method*** to iteratively search for an optimum. Conjugate gradient method is one of the most effective general-purpose methods for optimizing multidimensional differentiable functions. It saves much work by always searching in *conjugate* directions. See (NumericalRecipesSoftware; Polak 1971) for more information about conjugate gradient method we used. In particular, we used the standard Polak-Ribiere formula for conjugate gradient calculation, and Brent's method (Brent 1973) for line optimization.

Conjugate gradient method, as well as many other optimization methods, requires the gradient calculation. Intuitively, gradient information is important because it points to the direction of an optimum. E.g., a maximum is in the positive gradient direction and a minimum is in the negative gradient direction. The gradient approaches zero as we get close to an optimum.

We will use LGD, CLGD, CEGD, and SEGD to refer to the gradient methods for optimizing LL, LCL, CE and SE scores respectively. These methods correspond to ML, MCL, MCE and MSE approaches. ML approach is generative, whereas MCL, MCE and MSE approaches are discriminative. We refer to these three as DEPB, for discriminative estimation of parameters of Bayesian nets.

We now look at the gradient calculations of the various evaluation functions.

**Log Likelihood**

(Binder, Koller, Russell and Kanazawa 1997) used gradient method to maximize likelihood. We first look at their gradient calculation. Differentiate the log likelihood function $LL(\Theta,\mathbf{D})$ (3.2) with respect to a particular parameter $\theta_{x|\pi}$ of variable assignment $x$ and parents configuration $\pi_x$, where $\pi_x$ is specified in each training tuple $\mathbf{t}$,

$$\frac{dLL(\Theta,\mathbf{D})}{d\theta_{x|\pi_x}} = \frac{d\sum_{\mathbf{t}\in\mathbf{D}} \log p_\Theta(\mathbf{t})}{d\theta_{x|\pi_x}} = \sum_{\mathbf{t}\in\mathbf{D}} \frac{d\log p_\Theta(\mathbf{t})}{d\theta_{x|\pi_x}}$$

Now differentiate each term in the summation

$$\frac{d\log p_\Theta(\mathbf{t})}{d\theta_{x|\pi_x}} = \frac{1}{p_\Theta(\mathbf{t})}\frac{dp_\Theta(\mathbf{t})}{d\theta_{x|\pi_x}}$$

Expand $p_\Theta(\mathbf{t})$ by summing over the values of $X$, so we can separate $\theta_{x|\pi}$

$$\frac{d\log p_\Theta(\mathbf{t})}{d\theta_{x|\pi_x}} = \frac{1}{p_\Theta(\mathbf{t})} \frac{d\sum_{x'\in V_x} p_\Theta(\mathbf{t} \mid x',\pi_x)p_\Theta(x' \mid \pi_x)p_\Theta(\pi_x)}{d\theta_{x|\pi_x}}$$
$$= \frac{1}{p_\Theta(\mathbf{t})} \sum_{x'\in V_x} p_\Theta(\mathbf{t} \mid x',\pi_x) \frac{d\theta_{x'|\pi_x}}{d\theta_{x|\pi_x}} p_\Theta(\pi_x) \tag{3.10}$$

Assuming $d\theta_{x'|\pi}/d\theta_{x|\pi}=0$, for all $x' \neq x$

$$\frac{d\log p_\Theta(\mathbf{t})}{d\theta_{x|\pi_x}} = \frac{p_\Theta(\mathbf{t} \mid x,\pi_x)p_\Theta(\pi_x)}{p_\Theta(\mathbf{t})}$$
$$= \frac{p_\Theta(\mathbf{t},x,\pi_x)}{p_\Theta(\mathbf{t})} \frac{p_\Theta(\pi_x)}{p_\Theta(x,\pi_x)} \tag{3.11}$$
$$= \frac{p_\Theta(x,\pi_x \mid \mathbf{t})}{p_\Theta(x \mid \pi_x)} = \frac{p_\Theta(x,\pi_x \mid \mathbf{t})}{\theta_{x|\pi_x}}$$

We end up with a simple expression. Note this expression requires one BN inference to find $p_\Theta(x,\pi_x|\mathbf{t})$. We will discuss more about inference in Section 3.2.4.

But this result does not make sense – the gradient is always zero or positive! In addition, we cannot have $p_\Theta(x,\pi_x|\mathbf{t})=0$ for all $x,\pi_x$, as $\sum_{x,\pi_x} p_\Theta(x,\pi_x \mid \mathbf{t}) = 1$, so some gradient must be positive. It suggests that we should never decrease any of our parameters, but have to increase some, e.g., never decrease *p(smoking)* and *p(non-smoking)*, but increase at least one of them. This obviously violates the rule of probability. Where did we do wrong?

Our assumption $d\theta_{x'|\pi}/d\theta_{x|\pi}=0$, for $x' \neq x$ is incorrect! There is clearly a strong dependency between $\theta_{x|\pi}$ and $\theta_{x'|\pi}$, as we cannot have only $\theta_{x|\pi}$ changing and leaving $\theta_{x'|\pi}$ the same value. If we do, we will obtain probabilities that do not add up to 1. This incorrect assumption leads to incorrect gradient calculations.

Despite the problem, this result is still used by other researchers (Binder et al. 1997) for ML computation. As an ad hoc way to sidestep this problem, they renormalize the probabilities after each step, and manually fix any parameter not in [0,1] region.

With the incorrect gradient calculated, we cannot expect good performance from the gradient method. It is therefore essential to correct this problem.

We use another set of parameters $\beta$ that corresponds to the original parameters $\theta$ through the relationship $\theta_{x|\pi_x} = \dfrac{e^{\beta_{x|\pi_x}}}{\sum_{x' \in V_X} e^{\beta_{x'|\pi_x}}}$.

While keeping $\theta$ parameters dependent, $\beta$ parameters are now independent of each other, allowing easier differentiation. Also note that the set of $\theta$ will always remain in (0,1), while the set of $\beta$ ranges in ($-\infty$, $\infty$).

We can easily derive $\dfrac{d\theta_{x|\pi_x}}{d\beta_{x|\pi_x}} = \theta_{x|\pi_x}(1-\theta_{x|\pi_x})$ and for $x' \neq x$, $\dfrac{d\theta_{x'|\pi_x}}{d\beta_{x|\pi_x}} = -\theta_{x|\pi_x}\theta_{x'|\pi_x}$. Substituting these to (3.10) produces

$$\begin{aligned}
\frac{d \log p_\Theta(\mathbf{t})}{d\beta_{x|\pi_x}} &= \frac{1}{p_\Theta(\mathbf{t})} \frac{d \sum_{x' \in V_X} p_\Theta(\mathbf{t} \mid x', \pi_x) p_\Theta(x' \mid \pi_x) p_\Theta(\pi_x)}{d\beta_{x|\pi_x}} \\
&= \frac{1}{p_\Theta(\mathbf{t})} \sum_{x' \in V_X} p_\Theta(\mathbf{t} \mid x', \pi_x) \frac{d\theta_{x'|\pi_x}}{d\beta_{x|\pi_x}} p_\Theta(\pi_x) \\
&= \frac{1}{p_\Theta(\mathbf{t})} \begin{pmatrix} p_\Theta(\mathbf{t} \mid x, \pi_x) p_\Theta(x \mid \pi_x) p_\Theta(\pi_x) - \\ \theta_{x|\pi_x} \sum_{x' \in V_X} p_\Theta(\mathbf{t} \mid x', \pi_x) p_\Theta(x' \mid \pi_x) p_\Theta(\pi_x) \end{pmatrix} \\
&= \frac{p_\Theta(x, \pi_x, \mathbf{t}) - \theta_{x|\pi_x} p_\Theta(\pi_x, \mathbf{t})}{p_\Theta(\mathbf{t})}
\end{aligned}$$

$$\frac{d\log p_\Theta(\mathbf{t})}{d\beta_{x|\pi_x}} = p_\Theta(x,\pi_x\mid\mathbf{t}) - \theta_{x|\pi_x} p_\Theta(\pi_x\mid\mathbf{t}) \tag{3.12}$$

Although this expression contains two probability functions – $p_\Theta(x,\pi_x|\mathbf{t})$ and $p_\Theta(\pi_x|\mathbf{t})$, it requires only a single BN inference to find $p(X,\Pi_X|\mathbf{t})$, the result can then be marginalized to get $p(\Pi_X|\mathbf{t})$. Note the computation of marginalization is trivial comparing to that of inference. Thus, the computation time of using $\beta$ parameters is similar to using $\theta$ parameters.

The derivative of complete LL function is

$$\frac{dLL(\Theta,\mathbf{D})}{d\beta_{x|\pi_x}} = \sum_{\mathbf{t}\in\mathbf{D}} \frac{d\log p_\Theta(\mathbf{t})}{d\beta_{x|\pi_x}} = \sum_{\mathbf{t}\in\mathbf{D}} \left( p_\Theta(x,\pi_x\mid\mathbf{t}) - \theta_{x|\pi_x} p_\Theta(\pi_x\mid\mathbf{t}) \right) \tag{3.13}$$

In the special case where data $\mathbf{D}$ is complete, $p(x,\pi_x|\mathbf{t})$ is deterministic and requires no BN inference. To evaluate $p(x,\pi_x|\mathbf{t})$, we only need to check to see if the values $x,\pi$ are in tuple $\mathbf{t}$. If they are, $p(x,\pi_x|\mathbf{t})=1$, otherwise $p(x,\pi_x|\mathbf{t})=0$. Now we can find the exact maximum easily. Setting the derivative to 0,

$$\frac{dLL(\Theta,\mathbf{D})}{d\beta_{x|\pi_x}} = \sum_{\mathbf{t}\in\mathbf{D}} \left( p_\Theta(x,\pi_x\mid\mathbf{t}) - \theta_{x|\pi_x} p_\Theta(\pi_x\mid\mathbf{t}) \right) = 0$$

which implies

$$\theta_{x|\pi_x} = \frac{\sum_{\mathbf{t}\in\mathbf{D}} p_\Theta(x,\pi_x\mid\mathbf{t})}{\sum_{\mathbf{t}\in\mathbf{D}} p_\Theta(\pi_x\mid\mathbf{t})} = \frac{N_{x,\pi_x}}{N_{\pi_x}} \tag{3.14}$$

where $N_{x,\pi}$ is the number of times $x$ and $\pi$ appeared together in all tuples, and $N_\pi$ is the number of times $\pi$ appeared in all tuples. This frequency-based estimation (3.14) is a well-known result (Cooper and Herskovits 1992) and is widely used for ML estimation with complete data.

This shows that the maximum likelihood estimation for complete data is trivial to compute, much easier than the ML estimation of incomplete data. We call this efficient frequency based estimation LM, for likelihood maximization. In our actual implementation, to avoid the problem of $N_\pi=0$, we used a uniform prior probability $1/|V_X|$ for each parameter of variable $X$, where $|V_X|$ is the domain size of variable $X$, so the resulting parameter $\theta_{x|\pi_x} = \frac{N_{x,\pi_x}+1}{N_{\pi_x}+|V_X|}$. The prior information becomes insignificant as the amount of data increases.

Our experimental results have shown that the correct gradient calculated with the $\beta$ parameters (3.12) indeed provides much better performance than the incorrect gradient with $\theta$ parameters.

**Log Conditional Likelihood**

We know $\log p(q|\mathbf{e}) = \log p(q,\mathbf{e}) - \log p(\mathbf{e})$. Using result from (3.12),

$$\frac{d\log p_\theta(q\,|\,\mathbf{e})}{d\beta_{x|\pi_x}} = \left(p_\theta(x,\pi_x\,|\,q,\mathbf{e}) - \theta_{x|\pi_x} p_\theta(\pi_x\,|\,q,\mathbf{e})\right) - \left(p_\theta(x,\pi_x\,|\,\mathbf{e}) - \theta_{x|\pi_x} p_\theta(\pi_x\,|\,\mathbf{e})\right) \tag{3.15}$$

It takes two inferences to evaluate this expression, one for $p(X,\Pi_X|q,\mathbf{e})$, and one for $p(X,\Pi_X|\mathbf{e})$, which is twice of the number required by LL gradient calculation. Note however, this derivative is 0 when the family of variable $X$ is independent of the query variable $Q$ given the evidence $\mathbf{e}$. This is expected, as there is no need to change a parameter if it has no effect on the query. This allows us to ignore the irrelevant part of the network during the gradient calculation. It gives much saving for sparse, large networks.

**Classification Error**

AE is now easy to differentiate. We know

$$\frac{d\log p_\theta(q\,|\,\mathbf{e})}{d\beta_{x|\pi_x}} = \frac{1}{p_\theta(q\,|\,\mathbf{e})}\frac{dp_\theta(q\,|\,\mathbf{e})}{d\beta_{x|\pi_x}}$$

Using the result from (3.15),

$$\frac{d(1-p_\theta(q\mid\mathbf{e}))}{d\beta_{x|\pi_x}} = -p_\theta(q\mid\mathbf{e})\frac{d\log p_\theta(q\mid\mathbf{e})}{d\beta_{x|\pi_x}}$$
$$= -p_\theta(q\mid\mathbf{e})\begin{pmatrix}\left(p_\theta(x,\pi_x\mid q,\mathbf{e})-\theta_{x|\pi_x}p_\theta(\pi_x\mid q,\mathbf{e})\right)- \\ \left(p_\theta(x,\pi_x\mid\mathbf{e})-\theta_{x|\pi_x}p_\theta(\pi_x\mid\mathbf{e})\right)\end{pmatrix}$$

Comparing this to the gradient of LCL, the only difference is the extra $-p_\Theta(q|\mathbf{e})$, which can be easily evaluated with a single inference. Therefore, the method we used to optimize LCL can be used to optimize CA' with this small modification.

**Square Error**

Since we already know the derivative of $p_\Theta(q|e)$, it is now easy to find the derivative of $(1-p_\Theta(q|\mathbf{e}))^2$

$$\frac{d\left(1-p_B(q\mid\mathbf{e})\right)^2}{dB_{x|\pi_x}} = 2\left(p_\theta(q\mid\mathbf{e})-1\right)p_\theta(q\mid\mathbf{e})\begin{pmatrix}\left(p_\theta(x,\pi_x\mid q,\mathbf{e})-\theta_{x|\pi_x}p_\theta(\pi_x\mid q,\mathbf{e})\right)- \\ \left(p_\theta(x,\pi_x\mid\mathbf{e})-\theta_{x|\pi_x}p_\theta(\pi_x\mid\mathbf{e})\right)\end{pmatrix}$$

## 3.2.3 Expectation Maximization

*Expectation Maximization* (EM) (Dempster, Laird and Rubin 1977) (McLachlan and Krishnan 1997) is one of the most widely used algorithms for ML estimation with incomplete data. We briefly describe how EM works for ML estimation of BN parameters and refer interested readers to (McLachlan et al. 1997).

Each iteration of EM involves two steps, an expectation step and a maximization step. The expectation step finds the expected sufficient statistics $N_{x,\pi}$ based on the current estimate of parameters $\Theta$.

$$E_\Theta(N_{x,\pi_x}) = \sum_{\mathbf{t}\in\mathbf{D}} p_\Theta(x,\pi_x\mid\mathbf{t})$$

Like the gradient calculation of LL, this step also requires one inference for each family for each tuple.

Maximization step is simple. Since we know the expected frequency from the expectation step, we can now use LM to estimate a better set of parameters $\Theta'$.

$$\theta'_{x\pi_t} = \frac{E_\Theta\left(N_{x,\pi_t}\right)}{E_\Theta\left(N_{\pi_t}\right)}$$

We repeat this two-step process until we are satisfied with the results.

We call this method LEM (for optimizing likelihood using EM algorithm) in the following discussions.

### 3.2.4 Inference

Many NP-hard BN inferences are required in the evaluation functions and gradient calculations. They are by far the most expensive operations in all of the iterative learning methods discussed above. All other operations are trivial comparing to BN inferences. Therefore, the efficiency of these iterative learning methods depends on the number of inferences they require.

We used *junction tree* algorithm (Huang and Darwiche 1996; Jensen 1996) for BN inferences. Junction tree algorithm is very efficient for multiple queries with the same evidence – after the initial construction of the cluster tree structure and two probability propagations to make the probabilities consistent, we can infer the probability of any variable or family with little extra computation. This remains true until we change the evidence or the BN. Thus, the total amount work needed by inference depends on the number of different evidence assignments we have. For each case in the data, the ML approaches use only one set of evidence **t**, while the DEPB approaches each uses two sets of evidence, $\langle \mathbf{e} \rangle$, and $\langle q,\mathbf{e} \rangle$. Thus, DEPB seems to be twice as expensive as ML approaches. However, as discussed above, it is possible for DEPB to omit irrelevant parts of a BN, which may significantly reduce the number of inferences needed; this is especially true for large BNs with sparse structure. Typically, by focusing on the queries, DEPB also converges much faster than ML approaches.

The actual implementation of inference is based on JavaBayes (Cozman 2001), with various efficiency improvements and modifications to make it suitable for our task.

# Chapter 4

# Examples and Empirical Results

We argued that DEPB is usually more computationally efficient and provides more accurate predictions than ML approaches. To test the claim, we implemented all algorithms described in the previous chapter, and compared them empirically. We look at some of these empirical evidences in this chapter. We focus on the comparison between ML and MCL, because MCL is most closely related to ML We found the performance of MCE and MSE approach are similar to MCL approach.

## 4.1 Artificial Network

### 4.1.1 Computational Efficiency

In the special case of complete data and correct BN structure, ML estimation is very fast, and it often performs better than MCL. There is only one optimum for the likelihood function, and ML approach can find it easily using the frequency based estimation. In addition, ML approach has several other advantages, including intuitiveness and extensibility as mentioned before.

If the data is incomplete, however, the advantage of ML approach is less significant. In fact, MCL approach is often more computationally efficient by focusing on certain queries of interest. Consider the example BN shown in Figure 4.1, where a query variable is separate from many evidence variables in a complex BN.

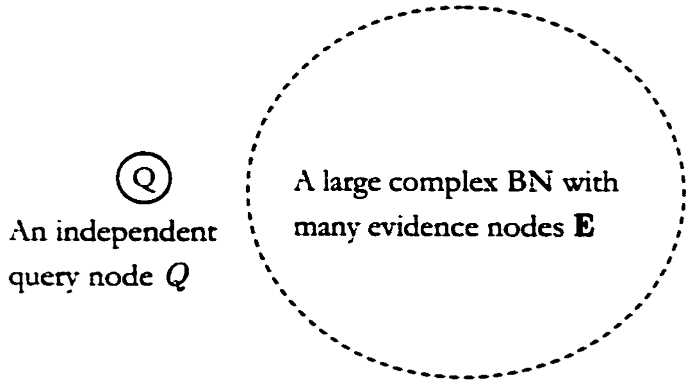

Q
An independent query node $Q$
A large complex BN with many evidence nodes **E**

Figure 4.1 An example to illustrate the efficiency of MCL estimation

In this example, MCL approach is very efficient in finding the parameter maximize $p(Q|\mathbf{E})$, as all evidence variables $\mathbf{E}$ are irrelevant to the query, and MCL learner simply ignores them. ML approach, however, needs to do much more computational work on the evidence variables, as it has to maximize $p(Q,\mathbf{E})= p(Q)\ p(\mathbf{E})$.

### 4.1.2 Accuracy with Incorrect Structure

We mentioned that the ML approach might not work well if the BN structure is incorrect. We designed a simple experiment to investigate how the "incorrectness" of the structure affects the performance. Consider the BNs shown in Figure 4.2.

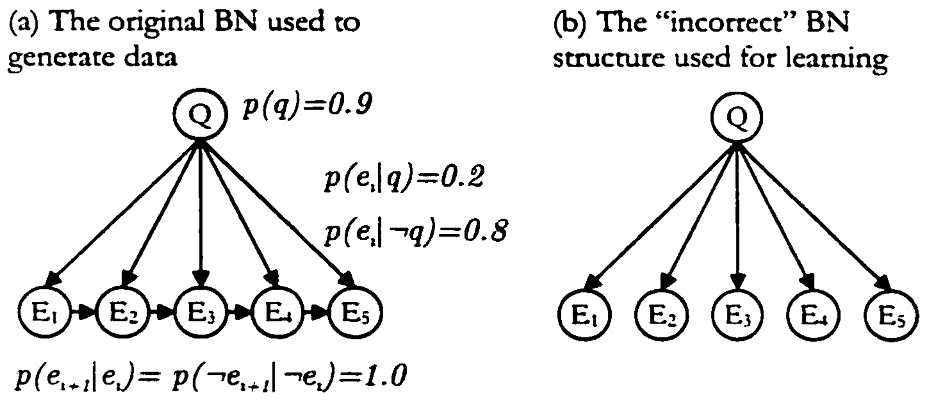

(a) The original BN used to generate data
Q
p(q)=0.9
p(e_i|q)=0.2
p(e_i|¬q)=0.8
E1
E2
E3
E4
E5
p(e_{i+1}|e_i)= p(¬e_{i+1}|¬e_i)=1.0
(b) The "incorrect" BN structure used for learning
Q
E1
E2
E3
E4
E5

Figure 4.2 An example to show the effect of incorrect structure

The BN in Figure 4.2a is our true model. Here, the presence of a link between each pair of evidence variables indicates that they are equivalent, i.e., the evidence are redundant. The parameters are as shown in the figure. We first generate complete data from this BN according to the distribution encoded in this model. We then use the data to estimate the parameters for the BN structure shown in Figure 4.2b, which is a naïve Bayes (NB) with all evidence variables independent given the query variable $Q$. We will look at NB structure more closely in Section 4.2.1. Our goal is most accurately predicting $q$ given $\mathbf{e}$ for all queries.

We consider a range of original structures, by varying the number of evidence-evidence arcs from $E_i$ to $E_{i+1}$ in the BN in Figure 4.2a. We use the number of missing arcs in NB structure as the indicator of the correctness of the NB structure. For example, with 4 evidence-evidence arcs present in the original as in Figure 4.2a, the NB structure is most incorrect with 4 missing arcs. With no evidence-evidence arcs in the original structure, NB is correct with 0 missing arcs.

We estimated the parameters for NB using LM for ML approach and CLGD for MCL approach based on 400 cases and 1 trial. These cases are used as tuples by LM, and they are used as queries by CLGD, with the parent node $Q$ being query variable and children $\mathbf{E}$ being evidence variables. We then evaluate the performance of the resulting NB from ML and MCL approaches using LL, LCL, CE, and MSE scores described earlier. The results are shown in the following graphs. Note the higher score is better in the first two graphs, and the lower score is better in the remaining two graphs.

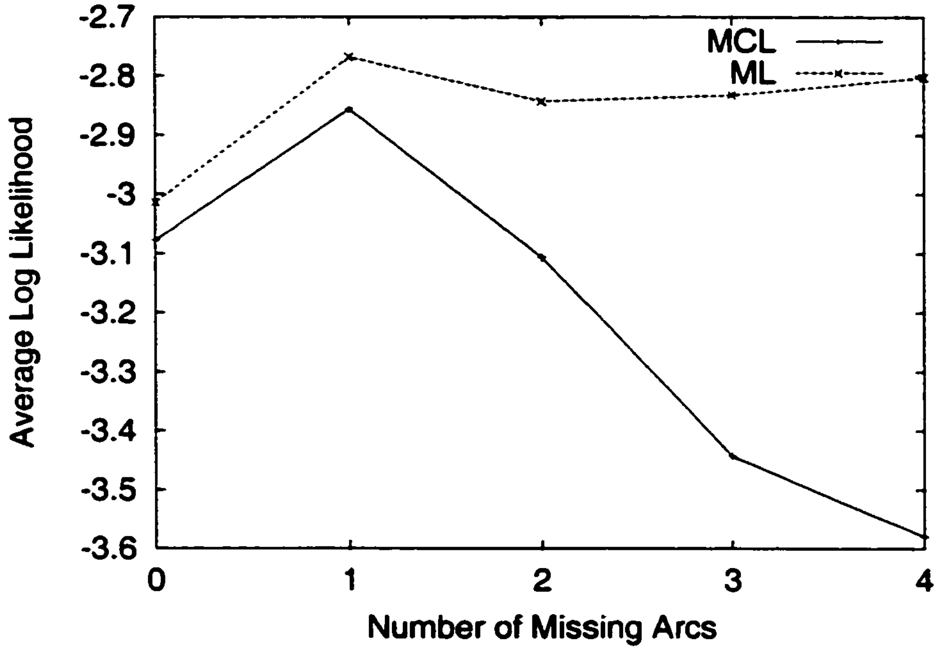


Figure 4.3 Comparing MCL and ML on incorrect structure using LL score

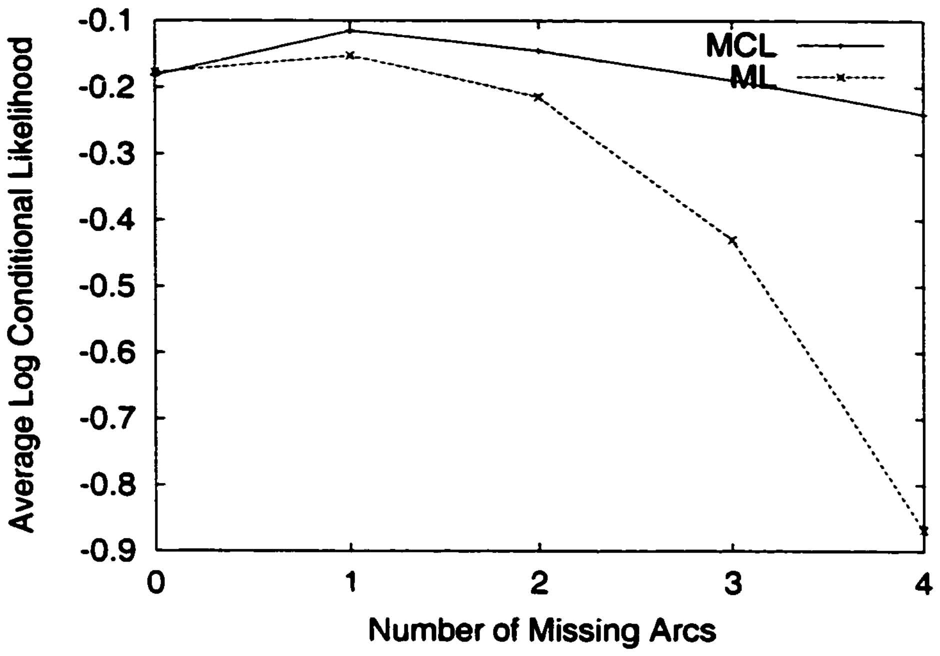


Figure 4.4 Comparing MCL and ML on incorrect structure using LCL score

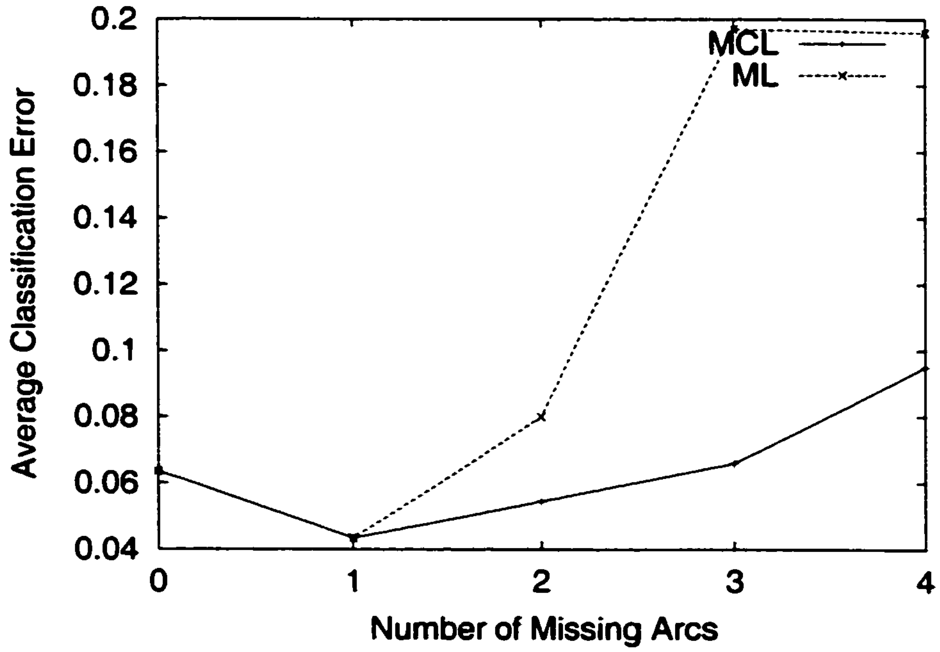


Figure 4.5 Comparing MCL and ML on incorrect structure using CE score

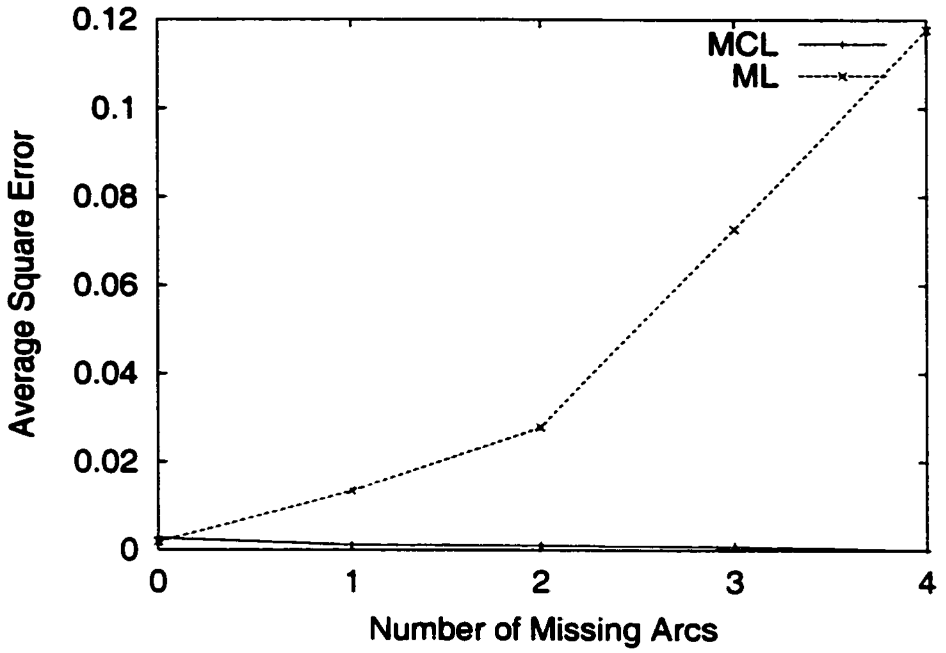


Figure 4.6 Comparing MCL and ML on incorrect structure using MSE score

As expected, the ML approach gets better LL score, and worse LCL, CE and MSE scores. This indicates that ML approach cannot do well on the query $\langle Q|\mathbf{E}\rangle$. The difference becomes greater as the number of missing arcs increases.

This example also shows that the redundant evidence in naïve Bayes creates problem for ML approach. If we know that an evidence variable is redundant, we can remove it – by removing the arc from query node to this variable – to achieve better performance for ML approaches. This is called feature selection (Kohavi and John 1997). We will discuss this again in the following section.

## 4.2 Real World Data

In this section, we look at the performance of these algorithms on real world data.

The following experiments focus on classification problem, as suitable real world data for classification are readily available. Classification is one of the most important problems in artificial intelligence and statistics. Many researchers have recently found that BNs work well as classifiers. One of the simplest BNs, naïve Bayes, performs especially well on classification, often competitive with other state-of-the-art classifiers(Langley et al. 1992). However, essentially all of the associated learners use LM to fill the parameters.

The goal of classification is to predict the class $q$ of a case given some observed attributes **e**. To use a BN as a classifier, we first use BN to infer the conditional probability $p(Q|\mathbf{e})$, and then use the classification strategy $class_{\Theta}(\mathbf{e}) = \arg\max_{q' \in V_Q} p_{\Theta}(q' \mid \mathbf{e})$ to pick a class value $q$. We use the classification error from (3.6) to evaluate classifiers.

The following experiements use the same 25 datasets that are used by (Friedman et al. 1997) as shown in Table 4-1. 23 of these are from UCI data repository (Blake and Merz 1998), plus *mofn-3-7-10* and *corral*, which were developed by (Kohavi et al. 1997) to study feature selection. To deal with continuous variables, we implemented supervised entropy discretization (Fayyad and Irani 1993). To evaluate our classifiers, we use holdout method for the large datasets, and five-fold cross validation for small datasets, as suggested by (Kohavi 1995). As our datasets and testing procedures followed (Friedman et al. 1997), we can compare our results against the other classifiers they considered.

|  | Dataset | # Attributes | # Classes | # Instances |  |
|---|---|---|---|---|---|
|  |  |  |  | Train | Test |
| 1 | australian | 14 | 2 | 690 | CV-5 |
| 2 | breast | 10 | 2 | 683 | CV-5 |
| 3 | chess | 36 | 2 | 2130 | 1066 |
| 4 | cleve | 13 | 2 | 296 | CV-5 |
| 5 | corral | 6 | 2 | 128 | CV-5 |
| 6 | crx | 15 | 2 | 653 | CV-5 |
| 7 | diabetes | 8 | 2 | 768 | CV-5 |
| 8 | flare | 10 | 2 | 1066 | CV-5 |
| 9 | german | 20 | 2 | 1000 | CV-5 |
| 10 | glass | 9 | 7 | 214 | CV-5 |
| 11 | glass2 | 9 | 2 | 163 | CV-5 |
| 12 | heart | 13 | 2 | 270 | CV-5 |
| 13 | hepatitis | 19 | 2 | 80 | CV-5 |
| 14 | iris | 4 | 3 | 150 | CV-5 |
| 15 | letter | 16 | 26 | 15000 | 5000 |
| 16 | lymphography | 18 | 4 | 148 | CV-5 |
| 17 | mofn-3-7-10 | 10 | 2 | 300 | 1024 |
| 18 | pima | 8 | 2 | 768 | CV-5 |
| 19 | satimage | 36 | 6 | 4435 | 2000 |
| 20 | segment | 19 | 7 | 1540 | 770 |
| 21 | shuttle-small | 9 | 7 | 3866 | 1934 |
| 22 | soybean-large | 35 | 19 | 562 | CV-5 |
| 23 | vehicle | 18 | 4 | 846 | CV-5 |
| 24 | vote | 16 | 2 | 435 | CV-5 |
| 25 | waveform-21 | 21 | 3 | 300 | 4700 |

Table 4-1 Description of datasets used in the experiments

see (Friedman et al. 1997)

### 4.2.1 Naïve Bayes

As we have already seen in Figure 4.2, a Naïve Bayes (NB) classifier has a very simple structure – the classification node is the root and the parent of all attributes; the attributes are independent given the class. This independence assumption is clearly not true for many problems, and it may results in poor performance of ML classifiers as shown in the previous example. Since we expect that CLGD works better with incorrect structures, we expect CLGD to give better classification results on NB. The following experiments verify this. The classification errors from these experiments are in Table 4-2.

The starting parameter values of iterative algorithms are often important, as a poor starting point can lead to a poor local optimum or slow convergence. To pick a good starting point, we can use LM to initialize the parameters first before applying CLGD. The similar idea is also mentioned in (Ripley 1996). Our empirical results show that this initialization is useful. It requires very little additional computation, sometimes produced better results than random or uniform initialization, and we did not encounter any situation where it gave much worse results. Therefore, all following experiments use LM initialization for CLGD.

We first use one of the larger datasets, CHESS, to illustrate the basic behaviour of the algorithms. We test the two learners across various sample sizes. The results are shown in Figure 4.7. We see CLGD is consistently better than LM on this dataset.

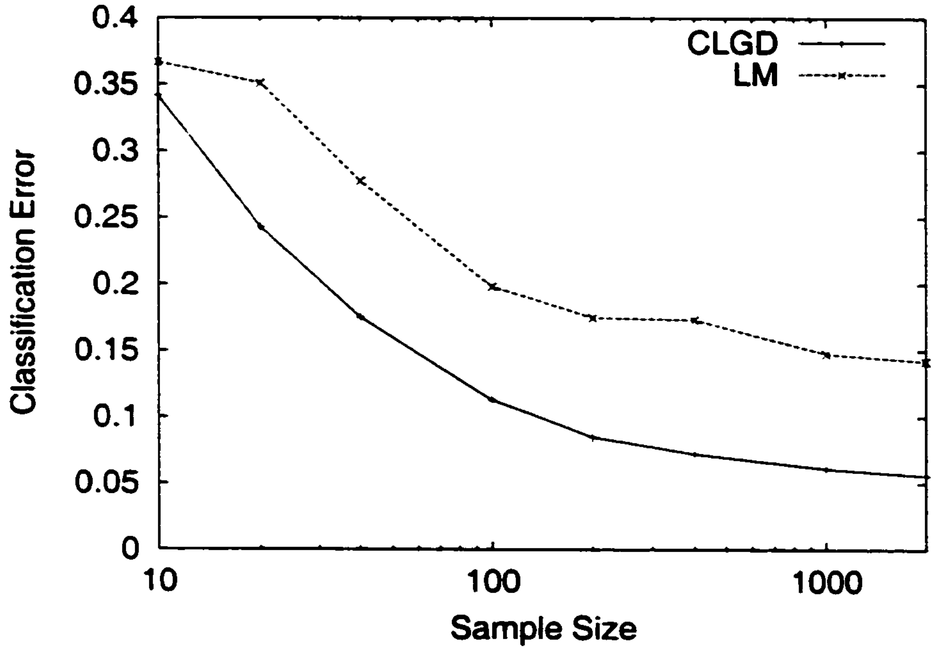


Figure 4.7 CLGD and LM on CHESS dataset

We then run experiments with all 25 datasets. The results are shown in Figure 4.8. In this graph, each point represents a dataset. The x coordinate of the point is the CE of LM on the dataset, and the y coordinate is the CE of CLGD on the dataset. The diagonal line is where the two classifiers give the same error. Each point below the diagonal line indicates that CLGD has smaller error (that is, better performance) than LM on a dataset. Each point above the diagonal indicates that CLGD has worse performance on a dataset. The horizontal and vertical lines around each point express the one standard-deviation error bars in each dimension. From the graph, CLGD clearly dominates LM. Based on one sided paired-t test, CLGD performs better than LM with a level of significance of 0.001.

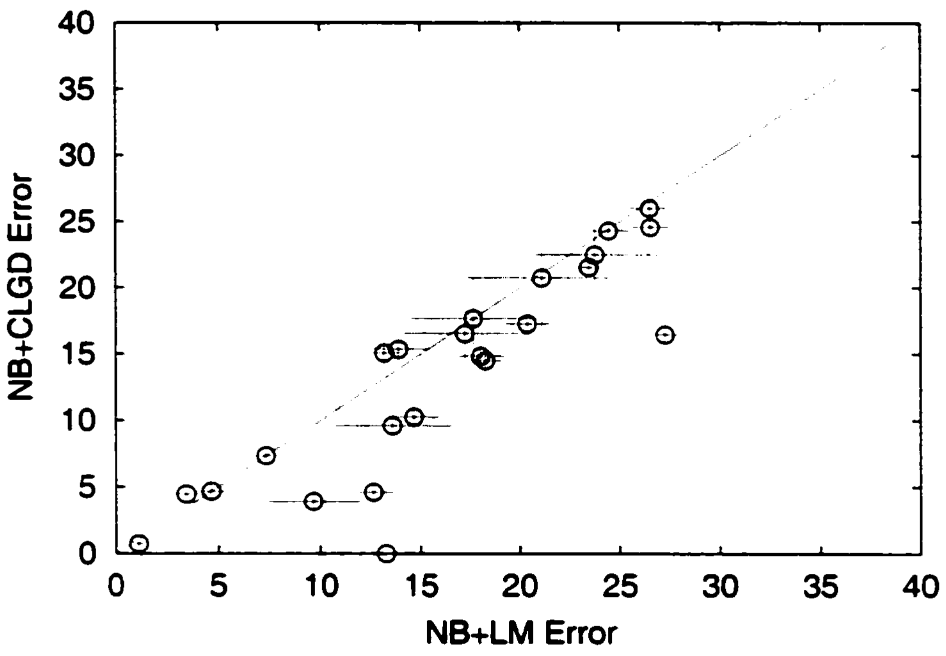


Figure 4.8 Comparing CLGD and LM on UCI datasets using naive Bayes structure

We also compare NB+CLGD (NB structure with CLGD parameter estimation) classifier with other classifiers, SNB+LM and C4.5. The classification error of these two classifiers are taken from (Friedman et al. 1997). Here we omitted three datasets, where we obtained significantly different classification errors on our common classifiers (NB+LM, TAN+LM). In particular, these three differences are more than four standard deviations large. We suspect these three significant differences are caused by different discretization implementations and/or different cross validation split.

| | Data set | NB+LM | NB+CLGD | TAN+LM | TAN+CLGD | SNB+LM | C4.5 |
|---|---|---|---|---|---|---|---|
| 1 | australian | 13.19±0.84 | 15.07±1.06 | 14.93±1.09 | 14.93±1.09 | 13.33±1.81 | 14.35±1.82 |
| 2 | breast | 3.45±0.83 | 4.46±0.48 | 3.45±0.58 | 3.88±0.37 | 3.81±0.63 | 5.27±0.59 |
| 3 | chess | 12.66±1.02 | 4.60±0.64 | 7.60±0.81 | 2.91±0.51 | 5.72±0.71 | 0.47±0.21 |
| 4 | cleve | 17.67±3.27 | 17.67±3.89 | 18.00±3.78 | 18.67±4.33 | 21.94±2.41 | 26.69±0.63 |
| 5 | corral | 13.60±2.99 | 9.60±1.60 | 0.80±0.80 | 0.00±0.00 | 16.43±3.15 | 2.31±2.31 |
| 6 | crx | 13.91±1.49 | 15.36±1.15 | 14.93±1.14 | 14.93±1.14 | 14.08±1.08 | 13.78±0.58 |
| 7 | diabetes | 24.44±1.32 | 24.31±1.26 | 24.18±1.17 | 24.05±1.25 | 23.96±0.83 | 23.96±0.85 |
| 8 | flare | 20.38±1.17 | 17.28±2.22 | 16.71±1.75 | 17.65±2.14 | 16.60±1.67 | 17.45±1.75 |
| 9 | german | 26.50±1.45 | 26.00±1.28 | 26.40±0.94 | 26.40±0.94 | 26.30±2.02 | 27.80±1.23 |
| 10 | glass | 58.10±2.45 | 58.10±2.45 | 58.10±2.45 | 58.10±2.45 | 28.02±2.15 | 30.38±1.95 |
| 11 | glass2 | 23.75±3.22 | 22.50±3.34 | 24.38±3.03 | 23.75±3.22 | 20.83±1.71 | 23.33±1.63 |
| 12 | heart | 21.11±3.91 | 20.74±3.81 | 18.89±3.38 | 20.00±3.68 | 18.15±2.83 | 18.89±3.77 |
| 13 | hepatitis | 18.06±1.29 | 14.84±0.79 | 14.84±2.41 | 14.84±1.64 | 10.00±4.24 | 13.75±4.15 |
| 14 | iris | 4.67±0.82 | 4.67±0.82 | 5.33±0.82 | 4.67±0.82 | 6.00±1.25 | 6.00±1.25 |
| 15 | letter | 27.28±0.63 | 16.46±0.52 | 15.48±0.51 | 11.10±0.44 | 24.64±0.61 | 22.30±0.59 |
| 16 | lymphography | 17.24±3.27 | 16.55±2.53 | 21.38±1.29 | 20.69±1.09 | 22.28±2.46 | 22.97±1.21 |
| 17 | mofn-3-7-10 | 13.28±1.06 | 0.00±0.00 | 9.28±0.91 | 0.00±0.00 | 12.50±1.03 | 14.45±1.10 |
| 18 | pima | 26.54±1.12 | 24.58±1.00 | 24.58±0.84 | 24.31±0.76 | 25.14±2.61 | 24.87±1.52 |
| 19 | satimage | 18.30±0.86 | 14.50±0.79 | 12.15±0.73 | 11.40±0.71 | 17.95±0.86 | 16.85±0.84 |
| 20 | segment | 14.68±1.28 | 10.26±1.09 | 10.65±1.11 | 10.26±1.09 | 6.75±0.90 | 6.36±0.88 |
| 21 | shuttle-small | 1.14±0.24 | 0.72±0.19 | 0.62±0.18 | 0.62±0.18 | 0.72±0.19 | 0.83±0.21 |
| 22 | soybean-large | 7.35±0.33 | 7.35±0.66 | 8.53±1.27 | 7.35±1.54 | 7.11±1.01 | 8.00±1.11 |
| 23 | vehicle | 43.20±1.46 | 37.28±1.62 | 38.93±1.10 | 35.03±0.68 | 38.64±2.33 | 30.26±1.52 |
| 24 | vote | 9.66±2.32 | 3.91±1.39 | 5.75±2.06 | 4.60±1.50 | 5.29±0.59 | 4.37±0.43 |
| 25 | waveform-21 | 23.45±0.62 | 21.55±0.60 | 23.70±0.62 | 23.26±0.62 | 23.47±0.62 | 25.30±0.63 |

Table 4-2 Classification error of six classifiers over 25 datasets for complete data

SNB and C4.5 results are from (Friedman et al. 1997)

C4.5 is a well-known decision-tree learner (Quinlan 1992). Figure 4.9 shows that NB+CLGD seem to perform better, however only with a significance level of 0.138.

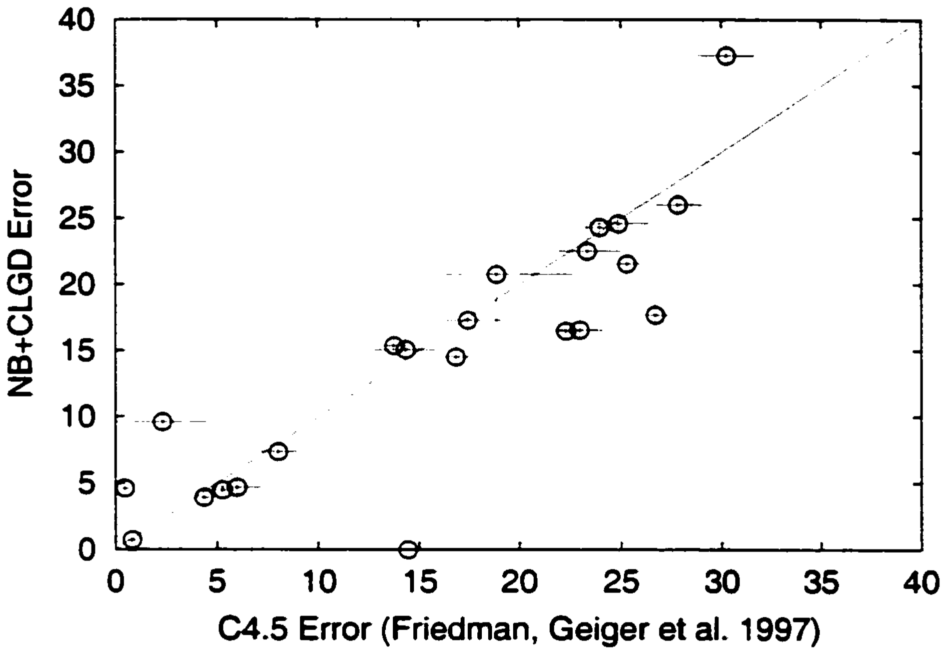


Figure 4.9 Comparing NB+CLGD with C4.5

**Feature selection**

There are two types of unneeded features, irrelevant features and redundant features. As we have seen in the experiment with artificial data, redundant features creates serious problem for NB+LM. By removing redundant features, we can achieve much better results with NB+LM. Because feature selection is often helpful in achieving more accurate and efficient classification, it is an important topic and it is a topic of active research.

Note we are not considering removing irrelevant features here, as irrelevant features do not pose much problem for NB classifiers. The irrelevant features are always discarded automatically, when the learner sets their CPT to be independent of the class values. Also note that the unneeded features pose little computational overhead. Consequently, we only need to consider removing redundant features for feature selection of NB classifiers.

SNB is a naïve Bays containing only a selected subset of the attributes found by a wrapper method described in (Kohavi et al. 1997). This wrapper method searches for an optimal feature subset tailored to a particular problem domain.

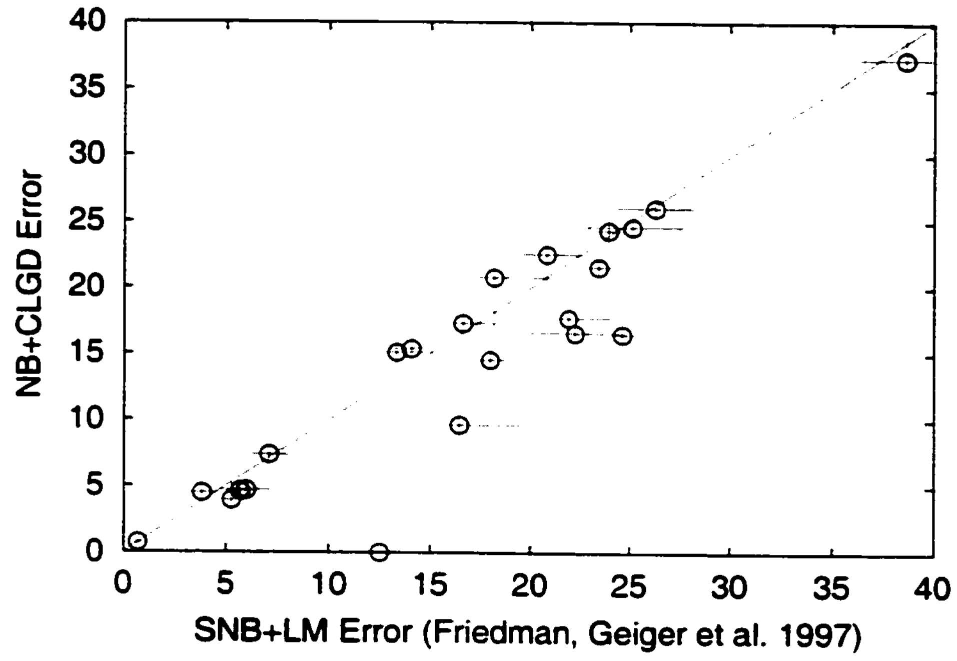


Figure 4.10 Comparing NB+CLGD and SNB+LM

Figure 4.10 shows that NB+CLGD completely dominates SNB+LM, with a significance level of 0.016. This is a little surprising; as we know, feature selection eliminates redundant features and makes SNB+LM much better than NB+LM. So why is SNB+LM still worse than NB+CLGD? This seems to suggest feature selection does not provide any additional benefits for NB+CLGD, i.e., feature selection is subsumed by CLGD. This argument is further supported by CLGD's superior performance on the datasets designed to test feature selection, *corral* and *mofn-3-7-10*, as shown in Table 4-2. NB+CLGD achieved perfect performance on *mofn-3-7-10*. We will soon see that another CLGD system – TAN+CLGD – also achieves perfect performance on *corral*. This is surprising. We did not design CLGD for feature selection, yet it produces perfect results on datasets designed to test feature selection. CLGD does not seem to be affected by redundant features; this can also be seen from the earlier example with redundant evidence.

By definition, DEPB approaches focus on optimizing the learner's performance on predicting $\langle q|\mathbf{e}\rangle$, and the maximum achievable accuracy is the same with or without the unneeded attributes (both redundant and irrelevant variables). As the result, unneeded variables have little effect on DEPB approaches.

On the other hand, LM optimizes the joint likelihood. Each redundant feature is considered as an additional, independent contribution to the joint likelihood. This will certainly produce an incorrect estimation of the CPTs for the redundant feature and result in incorrect joint likelihood. To solve this problem of LM, we must improve the BN structure, by either removing the unneeded features using feature selection so redundant features are not considered multiple times, or adding arcs among the features containing redundant information so redundant features are not considered independently. SNB takes the first approach. Note however, there is always the danger in eliminating useful information. We will now look at the second approach for improving the structure – adding arcs among the attributes. Adding arcs will not eliminate useful attributes, but a small disadvantage is that it will give a more complicated structure.

## 4.2.2 Tree Augmented Naïve Bayes

An obvious enhancement to a NB classifier is to improve its structure by adding arcs among the attributes. One such successful extension is the tree augmented Naïve Bayes (TAN) (Friedman et al. 1997), which uses Chow-Liu's algorithm (Chow and Liu 1968) to add arcs that form a tree among the attributes, so the resulting TAN structure maximizes the likelihood $p(q,\mathbf{e})$; see Figure 4.11 for a TAN structure. (Friedman et al. 1997) has shown that TAN produce excellent classification results. It outperforms NB, SNB, and C4.5.

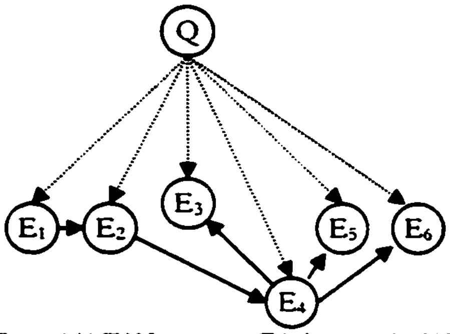


Figure 4.11 TAN structure (Friedman et al. 1997)

We know that CLGD gives significant improvement over LM on incorrect structures. Now that TAN is more general than NB, so we expect CLGD to give less improvement on TAN. However, TAN+CLGD should still outperform TAN+LM, as TAN structure is usually imperfect. We then have an even better classifier than TAN+LM. Motivated by this, we implemented TAN to see if TAN+CLGD is the best classifier.

We first compare NB+CLGD with TAN+LM. Figure 4.12 shows the result. We see that CLGD, even handicapped with the simple NB structure, performs as well as LM on TAN, if not better. (NB+CLGD is slightly better with significance level of 0.241) Of course, the limitations of NB structure may explain the poor performance of NB+CLGD on some datasets. For example, in the artificial dataset *corral*, the class $Q$ is a function of four relevant attributes, $Q=(E_1 \wedge E_2) \vee (E_3 \wedge E_4)$. To represent this function, one must connect these attributes. As NB does not permit such connections, all three NB based classifiers (NB+CLGD, NB+LM, SNB+LM) perform poorly on this data. Of course, as TAN allows more expressive structures, it has a significant advantage here, and achieves almost perfect score. However, it is interesting to note that NB+CLGD is still comparable to TAN+LM in general.

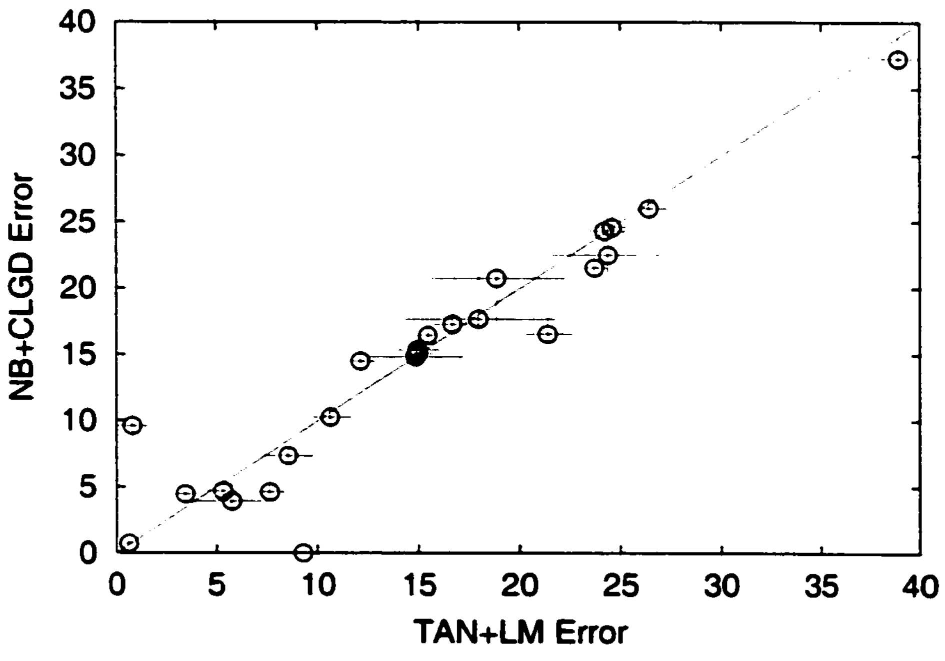


Figure 4.12 Comparing NB+CLGD and TAN+LM

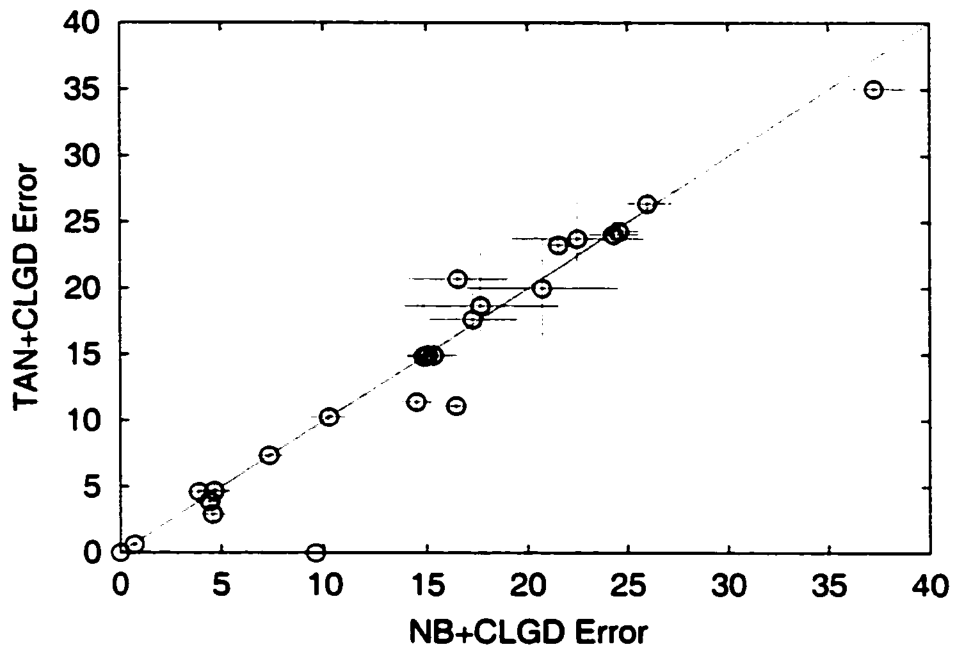


Figure 4.13 Comparing TAN+CLGD with NB+CLGD

We next test the TAN+CLGD classifier. Figure 4.13 shows TAN+CLGD is only slightly better than NB+CLGD with a significance level of 0.125. This is expected and it confirms that the correctness of structure is not very important for CLGD. Notice that TAN+CLGD now gets perfect accuracy on *corral* dataset, the dataset that caused problem for NB+CLGD. Figure 4.14 shows TAN+CLGD is consistently better than TAN+LM with a significance level of 0.015. This is an important result. It shows that that CLGD can indeed give more improvement on the already accurate TAN+LM classifier. This also makes TAN+CLGD the best classifier we tested. Figure 4.15 shows TAN+CLGD is now significantly better than C4.5 with significance 0.032, whereas NB+CLGD was only slightly better. Though this is not surprising, considering NB+LM and TAN+LM are already better than C4.5. Figure 4.16 again verifies that TAN+CLGD is consistently better than SNB+LM with significance 0.018.

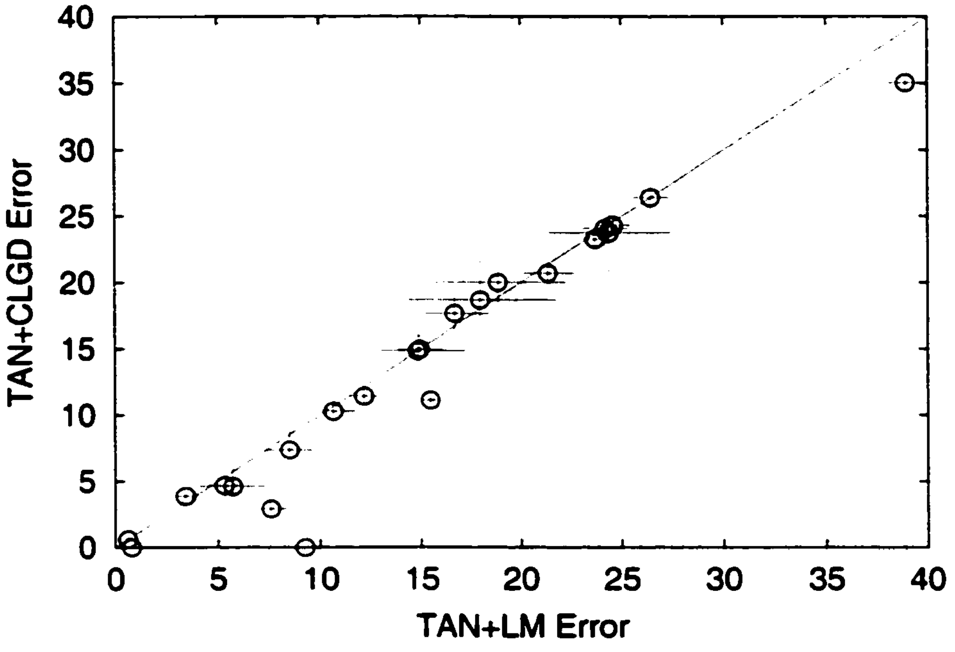


Figure 4.14 Comparing TAN+CLGD with TAN+LM

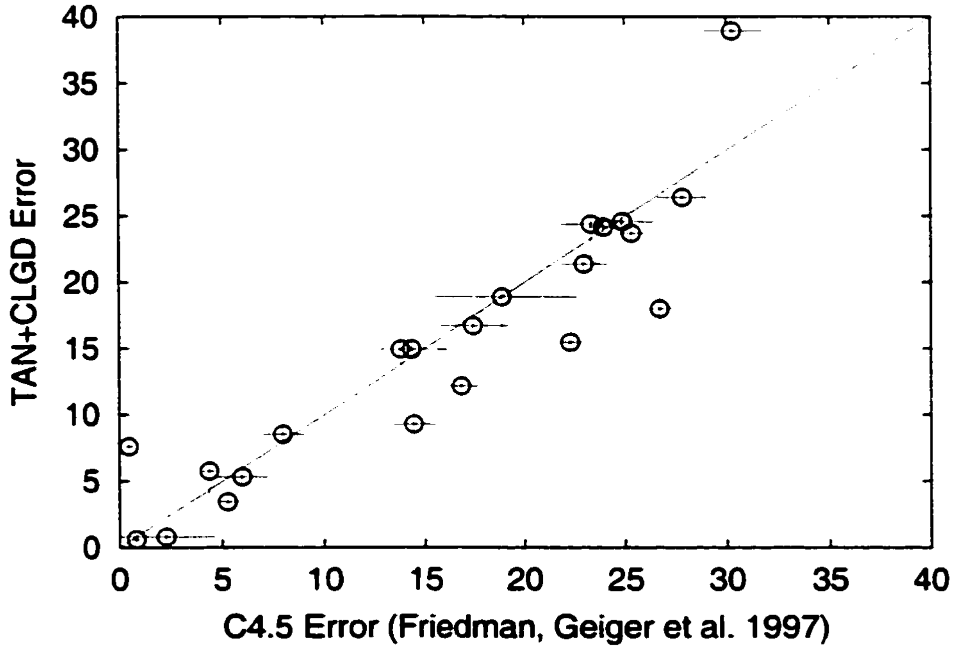


Figure 4.15 Comparing TAN+CLGD with C4.5

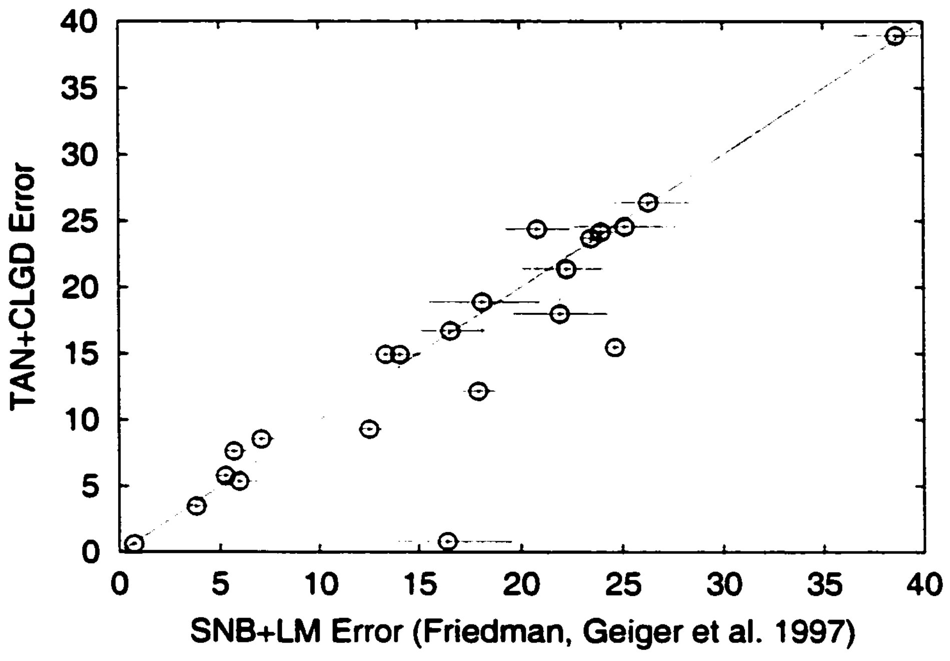


Figure 4.16 Comparing TAN+CLGD with SNB+LM

## 4.2.3 Incomplete Data

A sample case is incomplete if it does not contain values for all variables, i.e., if some variables are unobserved. The data is incomplete if it contains incomplete cases. All the experiments above use complete data. We now compare the performance of CLGD with ML approach on incomplete data.

We use the iterative methods discussed earlier, LEM and LGD, to maximize likelihood for incomplete data. We again use frequency estimations to initialize all the iterative algorithms, LEM, LGD, and CLGD. For example, if the data contains 5 observations of *Smoking=true*, and 10 observations of *Smoking = false*, we set *p(Smoking=true)=5/15*. Note that since some values are missing, we only update the frequency of a family if all variables in the family are observed. Note this simple frequency-based estimation often does not give maximum likelihood estimation. However, it could be a good approximation, especially so if the amount of missing data is very small.

We use the existing 25 datasets from above, and randomly remove the value of each attribute of both training data and testing data with a chance of 1/4. That is, this data is missing completely at random, MCAR (Little and

Rubin 1987). Note that many real missing data are not MCAR. We chose MCAR only for convenience. Also, keep in mind that this MCAR corruption affects CLGD more than ML approach. Consider a complete case with query variable assignment $q$ and evidence $\mathbf{e}$, we remove a value $x$ from evidence $\mathbf{e}$, leaving the remaining evidence $\mathbf{e}'$. ML approach really wants to maximize $p(q,\mathbf{e}) = p(q,\mathbf{e}',x)$, but instead only maximizes $p(q,\mathbf{e}')$. However, this is not too bad: as $p(q,\mathbf{e}',x) = p(q,\mathbf{e}')p(x|q,\mathbf{e}') = p(q,\mathbf{e}')\ p(x|\pi_x)$, we only miss $p(x|\pi_x)$, and we do not need to worry much about it, as it is only the CPT entry in the local family of $x$, and can be easily filled from other cases. On the other hand, CLGD really wishes to maximize $p(q|\mathbf{e}) = p(q|\mathbf{e}',x)$, but instead maximizes $p(q|\mathbf{e}')$. Note $p(q|\mathbf{e}',x) = p(q|\mathbf{e}')p(x|q,\mathbf{e}')/p(x|\mathbf{e}') = p(q|\mathbf{e}')p(x|\pi_x)/p(x|\mathbf{e}')$. We are not as lucky here, as the denominator $p(x|\mathbf{e}')$ involves all other attributes. Thus, $p(q|\mathbf{e}')$ is not just a simple component of $p(q|\mathbf{e}',x)$ as in the LM situation. Maximizing $p(q|\mathbf{e}')$ may even have negative effect to $p(q|\mathbf{e}',x)$.

Despite the above analysis, which shows that the parameters that maximize $p(q|\mathbf{e}')$ need not contribute to maximize $p(q|\mathbf{e}',x)$, we tested the algorithms on MCAR data to see what happens. All results of experiments on incomplete data are shown in Table 4-3. Figure 4.17 and Figure 4.18 shows NB+CLGD still performs consistently better than NB+LGD and NB+LEM, with significance 0.024 and 0.012 respectively.

|  | Data set | NB+LEM | NB+LGD | NB+CLGD | TAN+LEM | TAN+LGD | TAN+CLGD |
|---|---|---|---|---|---|---|---|
| 1 | australian | 16.38±1.69 | 16.38±1.60 | 16.67±1.65 | 17.10±1.48 | 16.38±1.20 | 17.68±0.96 |
| 2 | breast | 16.40±6.43 | 16.40±6.43 | 16.69±6.29 | 16.26±6.49 | 16.40±6.43 | 16.26±6.49 |
| 3 | chess | 18.11±1.18 | 17.92±1.17 | 15.38±1.11 | 15.20±1.10 | 15.38±1.11 | 12.76±1.02 |
| 4 | cleve | 19.00±3.40 | 18.00±3.70 | 19.67±2.07 | 19.33±4.07 | 18.33±4.74 | 19.67±3.89 |
| 5 | corral | 21.60±2.71 | 21.60±2.71 | 21.60±2.40 | 14.40±2.40 | 11.20±3.44 | 13.60±2.71 |
| 6 | crx | 17.25±0.77 | 17.25±0.96 | 16.96±0.88 | 17.68±0.75 | 18.70±0.93 | 17.68±0.75 |
| 7 | diabetes | 32.81±2.90 | 32.16±2.84 | 32.16±2.84 | 32.94±3.11 | 32.94±3.11 | 32.94±2.99 |
| 8 | flare | 19.72±1.58 | 20.00±1.53 | 17.28±2.10 | 16.81±2.03 | 18.50±1.64 | 17.46±1.95 |
| 9 | german | 26.90±0.97 | 26.60±1.04 | 27.20±1.22 | 29.50±1.08 | 30.00±0.92 | 29.50±1.08 |
| 10 | glass | 69.52±3.32 | 69.52±3.32 | 69.52±3.32 | 69.52±3.32 | 69.52±3.32 | 69.52±3.32 |
| 11 | glass2 | 44.38±1.53 | 44.38±1.53 | 44.38±1.53 | 44.38±1.53 | 44.38±1.53 | 44.38±1.53 |
| 12 | heart | 27.04±4.04 | 26.67±3.73 | 26.67±3.91 | 27.78±3.56 | 27.78±3.56 | 26.30±3.99 |
| 13 | hepatitis | 17.42±0.79 | 17.42±0.79 | 19.35±2.04 | 20.00±1.88 | 19.35±1.02 | 20.65±1.64 |
| 14 | iris | 51.33±10.98 | 50.67±10.61 | 42.67±11.22 | 51.33±10.98 | 51.33±10.36 | 42.67±11.22 |
| 15 | letter | 96.30±0.27 | 96.30±0.27 | 96.30±0.27 | 96.30±0.27 | 96.30±0.27 | 96.30±0.27 |
| 16 | lymphography | 17.93±3.84 | 17.93±2.97 | 17.93±4.14 | 24.14±1.09 | 24.83±3.68 | 24.14±1.09 |
| 17 | mofn-3-7-10 | 15.43±1.13 | 15.43±1.13 | 11.72±1.01 | 13.38±1.06 | 14.26±1.09 | 12.70±1.04 |
| 18 | pima | 29.54±0.96 | 29.28±1.04 | 29.28±1.04 | 29.41±0.92 | 29.15±1.00 | 29.41±0.92 |
| 19 | satimage | 32.45±1.05 | 32.80±1.05 | 29.35±1.02 | 29.40±1.02 | 28.35±1.01 | 28.75±1.01 |
| 20 | segment | 27.40±1.61 | 27.66±1.61 | 26.88±1.60 | 27.14±1.60 | 27.27±1.60 | 27.40±1.61 |
| 21 | shuttle-small | 21.10±0.93 | 21.10±0.93 | 21.10±0.93 | 21.10±0.93 | 21.10±0.93 | 21.10±0.93 |
| 22 | soybean-large | 11.18±0.91 | 10.44±0.85 | 10.44±1.26 | 23.53±1.12 | 16.76±1.34 | 17.94±1.72 |
| 23 | vehicle | 56.45±2.77 | 56.80±2.92 | 52.66±4.20 | 50.53±4.25 | 50.89±4.51 | 51.48±4.61 |
| 24 | vote | 10.34±2.21 | 10.34±2.21 | 7.36±1.34 | 5.06±0.86 | 4.83±0.92 | 5.75±1.21 |
| 25 | waveform-21 | 29.30±0.66 | 29.15±0.66 | 29.96±0.67 | 29.85±0.67 | 30.32±0.67 | 29.85±0.67 |

Table 4-3 Classification error of six classifiers over 25 datasets for incomplete data

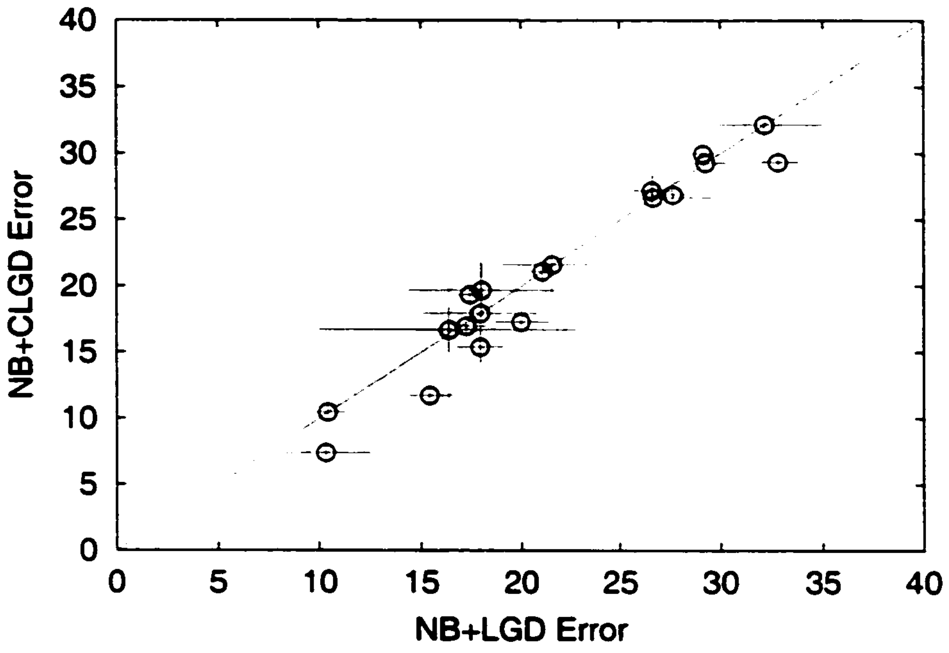


Figure 4.17 Comparing NB+CLGD with NB+LGD

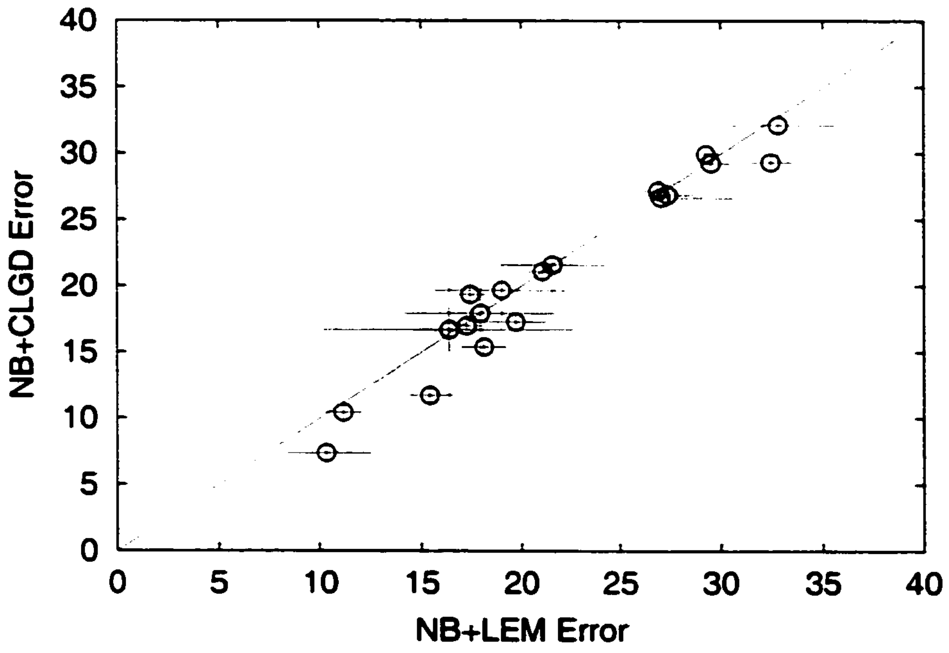


Figure 4.18 NB+CLGD with NB+LEM

We quickly noticed that NB+LEM always converged in a single iteration. This seems to suggest that simple exact solution exist here. We prove below that the frequency-based estimation still gives maximum likelihood on NB.

**Theorem 4.1**

When estimating the parameter $\theta_{e|q}$ of a NB from data **D** where the class variable Q is always observed but some attribute values are missing completely at random, the frequency ratio $N_{e,q} \big/ \sum_{e' \in V_E} N_{e',q}$ is the maximum likelihood estimate for the parameter $\theta_{e|q}$, where $N_{e,q}$ is the number of times $E=e, Q=q$ appeared together in the same case.

**Proof**

For complete data **D**, the likelihood on a naïve Bayes is

$$\begin{aligned} LL_\Theta(\mathbf{D}) &= \sum_{\langle q,\mathbf{e}\rangle \in \mathbf{D}} \log p_\Theta(q,\mathbf{e}) \\ &= \sum_{\langle q,\mathbf{e}\rangle \in \mathbf{D}} \log\left( p_\Theta(q) \prod_{e \in \mathbf{e}} p_\Theta(e \mid q)\right) \\ &= \sum_{\langle q,\mathbf{e}\rangle \in \mathbf{D}} \left( \log p_\Theta(q) + \sum_{e \in \mathbf{e}} \log p_\Theta(e \mid q)\right) \\ &= \left[\sum_{\langle q,\mathbf{e}\rangle \in \mathbf{D}} \log p_\Theta(q)\right] + \sum_{E \in \mathbf{E}} \left[\sum_{\langle q,\mathbf{e}\rangle \in \mathbf{D}} \log p_\Theta(e \mid q)\right] \end{aligned}$$

where $e$ is the value of variable $E$ from observed evidence **e**.

Notice each term in square brackets corresponds to a family in NB. To maximize the likelihood function, we can maximize each term separately. This is not surprising, as it is true for any BN.

Now if any e's are missing, we can simply omit them in the summation because they will not contribute to the likelihood score; i.e., $p_\Theta(Q=q, E=?)$ is just calculated as $p_\Theta(Q=q)$. This means, for incomplete data,

$$LL_{\Theta}(\mathbf{D}) = \left[\sum_{\langle q,e\rangle \in \mathbf{D}} \log p_{\Theta}(q)\right] + \sum_{E \in \mathbf{E}} \left[\sum_{\langle q,e\rangle \in \mathbf{D}, e \neq ?} \log p_{\Theta}(e \mid q)\right]$$

To maximize the overall LL score, we need only to maximize each individual term in the square bracket, whcih corresponds to each family. Taking derivative with respect to parameter $\theta_{e|q}$, we can easily show that each term is maximized by frequency based estimation, setting $\theta_{e|q}=N_{e,q}/N_{q}$, ignoring cases where $e$ is missing. ■

Note this result extends to other BNs, as long as none of the unobserved nodes has an observed child.

We also compared CLGD with LGD and LEM on TAN structure with incomplete data. The original TAN construction algorithm finds a tree that maximizes the conditional mutual information. With incomplete data, we update the mutual information between two attributes only if both attributes are observed. This approach quickly finds trees that approximately maximize the conditional mutual information.

Our experiments did not show much difference when comparing CLGD with LGD and LEM on these TAN structures. This is not surprising, as we know CLGD lost some of its advantage on the better structures, and MCAR data corruption hurts it more than LGD and LEM.

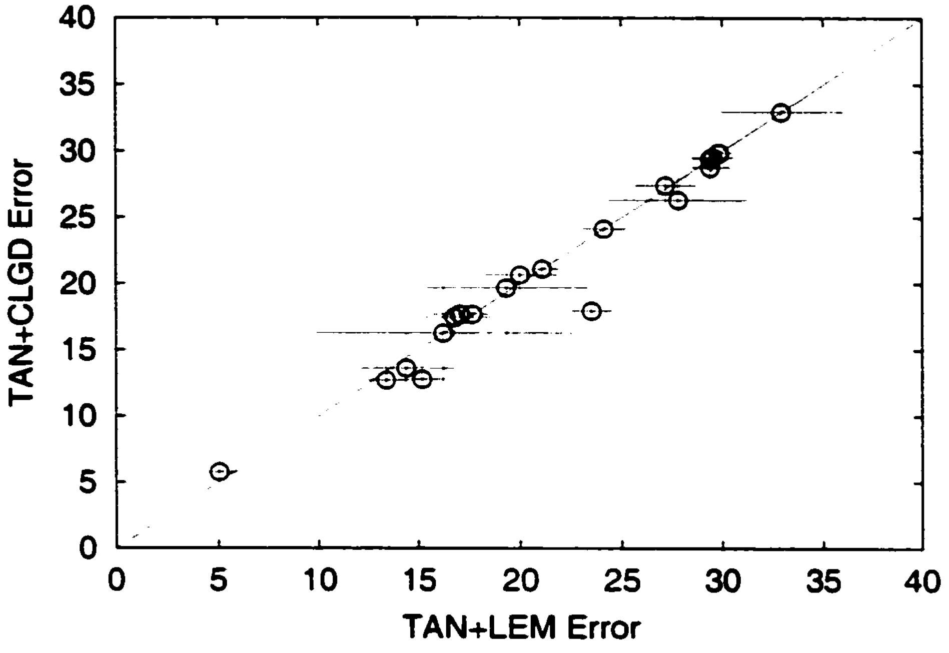


Figure 4.19 Comparing TAN+CLGD with TAN+LEM.

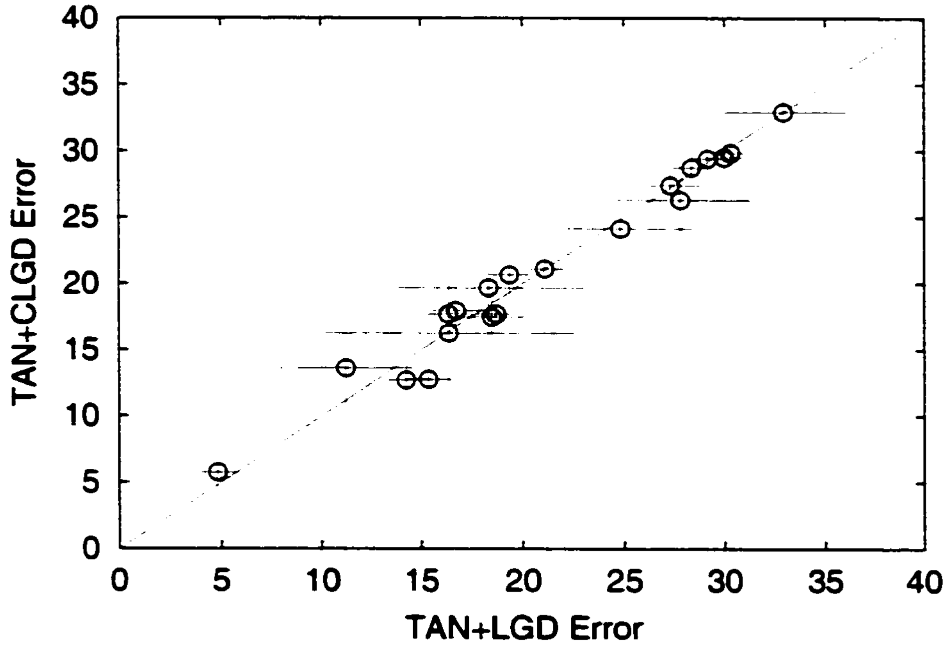


Figure 4.20 Comparing TAN+CLGD with TAN+LGD

We noticed the *letter* dataset gave high classification errors for all classifiers, as shown in Table 4-3. To understand this strange behavior, we ran more experiments on *letter* with other missing amount. The results, shown in Figure 4.21, indicate that the classification error is still a progressively increasing function of amount missing. However, with even very small amount of missing data, all classifiers begin to perform poorly. We believe this is the nature of this dataset, that every feature is important in the prediction.

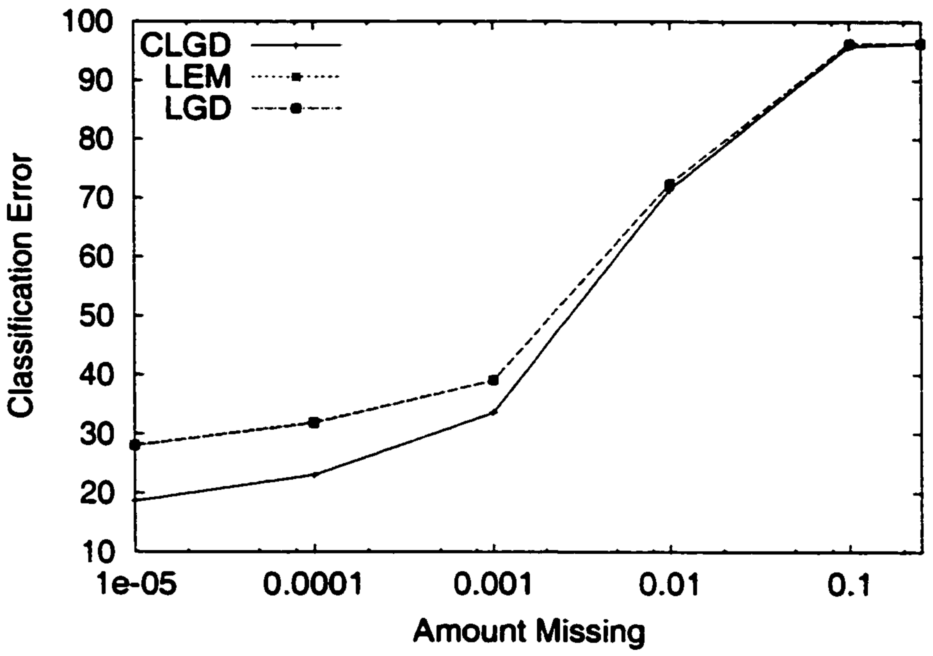


Figure 4.21 The effect of different missing amount on letter dataset

## 4.3 Summary

In this chapter, we have provided empirical evidence to show the remarkable performances of CGLD. The artificial examples illustrated the efficiency and robustness of CGLD. We also compared it against other state-of-art classifiers and found CGLD performed consistently better. Figure 4.22 summarizes all our experimental comparisons. A solid arrow from algorithm $A$ to algorithm $B$ indicates $A$ is significantly better than $B$ with a significance level less than 0.05, whereas a dotted arrow indicates that $A$ is slightly better with a significance level greater than 0.05.

$$\begin{tikzcd}
 & \text{NB+CLGD} \arrow[dl, dotted] \arrow[dd, dotted] \arrow[drrr] \arrow[ddrr] & & \text{TAN+CLGD} \arrow[ll] \arrow[dlll] \arrow[ddll] \arrow[dd] \arrow[dr] & \\
\text{TAN+LM} & & & & \text{NB+LM} \\
 & \text{C4.5} & & \text{SNB+LM} &
\end{tikzcd}$$

Figure 4.22 Comparing all classifiers

The fact that CLGD works well on a variety of real world datasets is especially appealing. It indicates that CLGD is an accurate general-purpose classifier, immediately applicable to many existing problems. We anticipate this observation will make BNs even more popular on tasks like classification or diagnosis.

Although our experiments focus on single query variable, our methods obviously work with situations where multiple query variables exist in each case. However, as the number of query variables increases, we can expect the performance of CLGD to degrade, approaching that of ML methods, as it has to consider more tasks.

## 4.3.1 Computational Efficiency

Typically, the time required by each CGLD iteration varied from a fraction of a second for small datasets to several minutes for larger datasets on a Pentium III 500MHz computer. The number of iterations also varied greatly from a couple to dozens. The computation time of our current implementation CLGD is roughly comparable to the incomplete data algorithms, LGD and LEM. Although LGD and LEM require only one inference for each variable and each case, which is a half of CLGD's two inferences, CLGD can however save much computation by ignoring irrelevant parameters in the network as discussed earlier. In addition, by focusing on specific queries rather than the entire joint distribution, CLGD requires fewer iterations to converge.

On complete data, LM is certainly much faster than CLGD. Still, it is worth noting that CLGD is also quite efficient, as CLGD spends much less computation time on inferences involving complete data (recall that inferences are the most expensive operation in the learning process, see Section 3.2.4).

One possible future research direction is to develop more efficient algorithms for MCL estimation on complete data.

Although LM is extremely fast, it has an additional requirement – the correctness of structure. We have shown that LM can perform poorly without a correct structure. Unfortunately, it is often difficult to obtain a correct structure. Hence, if we also consider the expense of constructing a good structure, LM becomes very inefficient.

# Chapter 5

# Discussions and Future Work

## 5.1 Related Work

Many researchers have worked on learning Bayesian nets. Much of their work focused on learning structures, either for a general Bayesian net, or with the context of some specific class of structures, such as TAN structure, or selective naive Bayes. Essentially all of these learners attempted to optimize likelihood and used fast LM to estimate parameters. See (Heckerman 1995; Buntine 1996) for extensive tutorials.

By contrast, we do not emphasize a good structure, but instead focus on discriminatively estimating parameters that allow BN to perform well on specific tasks such as classification. Instead of estimating parameters by maximizing likelihood, we optimize criterions that directly relate to a BN's performance on answering queries. We considered three such criteria – conditional likelihood, classification error, or square error.

Our method is related to (Binder et al. 1997), which also uses gradient method to learn the parameters of BN. However, they optimized the likelihood score, and used a problematic gradient calculation as noted in Section 3.2.2.

Our work is a significant extension of (Greiner, Grove and Schuurmans 1997), which minimizes square error of answering probabilities. It uses two types of samples – one with tuples, to estimate the value of $p(Q|\mathbf{E})$, and the other with queries, to estimate the probability of seeing the query $\langle Q|\mathbf{E}\rangle$. By contrast, we use only one type of data, the type that is commonly available, for example, in the datasets from UCI repository. As a result, our training methods are more practical, and can be directly applied to many real world problems. In addition, we consider new criteria for discriminatively estimating BN parameters. We also provide many new theoretical and empirical results.

Our results are closely related to the work on discriminatively learning of hidden Markov models. In particular, a gradient method *General Probabilistic Descent* (Katagiri, Lee and Juang 1991) is used to optimize *Maximum Mutual Information* criterion(Bahl, Brown, Souza and Mercer 1986).

While discriminative approaches are new for parameter estimation of Bayesian nets, they have been widely used outside of BN community. For example, a neural net minimizes square error to approximate functions; logistic regression maximizes conditional likelihood for classification. However, many of these systems do not deal with discrete data or incomplete data; many of them do not provide intuitive results. On the other hand, a BN handles discrete data and incomplete data, it has an intuitive structure that is easy for human experts to modify, and it provides intuitive reasoning.

## 5.2 Is Discriminative Approach Always Better?

We have been mainly comparing discriminative estimation of BN parameters to the traditional generative ML estimation of BN parameter. We argued that we could achieve better accuracy in answering queries by learning BNs discriminatively. However, this is not to say that discriminative approaches always achieve better prediction accuracy. In fact, generative BN classifiers such as NB or TAN are known to produce better results than many discriminative classifiers, including C4.5 as we have seen.

### 5.2.1 Bayesian Net for Avoiding Over-fitting

As an argument against discriminative approach, we note that discriminative approaches are prone to *over-fitting*, that is, fitting the model too well on the sample data, producing a system that does not generalize to the actual population. Over-fitting also exists for generative approaches, but it causes much less problem, because generative approaches are not as narrowly focused as discriminative approaches, it is more generalizable by definition – the goal of the generative approach is to produce a general model that resembles the reality and extensible to unseen tasks.

A typical way to reduce over-fitting is limiting the complexity of the model, for example, by using less nodes or arcs for neural nets or Bayesian nets, by pruning decision trees. Most of these complexity reduction methods, however, do not explain why the simpler results are more correct than the more complex ones, believing that the simpler models tend to be more generalizable.

By contrast, generative models such as Bayesian nets provide an intuitive way for controlling the model complexity – one can quickly see whether a BN structure is overly complex by looking at its structure because the BN structure is intuitive. For example, a link between age and gender does not make sense.

This argument shows that DEPB, as well as any learner that uses a generative model, has a unique advantage over other discriminative learners – its model complexity can be controlled easily and intuitively by a human, which helps in reducing over-fitting. We also argued earlier that DEPB is better than the generative approach, by providing better efficiency and accuracy. We believe that a discriminative approach based on a generative model, like DEPB, provides better results than either pure generative approach or discriminative approach. As hinted in the introduction, we recommend this combined approach, building a crude generative model first, and discriminatively refining it. This also supports the idea of using ML to initialize DEPB.

We have provided solid empirical evidence to show that DEPB provides better prediction accuracy than the generative ML approach. We also saw it performed better than one of the popular discriminative classifiers, C4.5. It would be useful to compare it with other discriminative approaches.

There is another way of mixing generative and discriminative approach. For the log likelihood $log\ p(q,\mathbf{e}) = log\ p(q|\mathbf{e}) + log\ p(\mathbf{e})$, we can put more emphasis on maximizing the conditional likelihood $p(q|\mathbf{e})$, and not completely ignoring the likelihood of the evidence $p(\mathbf{e})$. For example, we can use a mixed score, $h(q,\mathbf{e}) = log\ p(q|\mathbf{e}) + \alpha\ log\ p(e)$, where $\alpha \in \mathbb{R}$, $0 \leq \alpha \leq 1$. Note that when $\alpha = 0$, $h(q,\mathbf{e}) = log\ p(q|\mathbf{e})$, when $\alpha = 1$, $h(q,\mathbf{e}) = log\ p(q,\mathbf{e})$. Its derivative is

$$\frac{d \log h_\theta(q \mid \mathbf{e})}{d \beta_{x|\pi_x}} = \left( p_\theta(x, \pi_x \mid q, \mathbf{e}) - \theta_{x|\pi_x} p_\theta(\pi_x \mid q, \mathbf{e}) \right) - \alpha \left( p_\theta(x, \pi_x \mid \mathbf{e}) - \theta_{x|\pi_x} p_\theta(\pi_x \mid \mathbf{e}) \right)$$

It would be interesting to see how a learner performs by varying $\alpha$.

## 5.3 Feature Selection

Feature selection removes irrelevant and redundant features, and allows many classifiers to perform better.

Interestingly, we found that DEPB classifiers are unaffected by irrelevant and redundant features, and produces much better results than a classifier with feature selection. More notably, it performs perfectly on datasets designed to test feature selection. It would be interesting to relate this result to existing work on feature selection, and to extend it beyond NB classifiers.

## 5.4 Model Improvement

If we already have a BN model (built by either a human expert or a computer learner based on ML approach) that is believed to be quite good, a typical question is whether we can improve this BN model further. As we have previously seen, DEPB can usually improve a BN's performance on answering questions, especially so when the BN structure is not perfect. We can use this fact to evaluate and improve our BNs.

If DEPB gives a significant increase in LCL score, we can suspect that the BN structure is poor and needs improvement. In addition, by comparing the parameters estimated by DEPB with the original parameters, we may be able to find out which part of the model needs the most adjustment, in particular, a little change or no change indicates that this part of the model is ok, while a large change may indicate that the involved structure is poor and needs improvement. To locate the problem more precisely, we can compare the performance of the original BN and the BN improved by DEPB on individual testing queries, and find out which parameter changes resulted in the significant improvement for certain queries.

Although MCL approaches may not be good for building a general model from scratch, it may be helpful for improving the model.

## 5.5 Structure Learning

An obvious question following this work is what happens if we use MCL approach to learn BN structures. If we are only interested in producing a BN to answer certain queries, it may be wasteful to build good structure among the variables that do not contribute to the accuracy of answering queries. See Figure 4.1 for an extreme example.

If the data is complete, the MCL approach only needs to consider a tiny subset of all possible structures, as there are many equivalent structures – every structure is equivalent to its substructure that contains only the parents, children, and co-parents of the query node Q, as shown in Figure 5.1; no other nodes affect the conditional probability $p(Q|\mathbf{E})$; and only the CPT of Q and the children of Q are needed.

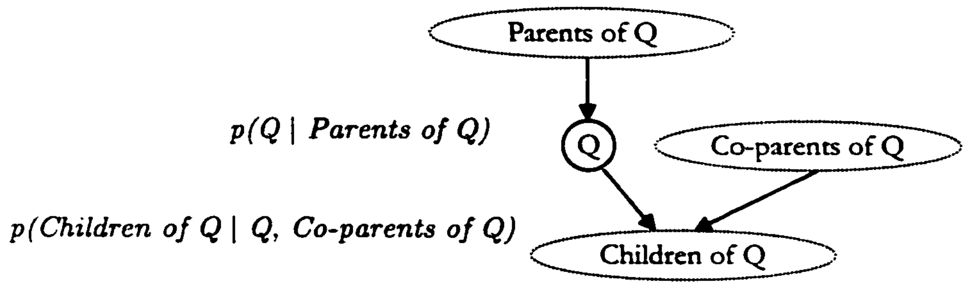


Figure 5.1 MCL structure for complete data

The BN structure is typically found by searching and evaluating using the ML approach. To evaluate each structure based on likelihood score, we estimate the ML parameters for the structure and calculate the likelihood of the data. Now to find a MCL structure, we can simply replace the ML parameter estimation with the MCL parameter estimation. However, our existing algorithm for MCL estimation is expensive; it may not be practical to use it to consider an exponential number of structures. Thus, it is useful to develop more efficient algorithms for MCL parameter estimation. Another direction is to investigate alternative ways of finding MCL structure that require fewer MCL parameter estimations, perhaps by reducing the search space, or by reusing the previous computations.

From the discussion on feature selection, we know connecting query node Q to all other variables does not seem to hurt the prediction accuracy. We can therefore start with a structure that has all nodes linked from Q, and remove the link if later it is found to be unnecessary.

## 5.6 Computational Efficiency

DEPB is already useful with its accurate predictions; it will be even better if we can improve its speed. We know exact DEPB is a NP-hard task in general. It may however be possible to find more efficient approximations. It may also be possible to find efficient exact solutions for some special cases, e.g., when data is complete.

From (3.15),

$$\theta_{x|\pi_x} = \frac{\sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} p_\Theta(x, \pi_x \mid q, \mathbf{e}) - p_\Theta(x, \pi_x \mid \mathbf{e})}{\sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} p_\Theta(\pi_x \mid q, \mathbf{e}) - p_\Theta(\pi_x \mid \mathbf{e})}$$

$$= \frac{\sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} p_\Theta(x, \pi_x \mid q, \mathbf{e}) - p_\Theta(x, \pi_x \mid \mathbf{e})}{\sum_{x' \in V_X} \sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} p_\Theta(x', \pi_{x'} \mid q, \mathbf{e}) - p_\Theta(x', \pi_{x'} \mid \mathbf{e})}$$

The summation in the denominator is the same for all parameters of variable $X$ with the same parents configuration $\pi_x$, thus, $\theta_{x|\pi}$ is proportional to the term in the numerator $\sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} p_\Theta(x, \pi_x \mid q, \mathbf{e}) - p_\Theta(x, \pi_x \mid \mathbf{e})$, which is the total influence of the query value $q$ on the conditional probability $p_\Theta(x, \pi_x|\mathbf{e})$. Therefore, we just need to find out this influence efficiently. Unfortunately, each term in the summation is a function of the parameters $\Theta$ and directly exact evaluation of each term involves NP-hard inference. However, if the data is complete, the inference becomes much easier. The first term $\sum_{\langle q|\mathbf{e}\rangle \in \mathbf{D}} p_\Theta(x, \pi_x \mid q, \mathbf{e})$ is just the count $N_{x,\pi}$, the second term still involves the parameters $\Theta$, but the inference is now a simple calculation, $\frac{\sum_{q' \in V_q} p_\Theta(x, \pi_x, q', \mathbf{e})}{\sum_{q' \in V_q} p_\Theta(q', \mathbf{e})}$.

We can now write down an equation system with one equation for each parameter. We need to investigate an efficient exact or approximate solution to this equation system.

## 5.7 Approximate Inference

Another important future work is to study how approximate inference can improve the efficiency of learning algorithms.

BN inference is NP-hard and it is by far the most expensive operation in our learning algorithms. Note exact inference is often unnecessary; we can tolerate some error in inference, as most of our learning algorithms are approximate algorithms anyway. This is especially true in the initial iterations of learning, where the estimation of parameters is grossly inaccurate; it certainly does not make sense to do exact inference using a poor model.

Since the accuracy of approximate inference needs to be increased as the BN model improves, we can adjust it according to the convergence rate of the learning algorithm.

Using approximate inference may also have a beneficial side effect – the noise introduced by approximation may help in avoiding local optimum.

Note that we can also try to approximate $p_\theta(x, \pi_x | q, \mathbf{e}) - p_\theta(x, \pi_x | \mathbf{e})$ – the influence of query value $q$ on the conditional probability $p_\theta(x, \pi_x | \mathbf{e})$ directly, skipping individual inference on each term. This may save additional computation as the inferences of $p_\theta(x, \pi_x | q, \mathbf{e})$ and $p_\theta(x, \pi_x | \mathbf{e})$ often contain much redundant calculation. We have seen the special case where this influence is 0 when $q$ and $x, \pi$ are conditionally independent given $\mathbf{e}$, here the calculations of two terms are exactly the same.

# Chapter 6

# Conclusion

While BNs are typically constructed by ML approaches, we looked at DEPB – alternative ways to estimate BN parameters discriminatively, by optimizing conditional likelihood, classification error, and square error, and we argued that DEPB provides better prediction results than the standard ML approaches.

We motivated these approaches, noting that DEPB directly maximizes BN's performance on answering queries, and it is more robust.

We noted the general DEPB task is intractable, and presented an effective iterative gradient method.

We provided examples to show the efficiency and robustness of MCL. We also compared CLGD with LM and other state-of-art classifiers on classification of real world data. We found that CLGD performed extremely well, better than all other classifiers we looked at. This suggests that CLGD is a practical classifier and can be directly applied to many real world problems.

**Main Contributions**

The following came out from this research:

- New ways of discriminatively learning BN parameters
- Accurate and practical general purpose classifiers
- New evidences and insights on discriminative learning on generative models

# Bibliography

Bahl, L. R., P. F. Brown, P. V. d. Souza and R. L. Mercer (1986). Maximum mutual information estimation of HMM parameters for speech recognition. Conf. on Acoustics, Speech, and Signal Processing, Tokyo, Japan.

Bilmes, J. (1999). Natural Statistical Models for Automatic Speech Recognition. Dept. of EECS. Berkeley, University of California.

Binder, J., D. Koller, S. J. Russell and K. Kanazawa (1997). "Adaptive Probabilistic Networks with Hidden Variables." Machine Learning Journal **29**(2-3): 213-244.

Blake, C. L. and C. Merz (1998). UCI Repository Of Machine Learning Databases, University of California, Irvine, Dept. of Information and Computer Sciences.

Brent, R. P. (1973). Algorithms for Minimization without Derivatives. Englewood Cliffs, NJ, Prentice Hall.

Buntine, W. (1996). "A guide to the literature on learning probabilistic networks from data." IEEE Transaction On Knowledge And Data Engineering **8**: 195--210.

Chickering, D. M., D. Geiger and D. Heckerman (1994). Learning Bayesian networks is NP-hard, Microsoft Research, Microsoft Corporation.

Chow, C. K. and C. N. Liu (1968). "Approximating discrete probability distributions with dependence trees." IEEE Transaction On Information Theory **14**: 462-467.

Cohen, J. and I. Stewart (1994). The collapse of chaos. New York, Penguin Books.

Cooper, G. (1990). "Computational Complexity Of Probabilistic Inference Using Bayesian Belief Networks (Research Note)." Artificial Intelligence **42**( ): 393-405.

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

Cozman, F. G. (2001). http://www-2.cs.cmu.edu/~javabayes/Home/

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

Fayyad, U. M. and K. B. Irani (1993). Multi-Interval Discretization of Continuous-Valued Attributes for Classification Learning. Proceedings of the Thirteeth International Joint Conference on Artificial Intelligence, San Francisco, Morgan Kaufmann.

Friedman, N., D. Geiger and M. Goldszmidt (1997). "Bayesian Network Classifiers." Machine Learning **29**: 131-163.

Greiner, R., A. Grove and D. Schuurmans (1997). Learning Bayesian nets that perform well. Uncertainty in Artificial Intelligence.

Heckerman, D. (1995). A tutorial on learning Bayesian networks, Microsoft Research.

Hooper, P. M. (2001). "Reference Point Logistic Regression and the Identification of DNA Functional Sites." Journal of Classification **18**.

Huang, C. and A. Darwiche (1996). "Inference in Belief Networks: A procedural guide." International Journal of Approximate Reasoning **15**(3): 225-263.

Jensen, F. (1996). An Introduction to Bayesian Networks, Springer.

Katagiri, S., C. H. Lee and B. H. Juang (1991). New discriminative training algorithms based on the generalized probabilistic descent method. IEEE Workshop on Neural Networks for Signal Processing.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. Proceedings of the Fourteeth International Joint Conference on Artificial Intelligence, San Mateo, CA, Morgan Kaufmann.

Kohavi, R. and G. H. John (1997). "Wrappers for Feature Subset Selection." Artificial Intelligence **97**(1-2): 273-323.

Langley, P., W. Iba and K. Thompson (1992). An analysis of Bayesian classifiers. Tenth National Conference on Artificial Intelligence, Menlo Parc, CA, AAAI Press.

Little, J. A. and D. B. Rubin (1987). Statistical Analysis with Missing Data. New York, Wiley.

McLachlan, G. J. and T. Krishnan (1997). The EM Algorithm and Extensions. New York, John Wiley & Sons.

Mitchell, T. M. (1997). Machine Learning. Boston, McGraw-Hill.

NumericalRecipesSoftware Numerical Recipe in C: the Art of Scientific Computing, Cambridge University Press.

Pearl, J. (1988). Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. San Mateo, Morgan Kaufmann.

Polak, E. (1971). Computational Methods in Optimization. New York, Academic Press.

Quinlan, J. R. (1992). C4.5: Programs for Machine Learning. San Mateo, Morgan Kaufmann Publishers.

Ripley, B. D. (1996). Pattern Recognition and Neural Networks. Cambridge, Cambridge University Press.

Spiegelhalter, D. J., A. P. Dawid, S. L. Lauritzen and R. G. Cowell (1993). "Bayesian Analysis in Expert Systems." Statistical Science **8**(3): 219-247.

Van Allen, T. and R. Greiner (2000). A Model Selection Criteria for Learning Belief Nets: An Empirical Comparison. International Conference on Machine Learning, Palo Alto.