MIND CHANGE OPTIMAL LEARNING: THEORY AND APPLICATIONS

by

Wei Luo

B.Sc., Huazhong University of Science and Technology, 1999

M.Sc., Huazhong University of Science and Technology, 2001

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

SIMON FRASER UNIVERSITY

Fall 2007

APPROVAL

Name: Wei Luo

Degree: Doctor of Philosophy

Title of thesis: Mind Change Optimal Learning: Theory and Applications

Examining Committee: Dr. Anoop Sarkar
Chair

Dr. Oliver Schulte, Senior Supervisor

Dr. Martin Ester, Supervisor

Dr. Arthur Kirkpatrick, SFU Examiner

Dr. Dale Schuurmans, External Examiner,
University of Alberta

Date Approved: Nov 28, 2007

Abstract

Learning theories play a significant role to machine learning as computability and complexity theories to software engineering. Gold's language learning paradigm is one cornerstone of modern learning theories. The aim of this thesis is to establish an inductive principle in Gold's language learning paradigm to guide the design of machine learning algorithms.

We follow the common practice of using the number of mind changes to measure complexity of Gold's language learning problems, and study efficient learning with respect to mind changes. Our starting point is the idea that a learner that is efficient with respect to mind changes minimizes mind changes not only globally in the entire learning problem, but also locally in subproblems after receiving some evidence. Formalizing this idea leads to the notion of *mind change optimality*. We characterize mind change complexity of language collections with Cantor's classic concept of accumulation order. We show that the characteristic property of mind change optimal learners is that they output conjectures (languages) with maximal accumulation order. Therefore, we obtain an inductive principle in Gold's language learning paradigm based on the simple topological concept *accumulation order*.

The new inductive principle enables the analysis of the practical problem of learning Bayes net structure in the rich theoretical framework of Gold's learning paradigm. Bayes net is one of the most prominent formalisms for knowledge representation and probabilistic and causal reasoning. Applying the inductive principle of mind change optimality leads to a unique fastest mind change optimal Bayes net learner. This learner conjectures a graph if it is a unique minimal "independence map", and outputs "no guess" otherwise.

As exact implementation of the fast mind change optimal learner for learning Bayes net structure is NP-hard, mind change optimality can be approximated with a hybrid criterion for learning Bayes net structure. The criterion combines search based on a scoring function with information from statistical tests. We show how to adapt local search algorithms

to incorporate the new criterion. Simulation studies provide evidence that one such new algorithm leads to substantially improved structure on small to medium samples.

Keywords: Learning theory, inductive inference, mind change, accumulation order, inductive principle, Bayes net, conditional independence, constraint-based learning, score-based learning

To Xinyue and Huanhuan!

Acknowledgments

I am truly grateful to my supervisor, Dr. Oliver Schulte, for his continuous support, guidance, encouragement and patience throughout my Ph.D. work. Oliver taught me not only point-set topology, but also how to write in correct English.

My thanks go to Dr. Martin Ester for giving out valuable advice on my study and in particular this thesis.

My thanks go to Dr. Dale Schuurmans and Dr. Ted Kirkpatrick for their time and work on examining the thesis, and more importantly to me, their genuine interest in my work.

My thanks go to Dr. Anoop Sarkar. Although he is not a member of my supervisory committee, he actively participated my depth exam, thesis proposal, and thesis defense. I enjoy very much the fruitful discussions with Anoop and all the help from him.

My thanks go to Dr. Russell Greiner for his participation in our research project of learning Bayes net. His help was essential in my completion of this thesis.

My thanks go to my friends and colleagues in and out of Simon Fraser University. Especially, I would like to thank Ceci Chen, Peter Chen, Song Gao, Vicky Hu, Duan Lei, Jianyuan Liu, Stephen Tse, Qing Wu, Rocky Zhang, Guangxing Zuo for memorable years at Simon Fraser University. I would like to thank Dr. Dana Angluin and Dr. Daniel Reidenbach for sharing their passion on theoretical research at COLT 2005.

My thanks go to my parents for their constant support. Last but not least, I wish to thank my wife Huanhuan and my daughter Xinyue for their love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

> Amazingly, almost all books on pattern recognition or neural networks include no real or realistic examples.
>
> B. D. Ripley

At the 2005 DARPA Grand Challenge, the Stanford team won the 2-million-dollar prize. Sebastian Thrun, head of the Stanford team, attributed the team's success partly to "[T]he pervasive use of machine learning, both ahead and during the race" [95, p. 691]. Behind the glamour of machine learning, computational learning theory (sometimes simply called learning theory) plays a significant role, as computability and complexity theory does for software engineering. For years, research in learning theory enables new machine learning algorithms that have greatly impacted our daily life. Well known examples include, but are not limited to, VC theory for support vector machine, PAC theory for boosting, and mistake bound theory for the weighted majority algorithm [103].

Gold's language learning paradigm [37](also known as algorithmic learning theory, or inductive inference) is a cornerstone of modern learning theories. The theory and its variants, however, do not provide explicitly an inductive principle that can guide the design of machine learning algorithms. This thesis seeks an inductive principle inside Gold's language learning paradigm by proposing the notion of *mind change optimality* (as known as *mind change efficient learning* [60]). The thesis also studies how to apply the inductive principle to design algorithms for learning Bayes net structure, which is an important problem in Bayesian inference. As Gold's language learning paradigm, Bayesian inference is another

leg of computational learning theory. To my knowledge, this is the first attempt to combine results from these two theories into machine learning algorithm design.

1.1 Background and Motivation

As Freud and Schapire put it [23], much of the work in computational learning theory can be traced to Valiant's work on PAC learning [99], and Gold's work on Language identification in the limit [37]. We situate our study in Gold's paradigm.

1.1.1 Gold's Language Learning Paradigm

In 1967, Gold in his seminal paper [37] proposed the model of **language identification in the limit**. Osherson and Weinstein, from the perspective of cognitive scientists, later named it **formal learning theory** [72, 73, 62, 43]. Kelly [49], Glymour [35], and Schulte [87], from the perspective of philosophers, also use the terms "logical reliability" and "means-ends epistemology". In addition to its impact on theories of natural languages [78, 102], Gold's work convened a research community inside the discipline of computer science, and led to the incarnation of the conference on learning theory (COLT) in 1988, the conference on Algorithmic Learning Theory (ALT) in 1990, and the EuroCOLT in 1993.

In Gold's model, a **language** is a subset of a predefined **universe** of **strings**. Considering a string can be encoding any thing, the theory has wide applicability. We shall give a brief account of Gold's model in terms of the data protocol and the success criterion prescribed by the theory (cf. [49, Chapter 2]). A formal description of the model is postponed to Chapter 2.

Data Protocol Training data are strings in the target language. They are presented to a learner through a **text**, which is an (indexed) enumeration of the language with possible repetitions and pauses. The sequential nature of a text gives a notion of "time steps" in Gold's model. At any step, a learner observes only an initial segment of the text up to that step.

Success Criterion Gold's model requires a learner to make a guess of the target language at each step. The learner **identifies** a language **in the limit**, if, given any text, on all but

finitely many steps, the learner's makes the right guess. Likewise a learner **identifies** a language collection if it identifies every individual language in the collection.

Therefore Gold's language learning paradigm is a general framework; it can be applied to a wide range of machine learning problems where convergence of outputs is of concern.

1.1.2 Mind Change Complexity and Other Classifications of Language Learning Problems

Much of the research in Gold's language learning paradigm is on **learnability** of language collections. With additional constraints on resources available to learners, just as in recursion theory, various complexity notions may arise. A widely adopted notion is the mind change complexity [2, 3, 100, 56, 57, 68, 91, 86, 51]. Formal definition of mind change complexity will be given in Chapter 2. Simply put, a mind change refers to a change of guess along the process of learning, and the mind change complexity of a language collection is the maximal number of mind changes any reliable learner may make in the worst case. Hence mind change complexity provides a natural gradation of learnability.

Since "hardness" of language collection highly depends on the assumptions about learners. When these assumptions change, an unlearnable problem can become a learnable problem, and vice versa. On the other hand, one can argue that the subtle differences among learner models are only of interest to mathematicians and computer theorists. Instead of looking at the "hardness" of language collections, an orthogonal way to classify language collections is to treat a language as a set and directly capture the set-theoretical characteristics of language collections. Examples include the study of thickness [92] and elasticity [105, 66] of language collections. To my knowledge, however, there are very few attempts to *characterize* hardness of language collections using set-theoretical concepts.

1.1.3 Aim of the Thesis

The aim of this thesis is to **establish an inductive principle in Gold's language learning paradigm to guide the design of machine learning algorithms.**

As suggested previously, the theories on Gold's language learning paradigm and its variants are rich and profound. In practice, however, the applications of the theories to *machine learning* are limited in terms of both number and influence. Successful applications of learning theories often depend on a good inductive principle. For examples, structural risk

minimization is an inductive principle that enables successful applications of VC theory; the "Occam's Razor" defined in [12] is an inductive principle that is essential to the applicability of PAC theory. Compared to other learning paradigms, Gold's learning paradigm seems to lack a clear operational inductive principle. In other words, much of previous studies is on the hardness of *problems*, but not on the optimality of *learners*. Only after we have a good idea of what an optimal learner looks like in the Gold's paradigm, can the theory effectively help the design of learning algorithms or strategies.

To achieve the aim of the thesis, I consider the following set of objectives.

To Provide a Refinement to the Theory of Mind Change Complexity Mind change complexity provides a useful hierarchy of language learning problems in terms of their hardness. The formalization, however, does not provide enough intuition on the behaviour of learners because complexity requirements do not constrain learner sufficiently. In other words, two learners that succeed with the same number of mind changes may adopt radically different strategies, with one maybe more natural than the other. Should some refinement constrain a learner to behave more "naturally", an inductive principle will follow naturally.

To Characterize Mind Change Complexity with Set-theoretical Concepts The language of set theory is rigorous and appeals to a broad audience. We shall use concepts from the point-set topology to characterize mind change complexity in language collections. By doing this, we bring together the research on set-theoretical features (e.g.,[92] and [66]) and the research on learning complexity (e.g., [3]). In additional, the optimality behavior of a learner will be better understood when its inputs and outputs are described in terms of set-theoretical concepts or related mathematical notions. Last but not least, the characterization helps translate learning theoretical problems into set theoretical ones, and hence enables the use of existing theorems in set theory.

To Apply the Theory to a *Real-life* Application To evaluate the inductive principle obtained, we shall apply it to a machine learning problem of practical significance. We choose the problem of learning Bayes net structure. The problem serves our purpose because it is an important (see [83]) and real problem (for lack of perfect solution, see [39]).

1.2 Structure of the Thesis

This thesis is organized as follows.

Chapter 2 covers the basic definitions of Gold's language learning paradigm and mind change complexity of language collections. We start with the idea that a learner efficient with respect to mind changes minimizes mind changes not only globally in the entire learning problem, but also locally in subproblems after receiving some evidence. Formalizing this idea leads to an inductive principle: *strong mind change optimality.*

The major theoretical framework of this thesis is built in Chapter 3. After a brief review of point-set topology, we give a characterization of mind change complexity using Cantor's classic concept of *accumulation order.* We use accumulation order to rigorously prescribe the input-output behavior of a mind change optimal learner. The chapter ends with a comparative study of accumulation order with respect to other set-theoretical and recursion-theoretical notions of language collections such as thickness, inclusion depth, elasticity, and intrinsic complexity.

Chapter 4 illustrates the theory with a group of applications in formal language theory. In particular, the optimal learning problems for patterns, one-variable patterns, and fixed-length patterns are examined. These are interesting problems in their own right.

The remaining chapters constitute a major application of the theory: mind change optimal learning of Bayes net structure. In Chapter 5, we reformulate the structure learning problem into a language learning problem so that our theory can be applied. We also show that imposing exact optimality criterion leads to intractability. Chapter 6 shows that existing algorithms for learning Bayes net structure are not mind change optimal. They do not even always produce valid outputs according to the Bayes net theory. This shows consistency between our theory and the Bayes net theory, and suggests a new algorithm using the inductive principle offered by the theory of mind change efficient learning. Chapter 7 describes the algorithm and gives empirical evaluation for it.

The structure of the thesis is demonstrated in Figure 1.1.

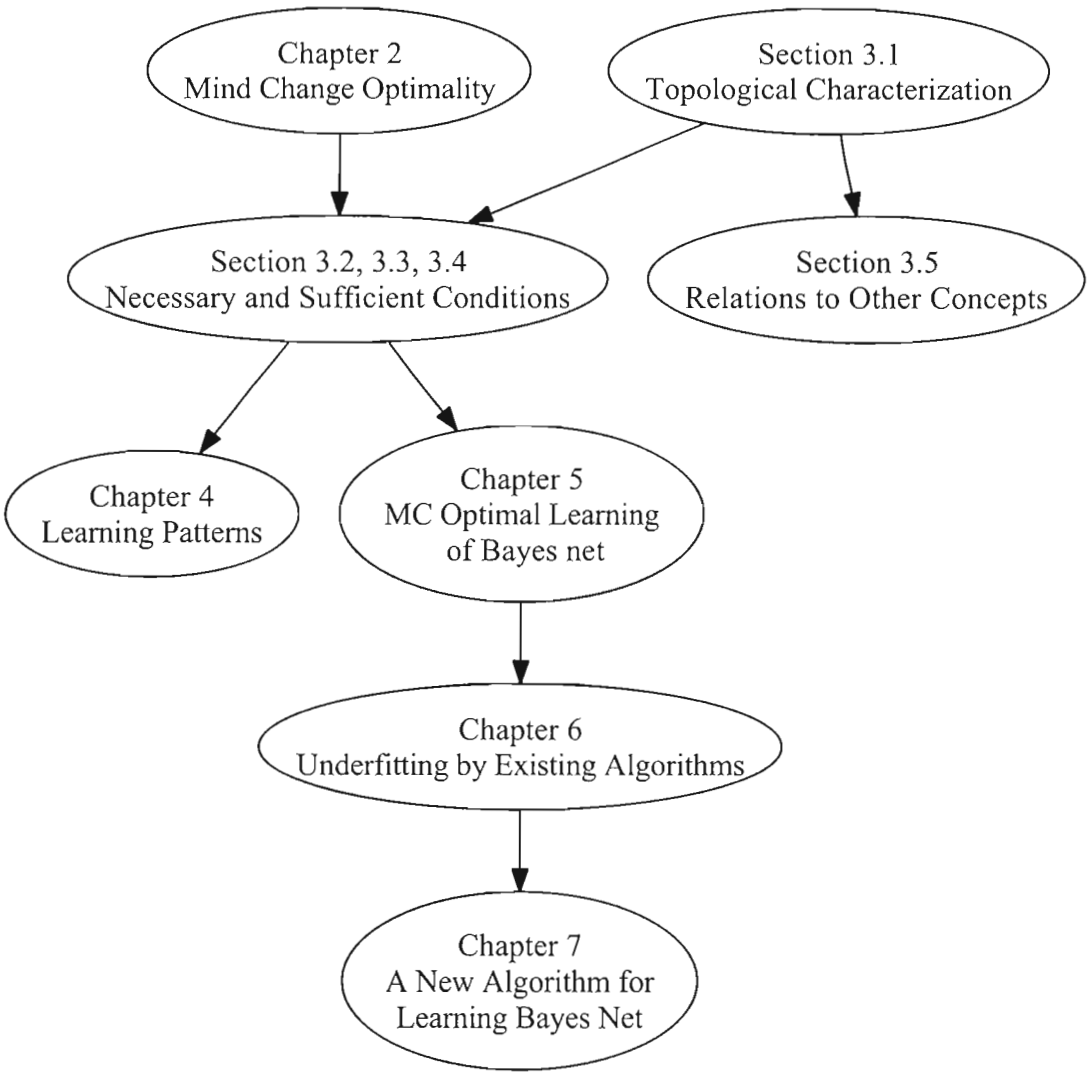


Figure 1.1: Organization of the thesis

Chapter 2

Language Learning and Mind Change Optimality

> Computer Science is just applied mathematics.
>
> a computer scientist

This chapter reviews standard concepts in Gold's paradigm of language identification and presents our definition of mind change optimality.

2.1 Preliminaries: Language Identification

In natural language processing, historically researchers try to model the process of children learning a language by just listening to sentences in that language. In his seminal paper [37], Gold considered the following problem: given a class of candidate languages and a way to present information about the target language to a child, does the child have sufficient information to single out the target language from the class? This setup later evolves into a more general learning paradigm.

We employ notation and terminology from [45], [62, Chapter 1], and [37]. We write $\mathbb{N}$ for the set of natural numbers: $\{0, 1, 2, ...\}$. The symbols $\subseteq, \supseteq, \subset, \supset$, and $\emptyset$ respectively stand for subset, superset, proper subset, proper superset, and the empty set. As in natural language processing, we view a language as a set of strings. We identify strings with natural numbers encoding them. Thus we define a **language** to be a subset of $\mathbb{N}$ and write L

for a generic language [37, page 449]. A **language learning problem** is a collection of languages; we write $\mathcal{L}$ for a generic collection of languages. A **text** T is a mapping of $\mathbb{N}$ (natural numbers representing time steps) into $\mathbb{N} \cup \{\#\}$, where $\#$ is a symbol not in $\mathbb{N}$. (The symbol $\#$ models pauses in data presentation.) We write content(T) for the intersection of $\mathbb{N}$ and the range of T. A text T is **for** a language L if $L = \text{content}(T)$. The initial sequence of text T of length n is denoted by $T[n]$. The set of all finite initial sequences over $\mathbb{N} \cup \{\#\}$ is denoted by SEQ. We also use SEQ($\mathcal{L}$) to denote finite initial sequences consistent with languages in $\mathcal{L}$. We let σ and τ range over SEQ. We write content(σ) for the intersection of $\mathbb{N}$ and the range of σ. The initial sequence of σ of length n is denoted by $\sigma[n]$. We say that a language L is **consistent** with σ if content(σ) $\subseteq L$. We write $\sigma \subset T$ or $T \supset \sigma$ to denote that text T extends initial sequence σ.

Example 2.1.1.

1. Let $L_i \equiv \{n : n \geq i\}$, where $i \in \mathbb{N}$; we use COINIT to denote the class of languages $\{L_i : i \in \mathbb{N}\}$ [3, page 324].

2. In the n-dimensional linear space $\mathbb{Q}^n$ over the field of rationals $\mathbb{Q}$, we can effectively encode every vector $\vec{v}$ by a natural number. Then a linear subspace of $\mathbb{Q}^n$ corresponds to a language. We write LINEAR_n for the collection of all (encodings of) linear subspaces of $\mathbb{Q}^n$.

$$\begin{aligned}\{&\{0,1,2,3,...\}\\ &\{1,2,3,...\}\\ &\{2,3,...\}\\ &\{3,...\}\\ &...\\ &\}\end{aligned}$$

Figure 2.1: The language collection COINIT.

A **learner** is a function that maps a finite sequence to a language or the question mark ?, meaning "no answer for now". We normally use the Greek letter Ψ and variants to denote a learner. Our term "learner" corresponds to the term "scientist" in [62, Chapter 2.1.2].

In typical applications we have available a syntactic representation for each member of the language collection $\mathcal{L}$ under investigation. In such settings we assume the existence of an index for each member of $\mathcal{L}$, that is, a function $index : \mathcal{L} \mapsto \mathbb{N}$ (cf. [43, page 18]), and we can take a **learning function** to be a function that maps a finite sequence to an index for a language (learning functions are called "scientists" in [43, Chapter 3.3]). A computable learning function is a **learning algorithm**. We use the general notion of a learner for more generality and simplicity until we consider issues of computability.

Let $\mathcal{L}$ be a collection of languages. A learner Ψ **for** $\mathcal{L}$ is a mapping of SEQ into $\mathcal{L} \cup \{?\}$. Thus the learners we consider are class-preserving; for the results in this dissertation, this assumption carrics no loss of generality. Usually context fixes the language collection $\mathcal{L}$ for a learner Ψ.

We say that a learner Ψ **identifies** a language L on a text T for L, if $\Psi(T[n]) = L$ for all but a finite number of stages n. Next we define identification of a language collection relative to some evidence.

Definition 2.1.1. A learner Ψ identifies $\mathcal{L}$ given $\sigma \iff$ for every language $L \in \mathcal{L}$, and for every text $T \supset \sigma$ for L, we have that Ψ identifies L on T.

Thus a learner Ψ identifies a language collection $\mathcal{L}$ if Ψ identifies $\mathcal{L}$ given the empty sequence Λ.

Example 2.1.2.

1. The following learner Ψ_{CO} identifies COINIT: If content$(\sigma) = \emptyset$, then $\Psi_{\mathrm{CO}}(\sigma)$:=?. Otherwise set $m :=$ min(content(σ)), and set $\Psi_{\mathrm{CO}}(\sigma) := L_m$.

2. Let vectors(σ) be the set of vectors whose code numbers appear in σ. Then define $\Psi_{\mathrm{LIN}}(\sigma) =$ span(vectors(σ)), where span(V) is the linear span of a set of vectors V. The learner Ψ_{LIN} identifies LINEAR_n. The problem of identifying a linear subspace of reactions arises in particle physics, where it corresponds to the problem of finding a set of conservation principles governing observed particle reactions [53, 97]. Interestingly, it appears that the theories accepted by the particle physics community match the output of Ψ_{LIN} [98, 89].

A learner Ψ **changes its mind** at some nonempty finite sequence $\sigma \in$ SEQ if $\Psi(\sigma) \neq \Psi(\sigma^-)$ and $\Psi(\sigma^-) \neq ?$, where σ^- is the initial segment of σ with σ's last element removed [31, 3]. (No mind changes occur at the empty sequence Λ.)

Definition 2.1.2 (based on [3]). Let Ψ be a learner and c be a function that assigns an ordinal to each finite sequence $\sigma \in \mathrm{SEQ}$.

1. c is a **mind-change counter** for Ψ and $\mathcal{L}$ if $c(\sigma) < c(\sigma^-)$ whenever Ψ changes its mind at some nonempty sequence σ. When $\mathcal{L}$ is fixed by context, we simply say that c is a mind change counter for Ψ.

2. Ψ identifies a class of languages $\mathcal{L}$ **with mind-change bound** α given $\sigma \iff \Psi$ identifies $\mathcal{L}$ given σ and there is a mind-change counter c for Ψ and $\mathcal{L}$ such that $c(\sigma) = \alpha$.

3. A language collection $\mathcal{L}$ is **identifiable with mind change bound** α given $\sigma \iff$ there is a learner Ψ such that Ψ identifies $\mathcal{L}$ with mind change bound α given σ.

Example 2.1.3.

1. For COINIT, define a counter c_0 as follows: $c_0(\sigma) := \omega$ if $\mathrm{content}(\sigma) = \emptyset$, where ω is the first transfinite ordinal, and $c_0(\sigma) := \min(\mathrm{content}(\sigma))$ otherwise. Then c_0 is a mind change counter for Ψ_{CO} given Λ. Hence Ψ_{CO} identifies COINIT with mind change bound ω (cf. [3, Sect.1]).

2. For LINEAR_n, define the counter $c_1(\sigma)$ by $c_1(\sigma) := n - \dim(\mathrm{span}(\mathrm{vectors}(\sigma)))$, where $\dim(V)$ is the dimension of a space V. Then c_1 is a mind change counter for Ψ_{LIN} given Λ, so Ψ_{LIN} identifies LINEAR_n with mind change bound n.

3. Let FIN be the class of languages $\{D \subseteq \mathbb{N} : D \text{ is finite}\}$. Then a learner that always conjectures $\mathrm{content}(\sigma)$ identifies FIN. However, there is no mind change bound for FIN [3].

2.2 Strong Mind Change Optimality

In this section we introduce a new identification criterion that is the focus of this dissertation. Our point of departure is the idea that learners that are efficient with respect to mind changes should minimize mind changes not only globally in the entire learning problem but also locally after receiving specific evidence. For example, in the COINIT problem, the best global mind change bound for the entire problem is ω [3, Sect.1], but after observing initial

data $\langle 5 \rangle$, a mind change efficient learner should succeed with at most 5 more mind changes, as does Ψ_{CO}. However, there are many learners that require more than 5 mind changes after observing $\langle 5 \rangle$ yet still succeed with the optimal mind change bound of ω in the entire problem.

To formalize this motivation, consider a language collection $\mathcal{L}$. If a mind change bound exists for $\mathcal{L}$ given σ, we write $\mathbf{MC}_{\mathcal{L}}(\sigma)$ for the least ordinal α such that $\mathcal{L}$ is identifiable with α mind changes given σ. We require that a learner should succeed with $\mathbf{MC}_{\mathcal{L}}(\sigma)$ mind changes after each data sequence $\sigma \in \mathrm{SEQ}(\mathcal{L})$. For example, the learner Ψ_{CO} achieves this performance for COINIT. This leads us to the following definition.

Definition 2.2.1. A learner Ψ is **strongly mind change optimal** for $\mathcal{L}$ if there is a mind change counter c for Ψ such that $c(\sigma) = \mathbf{MC}_{\mathcal{L}}(\sigma)$ for all sequences σ.

We use the abbreviation "SMC-optimal" for "strongly mind change optimal" (The terminology and intuition is similar to Kelly's in [50, 52]). A learner Ψ is simply SMC-optimal for $\mathcal{L}$ if Ψ is SMC-optimal given Λ.

Example 2.2.1.

1. In the COINIT problem, $\mathbf{MC}_{\mathcal{L}}(\Lambda) = \omega$, and $\mathbf{MC}_{\mathcal{L}}(\sigma) = \min(\mathrm{content}(\sigma))$ when $\mathrm{content}(\sigma) \neq \emptyset$. Since c_0 is a mind change counter for Ψ_{CO}, it follows that Ψ_{CO} is SMC-optimal. Any learner Ψ such that (1) $\Psi(\sigma) = \Psi_{\mathrm{CO}}(\sigma)$ if $\mathrm{content}(\sigma) \neq \emptyset$ and (2) $\Psi(\sigma) = \Psi(\sigma^-)$ if $\mathrm{content}(\sigma) = \emptyset$ is also SMC-optimal. (The initial conjecture $\Psi(\Lambda)$ is not constrained.)

2. The learner Ψ_{LIN} is SMC-optimal. Thus for the problem of inferring conservation laws, *SMC-optimality coincides with the inferences of the physics community.*

2.3 Summary and Discussion

We reviewed the basic definitions of Gold's language learning paradigm and mind change complexity of language collections. After a motivating example of the language collection COINIT, we formalized the properties of strongly mind change optimal learners: roughly, a learner is strongly mind change optimal if it realizes the best possible mind change bound not only in the entire learning problem, but also in subproblems that arise after observing some data. In the next chapter, we shall see a characterization of this optimality.

Chapter 3

Properties of Mind Change Efficient Learners

> The fear of infinity is a form of myopia that destroys the possibility of seeing the actual infinite, even though it in its highest form has created and sustains us, and in its secondary transfinite form occurs all around us and even inhabits our minds.
>
> Georg Cantor

In this chapter, we study a new inductive principle in Gold's model using point-set topology.

3.1 A Topological Characterization of Mind-Change Bounded Identifiability

Set-theoretical aspects of inductive inference have been studied by many learning theorists (e.g., [43] and [62]). As Jain et. al. observe [43, page 34]:

> Many results in the theory of inductive inference do not depend upon computability assumptions; rather, they are information theoretic in character. Consideration of noncomputable scientists thereby facilitates the analysis of proofs,

making it clearer which assumptions carry the burden.

Example 3.1.1. In [5], Angluin studied a series of conditions for language collections. A language collection $\{L_i\}$ satisfies Condition 1 if there exists an effective procedure M such that $M(i)$ is a finite subset of L_i and $\{L_j : M(i) \subseteq L_j\}$ contains no proper subset of L_i. Angluin showed that Condition 1 characterizes the indexed families of nonempty recursive languages inferable from positive data by computable learners [5, page 121]. A language collection $\{L_i\}$ satisfies Condition 2 if every language L_i has a finite subset T_i such that $\{L_j : T_i \subseteq L_j\}$ contains no proper subset of L_i. Condition 2 is the noneffective version of Condition 1, and hence is a necessary condition for inferability by computable learners.[1] Variants of Condition 2 turn out to be both sufficient and necessary for various models of language identifiability by noncomputable learners ([62, Chapter 2.2.2][43, Theorem.3.26]).

Information theoretic requirements such as Condition 2 constitute necessary conditions for computable learners, and are typically the easiest way to prove the unsolvability of some learning problems when they do apply. For example, Apsitis used the Baire topology on total recursive functions to show the difference between problem classes $\mathbf{EX}_\alpha$ and $\mathbf{EX}_{\alpha+1}$ (for the definition of these problem classes, see [8, Sect.3]). On the positive side, if a sufficient condition for noneffective learnability is met, it often yields insights that lead to the design of a successful learning algorithm.

It has often been observed that point-set topology, one of the most fundamental and well-studied mathematical subjects, provides useful concepts for describing the information theoretic structure of learning problems [73, Chapter 10], [64, 8, 49, 63]. In particular, Apsitis investigated the mind change complexity of function learning problems in terms of the Baire topology [8]. He showed that Cantor's 1883 notion of accumulation order in a topological space [15] defines a natural ordinal-valued measure of complexity for function learning problems, and that accumulation order provides a lower bound on the mind change complexity of a function learning problem. We generalize Apsitis' use of topology to apply it to language collections.

This chapter characterizes strongly mind change optimal (SMC-optimal)algorithms. We give specific examples of SMC-optimal algorithms in Chapter 4. The general form of the characterization is this. We assign a topological measure of complexity to each language

[1] Condition 2 characterizes BC-learnability for computable learners [10].

or hypothesis in a language collection. SMC-optimal algorithms are those that output the simplest languages where "simplicity" is defined by the topological measure.

The following section briefly reviews the relevant topological concepts.

3.1.1 Basic Definitions in Point-set Topology

A **topological space** over a set X is a pair $(X, \mathcal{O})$, where $\mathcal{O}$ is a collection of subsets of X, called **open sets**, such that $\emptyset$ and X are in $\mathcal{O}$ and $\mathcal{O}$ is closed under arbitrary union and finite intersection. One way to define a topology for a set is to find a base for it. A **base** $\mathcal{B}$ for X is a class of subsets of X such that

1. $\bigcup \mathcal{B} = X$, and
2. for every $x \in X$ and any $B_1, B_2 \in \mathcal{B}$ that contain x, there exists $B_3 \in \mathcal{B}$ such that $x \in B_3 \subseteq B_1 \cap B_2$.

For any base $\mathcal{B}$, the set $\{\bigcup \mathcal{C} : \mathcal{C} \subseteq \mathcal{B}\}$ is a topology for X [55, page 52]. That is, we may take an open set to be a union of sets in the base. Let $\mathcal{L}$ be a class of languages and $\sigma \in \text{SEQ}$. We use $\mathcal{L}|\sigma$ to denote all languages in $\mathcal{L}$ that are consistent with σ (i.e., $\{L \in \mathcal{L} : L \text{ is consistent with } \sigma\}$); similarly $\mathcal{L}|D$ denotes the languages in $\mathcal{L}$ that include a given finite subset D. The next proposition shows that $\mathcal{B}_{\mathcal{L}} = \{\mathcal{L}|\sigma : \sigma \in \text{SEQ}\}$ constitutes a base for $\mathcal{L}$.

Proposition 3.1.1. *$\mathcal{B}_{\mathcal{L}} = \{\mathcal{L}|\sigma : \sigma \in \text{SEQ}\}$ is a base for $\mathcal{L}$; hence $\mathcal{T}_{\mathcal{L}} = \{\bigcup \mathcal{S} : \mathcal{S} \subseteq \mathcal{B}_{\mathcal{L}}\}$ is a topology for $\mathcal{L}$.*

The topology $\mathcal{T}_{\mathcal{L}}$ generalizes the **positive information topology** from recursion theory [71, page 186] if we consider the graphs of functions as languages (as in [43, Chapter 3.9.2][62, Chapter 2.6.2]).

Examples. For the language collection COINIT we have that $\mathsf{COINIT}\,|\{2,3\} = \{L_0, L_1, L_2\}$ and $\mathsf{COINIT}\,|\{0\} = \{L_0\}$. The base $\mathcal{B}_{\mathsf{COINIT}}$ consists of all sets of the form $\mathsf{COINIT}\,|d$, where d is a finite subset of $\mathbb{N}$.

In a topological space $(X, \mathcal{T})$, a point x is an **isolated point** of a set $A \subseteq X$ if there is an open set $O \in \mathcal{T}$ such that $x \in O$ and $A \cap O \setminus \{x\} = \emptyset$. If x is not isolated point of $A \subseteq X$, then x is an **accumulation point of** A. Following Cantor [15], we define the **derived sets** using the concept of accumulation points.

Definition 3.1.1 (Cantor). Let $(X, \mathcal{T})$ be a topological space.

1. The **0-th derived set** of X, denoted by $X^{(0)}$, is just X.
2. For every successor ordinal α, the **α-th derived set** of X, denoted by $X^{(\alpha)}$, is the set of all accumulation points of $X^{(\alpha-1)}$.
3. For every limit ordinal α, the set $X^{(\alpha)}$ is the intersection of all β-th derived sets, where $\beta < \alpha$. That is, $X^{(\alpha)} = \bigcap_{\beta<\alpha} X^{(\beta)}$.

We give an example from the topology of the real plane that illustrates the geometrical intuitions behind the topological concepts.

Example. Let

$$A = \{(\frac{1}{n}, \frac{1}{m}) : n, m \in \mathbb{N}\} \cup \{(\frac{1}{n}, 0) : n \in \mathbb{N}\} \cup \{(0, \frac{1}{m}) : m \in \mathbb{N}\}$$

be a set of points in the real plane $\mathbb{R}^2$ with the standard topology. We use $\text{iso}(X)$ to denote all isolated points in X. Then $\text{iso}(A) = \{(\frac{1}{n}, \frac{1}{m}) : n, m \in \mathbb{N}\}$. Therefore

$$A^{(1)} = \{(\frac{1}{n}, 0) : n \in \mathbb{N}\} \cup \{(0, \frac{1}{m}) : m \in \mathbb{N}\}.$$

Similarly, we have $A^{(2)} = (0, 0)$, and $A^{(3)} = \emptyset$ (see Figure 3.1).

In the topology $\mathcal{T}_{\mathcal{L}}$, a language L is an isolated point of $\mathcal{L}$ if there is a finite subset $D \subseteq L$ such that the observation of D entails L (i.e., $\mathcal{L}|D = \{L\}$). The derived sets of $\mathcal{L}$ can be defined inductively as shown in Definition 3.1.1. Note if $\alpha < \beta$ then $\mathcal{L}^{(\alpha)} \supseteq \mathcal{L}^{(\beta)}$. It can be shown in set theory that there is an ordinal α such that $\mathcal{L}^{(\beta)} = \mathcal{L}^{(\alpha)}$, for all $\beta > \alpha$ [46]. In other words, there must be a fix-point for the derivation operation. If $\mathcal{L}$ has an empty fix-point, then we say $\mathcal{L}$ is **scattered** [55, page 78]. In a non-scattered space, the nonempty fixed point is called a **perfect kernel**.

The **accumulation order of a language** L in $\mathcal{L}$, denoted by $\text{acc}_{\mathcal{L}}(L)$ is the maximum ordinal α such that $L \in \mathcal{L}^{(\alpha)}$; when $\mathcal{L}$ is fixed by context, we simply write $\text{acc}(L) = \alpha$. The **accumulation order of a class of languages** $\mathcal{L}$, denoted by $\text{acc}(\mathcal{L})$, is the supremum of the accumulation order of all languages in it. Therefore a language collection has an accumulation order if and only if it is scattered.[2]

[2] Accumulation order is also called scattering height, derived length, Cantor-Bendixson rank, or Cantor-Bendixson length [46].

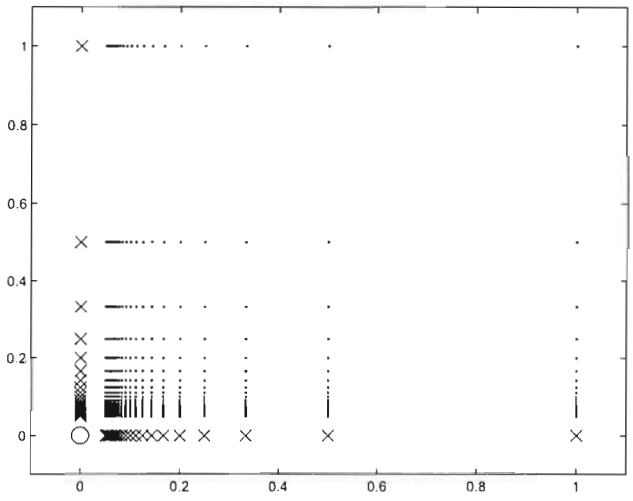


Figure 3.1: A set A on the real plane. Applying derivation once will remove the points marked with dots; applying derivation twice will remove the points marked with crosses; applying derivation again will remove the point marked with the circle.

Examples.

(1) The only isolated point in COINIT is $L_0 = \mathbb{N}$, for $\mathsf{COINIT}\,|\{0\} = \{L_0\}$. Therefore $\mathsf{COINIT}^{(1)} = \{L_i : i \geq 1\}$. Similarly L_1 is the only isolated point in $\mathsf{COINIT}^{(1)}$; hence $\mathsf{COINIT}^{(2)} = \{L_i : i \geq 2\}$. It is easy to verify that $\mathsf{COINIT}^{(n)} = \{L_i : i \geq n\}$. Therefore the accumulation order of language L_i in COINIT is i and the accumulation order of COINIT is $\omega = \sup \mathbb{N}$.

(2) In $\mathsf{LINEAR}_n = \{\text{linear subspaces of } \mathbb{Q}^n\}$, the only isolated point is $\mathbb{Q}^n$ itself: Let S be a set of n linearly independent points in $\mathbb{Q}^n$; then $\mathsf{LINEAR}_n\,|S = \{\mathbb{Q}^n\}$. Similarly every $(n-i)$-dimensional linear subspace of $\mathbb{Q}^n$ is an isolated point in $\mathsf{LINEAR}_n^{(i)}$. Therefore the accumulation order of LINEAR_n is n.

(3) In FIN, there is *no* isolated point. This is because for every finite subset S of $\mathbb{N}$, there are infinitely many languages in FIN that are consistent with S. Therefore FIN is a perfect kernel of itself and FIN has no accumulation order.

3.1.2 How Accumulation Order Characterizes Mind Change Complexity

In this section we show that the accumulation order of a language collection $\mathcal{L}$ is an exact measure of its mind change complexity for (not necessarily effective) learners: if $\text{acc}(\mathcal{L})$ is unbounded, then $\mathcal{L}$ is not identifiable with any ordinal mind change bound; and if $\text{acc}(\mathcal{L}) = \alpha$, then $\mathcal{L}$ is identifiable with a mind change bound.[3]

In a language topology, accumulation order has two fundamental properties that we apply often. Let $\text{acc}_{\mathcal{L}}(\sigma) \equiv \sup\{\text{acc}_{\mathcal{L}}(L) : L \in \mathcal{L}|\sigma\}$; as usual, we omit the subscript in context. A language L **tops** $\mathcal{L}|\sigma$ if $\text{acc}_{\mathcal{L}}(L) = \text{acc}_{\mathcal{L}}(\sigma)$; a sequence σ is topped if there is some language that tops σ. Note that if $\text{acc}_{\mathcal{L}}(\sigma)$ is a successor ordinal (e.g., finite), then σ is topped. All data sequences in SEQ(LINEAR_n) are topped. In COINIT, the initial sequence Λ is *not* topped. A language L **uniquely tops** σ in $\mathcal{L}$ if L is the only language that tops σ in $\mathcal{L}$.

Lemma 3.1.1. *Let $\mathcal{L}$ be a scattered class of languages with bounded accumulation order.*

1. *For every language $L \in \mathcal{L}$, for every text T for L, there exists a time n such that L uniquely tops $T[n]$ in $\mathcal{L}$; moreover, for every $m > n$, language L uniquely tops $T[m]$ in $\mathcal{L}$.*

2. *For any two languages $L_1, L_2 \in \mathcal{L}$ such that $L_1 \subset L_2$ it holds that $\text{acc}_{\mathcal{L}}(L_1) > \text{acc}_{\mathcal{L}}(L_2)$.*

Proof. Part 2 is immediate. Part 1: For contradiction, assume there is a text T for L such that for all n, $\mathcal{L}|(T[n])$ contains some language L' such that $\text{acc}(L') \geq \text{acc}(L) = \alpha$. Then L is an accumulation point of $\mathcal{L}^{(\alpha)}$, the subclass of $\mathcal{L}$ that contains all languages with accumulation order less than or equal to α. Therefore $\text{acc}(L) \geq \alpha + 1$, which is a contradiction. □

We now establish the correspondence between mind change complexity and accumulation order: $\mathbf{MC}_{\mathcal{L}}(\sigma) = \text{acc}_{\mathcal{L}}(\sigma)$.

Theorem 3.1.1. *Let $\mathcal{L}$ be a language collection and let σ be a finite data sequence. Then there is a learner Ψ that identifies $\mathcal{L}$ given σ with mind change bound $\alpha \iff \text{acc}_{\mathcal{L}}(\sigma) \leq \alpha$.*

[3] Necessary and sufficient conditions for finite mind change identifiability by learning *algorithms* appear in [56, 68].

Proof. ($\Leftarrow$) Define the mind change counter c by $c(\sigma) := \mathrm{acc}_{\mathcal{L}}(\sigma)$. We show that c is a mind change counter for the following learner Ψ that identifies $\mathcal{L}$:

1. $\Psi(\Lambda) :=?$,
2. $\Psi(\sigma) :=?$ if $\mathrm{acc}_{\mathcal{L}}(\sigma) < \mathrm{acc}_{\mathcal{L}}(\sigma^-)$,
3. $\Psi(\sigma) := L$ if $\mathrm{acc}_{\mathcal{L}}(\sigma) = \mathrm{acc}_{\mathcal{L}}(\sigma^-)$ and L uniquely tops σ in $\mathcal{L}$,
4. $\Psi(\sigma) := \Psi(\sigma^-)$ if $\mathrm{acc}_{\mathcal{L}}(\sigma) = \mathrm{acc}_{\mathcal{L}}(\sigma^-)$ and there is no language L that uniquely tops σ in $\mathcal{L}$.

It is easy to see that Ψ identifies $\mathcal{L}$. Let T be any text for any language $L \in \mathcal{L}$. Then by Lemma 3.1.1 there is a time n such that L uniquely tops $\mathcal{L}|T[n']$ for all $n' > n$. This is because if language L uniquely tops σ, then L uniquely tops any data sequence $\tau \supset \sigma$ if τ is consistent with L (i.e., $\mathrm{content}(\tau) \subseteq L$). Hence Clause 3 applies at all times $n' > n$, and Ψ converges to L on T, as required.

It remains to show that c is a mind change counter for Ψ. We begin with an auxiliary observation (a): For all languages $L \notin \mathcal{L}|\sigma$, if $\Psi(\sigma) \neq L$, then $\Psi(\tau) \neq L$ for every $\tau \in \mathrm{SEQ}(\mathcal{L})$ such that $\tau \supset \sigma$. In other words, if Ψ rejects a hypothesis L inconsistent with σ, then Ψ never returns to L after σ. To see that this holds, consider some $\tau \supset \sigma$ and suppose for reductio that $\Psi(\tau) = L$. Then there must be some ν with $\sigma \subset \nu \subseteq \tau$ such that $\Psi(\nu^-) \neq L$ and $\Psi(\nu) = L$. Then Clause 3 implies that L uniquely tops ν, which contradicts the assumption that L is inconsistent with σ and hence with ν.

We argue that (*) if Clause 3 applies at σ, then no mind change occurs at σ, such that either $\Psi(\sigma) = \Psi(\sigma^-)$ or $\Psi(\sigma^-) =?$. Suppose that $\Psi(\sigma^-) = L' \neq L$ and $\mathrm{acc}_{\mathcal{L}}(\sigma) = \mathrm{acc}_{\mathcal{L}}(\sigma^-)$ and L uniquely tops $\mathcal{L}|\sigma$. Let $n < |\sigma|$ be the least time such that $\Psi(\sigma[n]) = L'$. Then by definition of Ψ, Clause 3 applies at $\sigma[n]$, and and so L' uniquely tops $\sigma[n]$ in $\mathcal{L}$. Since L uniquely tops $\mathcal{L}|\sigma$ and $L \neq L'$, we know that $\mathrm{acc}_{\mathcal{L}}(\sigma) < \mathrm{acc}_{\mathcal{L}}(\sigma[n])$, and therefore $n < |\sigma^-|$ since $\mathrm{acc}_{\mathcal{L}}(\sigma) = \mathrm{acc}_{\mathcal{L}}(\sigma^-)$.

Thus $\mathrm{acc}_{\mathcal{L}}(\sigma^-) < \mathrm{acc}_{\mathcal{L}}(\sigma[n])$. Therefore by Clause 2, there is some time m such that $n < m < |\sigma|$ such that $\Psi(\sigma[m]) =?$, and moreover $L' \notin \mathcal{L}|\sigma[m]$. Therefore the observation (a) implies that $\Psi(\sigma^-) \neq L'$. This contradiction shows that either $\Psi(\sigma^-) =?$ or $\Psi(\sigma^-) = L$, and thus no mind change occurs at σ, as required.

It is immediate from the construction that Ψ changes its mind at σ only if Clauses 2 or 3 apply, so (*) implies that Ψ changes its mind only if Clause 2 applies. In that case

$c(\sigma) = \text{acc}_{\mathcal{L}}(\sigma) < \text{acc}_{\mathcal{L}}(\sigma^-) = c(\sigma^-)$. So counter c is a mind change counter for Ψ since this holds for all mind changes of Ψ.

($\Rightarrow$) Let Ψ be a learner that identifies $\mathcal{L}$ given σ. Suppose c is a mind change counter such that $c(\sigma) = \alpha$. We prove by transfinite induction that if $\text{acc}(\sigma) > \alpha$, then c is not a mind change counter for $\mathcal{L}$. Assume the claim holds for all $\beta < \alpha$ and consider α. Suppose $\text{acc}(\sigma) > \alpha$; then there is $L \in \mathcal{L}|\sigma$ such that $\text{acc}(L) = \alpha + 1$. Case 1: $\Psi(\sigma) = L$. Then since L is a limit point of $\mathcal{L}^{(\alpha)}$, there is L' in $\mathcal{L}^{(\alpha)}$ such that $L' \neq L$ and $\text{acc}(L') = \alpha$. Let $T' \supset \sigma$ be a text for L'. Since Ψ identifies L', there is a time $n > |\sigma|$ such that $\Psi(T'[n]) = L'$. Since $\Psi(T'[n]) \neq \Psi(\sigma)$ and $\Psi(\sigma) \neq ?$, this is a mind change of Ψ, hence $c(T'[n]) < c(\sigma)$. That is, $c(T'[n]) = \beta < \alpha$. On the other hand, since $\text{acc}(L') = \alpha$, we have $\text{acc}(T'[n]) > \beta$. By inductive hypothesis, c is not a mind change counter for Ψ. Case 2: $\Psi(\sigma) \neq L$. Let $T \supset \sigma$ be a text for L. Since Ψ identifies L, there is a time $n > |\sigma|$ such that $\Psi(T[n]) = L$. As $c(T[n]) \leq c(\sigma) = \alpha$ and $\text{acc}(T[n]) > \alpha$, as in Case 1, c is not a mind change counter for Ψ. □

Corollary 3.1.1. *Let $\mathcal{L}$ be a class of languages. Then there exists a mind-change bound for $\mathcal{L}$ if and only if $\mathcal{L}$ is scattered in the topology $\mathcal{T}_{\mathcal{L}}$.*

3.2 Necessary and Sufficient Conditions for Mind Change Optimal Learners

Theorem 3.1.1 establishes that if the accumulation order of a language collection $\mathcal{L}$ is bounded by an ordinal α, then there is a learner Ψ that identifies $\mathcal{L}$ with at most α mind changes; moreover, the proof of the theorem shows that there is a strongly mind-change optimal learner Ψ that does so.

The goal of this section is to characterize the behaviour of strongly mind-change optimal learners. These results allow us to design mind change optimal learners and to prove their optimality.

Proposition 3.2.1. *Let Ψ be a learner that identifies a language collection $\mathcal{L}$. Then Ψ is SMC-optimal for $\mathcal{L}$ if and only if for all data sequences σ:*

1. *If there is a language L topping σ in $\mathcal{L}$, and $\Psi(\sigma) \neq ?$, then $\Psi(\sigma)$ uniquely tops σ in $\mathcal{L}$.*

2. *If* $\sigma \neq \Lambda$ *is not topped and* $\mathrm{acc}_{\mathcal{L}}(\sigma) = acc_{\mathcal{L}}(\sigma^-)$, *then no mind change occurs at* σ.

Proof. ($\Rightarrow$) Clause 2 follows immediately from the fact that if Ψ is SMC-optimal, then $\mathrm{acc}_{\mathcal{L}}$ is a mind change counter for Ψ. For Clause 1, suppose σ is topped. Assume for contradiction that $\Psi(\sigma) = L' \neq ?$ and L' is not the only language topping $\mathcal{L}|\sigma$. Then there exists a language $L \in \mathcal{L}|\sigma$ such that $L \neq L'$ and $\mathrm{acc}_{\mathcal{L}}(L) = \mathrm{acc}_{\mathcal{L}}(\sigma)$. Let T be a text for L such that $T \supseteq \sigma$. If Ψ identifies $\mathcal{L}$, there exists a time $n > |\sigma|$ such that $\Psi(T[n]) = L$. Therefore Ψ makes at least one mind change between σ and $T[n]$. If $\mathrm{acc}_{\mathcal{L}}$ is a mind change counter for Ψ, then $\mathrm{acc}_{\mathcal{L}}(\sigma) > acc_{\mathcal{L}}(T[n])$. On the other hand, we have $\mathrm{acc}_{\mathcal{L}}(T[n]) = \mathrm{acc}_{\mathcal{L}}(L) = \mathrm{acc}_{\mathcal{L}}(\sigma)$. This contradiction shows that Ψ is not SMC-optimal.

($\Leftarrow$) We want to show that $\mathrm{acc}_{\mathcal{L}}$ is a mind change counter for Ψ.

Let σ be an arbitrary sequence in $\mathrm{SEQ}(\mathcal{L})$. There are four cases to consider:

1. σ is topped and $\mathrm{acc}(\sigma) < \mathrm{acc}(\sigma^-)$.

2. σ is topped and $\mathrm{acc}(\sigma) = \mathrm{acc}(\sigma^-)$.

3. σ is not topped and $\mathrm{acc}(\sigma) < \mathrm{acc}(\sigma^-)$.

4. σ is not topped and $\mathrm{acc}(\sigma) = \mathrm{acc}(\sigma^-)$.

We prove that $\Psi(\sigma) \neq \Psi(\sigma^-)$ and $\Psi(\sigma^-) \neq ?$ imply $\mathrm{acc}(\sigma) < \mathrm{acc}_{\mathcal{L}}(\sigma^-)$ in all four cases. That is, if a mind change occurs at σ, then the accumulation order drops at σ.

In cases 1 and 3, the implication holds trivially. In case 4, we have by Condition 2 of the proposition that there is no mind change at σ.

Case 2a: $\Psi(\sigma^-) = ?$; then there is no mind change at σ. Case 2b: $\Psi(\sigma^-) \neq ?$. We note that σ^- is topped since $\mathrm{acc}(\sigma) = \mathrm{acc}(\sigma^-)$ and σ is topped. So $\Psi(\sigma^-)$ has the highest accumulation order by Condition 1. Since $\Psi(\sigma^-)$ and $\Psi(\sigma)$ both have the highest accumulation order $\mathrm{acc}(\sigma)$, we have $\Psi(\sigma^-) = \Psi(\sigma)$. □

Proposition 3.2.1 shows that the key property of strongly mind change optimal learners is that when they output a consistent informative conjecture L different from ?, the conjecture L maximizes accumulation order. In many applications, hypotheses with higher accumulation order are intuitively simpler than those with lower accumulation order. In such language collections, we can think of mind change optimal methods as choosing the

simplest hypothesis consistent with the data when a unique simplest hypothesis is available.[4] In the following chapter, we shall see examples of SMC-optimal algorithms.

3.3 SMC-optimal Learning for Language Collections with Finite Thickness

It follows from Clause 2 of Lemma 3.1.1 that the accumulation order of a language L in a language collection $\mathcal{L}$ is at least as great as the length of a chain of supersets of L. We refer to this length as the inclusion depth of $\mathcal{L}$, as formalized the following definition.

Definition 3.3.1. Let $\mathcal{L}$ be a language collection and L be a language in $\mathcal{L}$. The **inclusion depth** of L in $\mathcal{L}$ is the size n of the largest index set $\{L_i\}_{1 \leq i \leq n}$ of distinct languages in $\mathcal{L}$, such that $L \subset L_1 \subset \cdots \subset L_i \subset \cdots \subset L_n$. The **inclusion depth** of $\mathcal{L}$ is the maximum of the inclusion depths of languages in $\mathcal{L}$ (see [59]).

For example, in COINIT, the inclusion depth of language $L_n = \{i \in \mathbb{N} : i \geq n\}$ is n. The inclusion depth of COINIT is ω. For many language collections, the inclusion depth of a language L is not only a lower bound on its accumulation order but characterizes it exactly. As we will show, examples include COINIT, LINEAR_n, and language collections P_1, T_n, and PATTERN defined in later chapters. The following proposition shows that a fairly simple property due to Angluin [5, Condition 3] is a sufficient condition for the accumulation order of a language to be equal to its inclusion depth. Following Angluin [5, Condition 3] and Shinohara [92], we say that a class of languages $\mathcal{L}$ has **finite thickness** if $\mathcal{L}|\{s\}$ is finite for every string $s \in \bigcup \mathcal{L}$. Note that if the language collection $\mathcal{L}$ has finite thickness, then every language in $\mathcal{L}$ has finite inclusion depth, so the inclusion depth of $\mathcal{L}$ is at most ω.

In language collections of finite thickness, the inclusion depth of a language is exactly its accumulation order.

Proposition 3.3.1. *Let $\mathcal{L}$ be a language collection with finite thickness and L be a language in $\mathcal{L}$.*

1. *There is a finite subset $S \subseteq L$ such that L is a $\subseteq$-minimum in $\mathcal{L}|S$.*

[4] We are indebted to S. Jain for suggesting this interpretation of Proposition 3.2.1. Kelly develops the idea of linking mind change efficient learning with simplicity of hypotheses, and presents it as a formalization of Occam's Razor [52][51].

2. *The inclusion depth of* L *is* $\mathrm{acc}_{\mathcal{L}}(L)$.

Proof. For clause 1, let s be a string in L; then $\mathcal{L}|\{s\}$ is finite since $\mathcal{L}$ has finite thickness. For every language $L' \in \mathcal{L}|\{s\}$ such that $L' \not\supseteq L$, the set $L \setminus L'$ is nonempty. For each L', choose a string $s_{L'}$ from $L \setminus L'$, and $S := \{s\} \cup \{s_{L'} : L' \in \mathcal{L}|\{s\} \setminus \{L\}\}$. Then $\mathcal{L}|S$ contains only languages that include L.

We prove clause 2 by induction.

Base case: Let L be a language with the inclusion depth 0, which means that there is no language that properly includes L. Then there exists a finite set $S \subseteq L$ such that $\mathcal{L}|S = \{L\}$. Therefore $\mathrm{acc}_{\mathcal{L}}(L) = 0$ by the definition of accumulation order.

inductive step: Assume for every language with inclusion depth less than k that its accumulation order equals its inclusion depth. Consider the case that L has inclusion depth k. From the induction assumption, we know that there exists a language L' such that $L \subset L'$ and $\mathrm{acc}_{\mathcal{L}}(L') = k - 1$. Therefore (1) $\mathrm{acc}_{\mathcal{L}}(L) \geq k$ by Clause 2 of Lemma 3.1.1. On the other hand, since $\mathcal{L}$ has finite thickness, there exists a subset $S \subseteq L$ such that $\mathcal{L}|S$ contains only languages that include L. It is clear that for every language $L' \in \mathcal{L}|S$, if $L' \neq L$ then $L' \supset L$; this implies that L' has inclusion depth less than k for every language $L' \in \mathcal{L}|S - L$, otherwise L would have inclusion depth greater than k. Therefore $\mathrm{acc}((\mathcal{L}|S) \setminus \{L\}) = \sup(\mathrm{acc}_{\mathcal{L}}\{L' \in \mathcal{L}|S \setminus \{L\}\}) < k$; thus (2) $\mathrm{acc}_{\mathcal{L}}(L) \leq k$. Combining the two inequalities (1) and (2), we have $\mathrm{acc}_{\mathcal{L}}(L) = k$, which complete the inductive step. □

As it is easy to verify that each of the language collections COINIT, P_1, T_n, and PATTERN has finite thickness, the proposition implies that the accumulation order of each language L in these collections is the inclusion depth of L, or the maximum length of a chain of supersets of L. Clause 1 of the proposition establishes the following corollary.

Corollary 3.3.1. *Let* Ψ *be a learner that identifies a language collection* $\mathcal{L}$ *with finite inclusion depth. Then* Ψ *is SMC-optimal for* $\mathcal{L}$ *if and only if for all data sequences* σ*: if* $\Psi(\sigma) \neq ?$*, then* $\Psi(\sigma)$ *is the unique language with the largest inclusion depth for* σ.

Among all SMC-optimal learners, there are ones that produce a guess faster; in fact, using Gold's notion of "uniformly faster", we can show that for a language collection with finite inclusion depth there is a unique fastest SMC-optimal learner. Gold proposed the following way to compare the convergence speed of two learners [37, page 462] (we follow the notation of [73, Definition 8.1.C]).

Definition 3.3.2. Let $\mathcal{L}$ be a language collection.

1. The convergence time of a learner Ψ on text T is defined as $\text{CONV}(\Psi, T) \equiv$ the least time m such that for all $m' \geq m$ we have $\Psi(T[m]) = \Psi(T[m'])$.

2. A learner Ψ identifies $\mathcal{L}$ uniformly faster than learner $\Phi \iff$

   (a) for all languages $L \in \mathcal{L}$ and all texts T for L, we have $\text{CONV}(\Psi, T) \leq \text{CONV}(\Phi, T)$, and

   (b) for some language $L \in \mathcal{L}$ and some text T for L, we have $\text{CONV}(\Psi, T) < \text{CONV}(\Phi, T)$.

For a language collection $\mathcal{L}$ with finite inclusion depth, Proposition 3.1.1 implies that if there is no language L that uniquely maximizes inclusion depth given σ, then a learner that is SMC-optimal outputs ? on σ. Intuitively, the fastest SMC-optimal learner procrastinates with making a conjecture no longer than is necessary to meet this condition. Formally, this learner is defined as follows for all sequences $\sigma \in \text{SEQ}(\mathcal{L})$:

$$\Psi^{\mathcal{L}}_{\text{fast}}(\sigma) = \begin{cases} ? & \text{if no language uniquely maximizes inclusion depth given } \sigma \\ L & \text{where } L \in \mathcal{L} \text{ uniquely maximizes inclusion depth given } \sigma. \end{cases}$$

The next observation shows that $\Psi^{\mathcal{L}}_{\text{fast}}$ identifies $\mathcal{L}$ faster than any other SMC-optimal method.

Observation 1. *Let $\mathcal{L}$ be a language collection with finite inclusion depth. Then $\Psi^{\mathcal{L}}_{\text{fast}}$ is SMC-optimal and identifies $\mathcal{L}$ uniformly faster than any other SMC-optimal learner for $\mathcal{L}$.*

Proof. It is easy to see that $\Psi^{\mathcal{L}}_{\text{fast}}$ identifies $\mathcal{L}$, so Proposition 3.1.1 implies that $\Psi^{\mathcal{L}}_{\text{fast}}$ is SMC-optimal. Let $\Psi \neq \Psi^{\mathcal{L}}_{\text{fast}}$ be any other SMC-optimal learner that identifies $\mathcal{L}$. Again by Proposition 3.1.1, if $\Psi(\sigma) \neq ?$, then $\Psi(\sigma) = \Psi^{\mathcal{L}}_{\text{fast}}$. Therefore for any language $L \in \mathcal{L}$ and text T for L, we have $\text{CONV}(\Psi^{\mathcal{L}}_{\text{fast}}, T) \leq \text{CONV}(\Psi, T)$. Now let $\sigma \in \text{SEQ}(\mathcal{L})$ be a data sequence such that $\Psi(\sigma) \neq \Psi^{\mathcal{L}}_{\text{fast}}(\sigma)$. Since both Ψ and $\Psi^{\mathcal{L}}_{\text{fast}}$ are SMC-optimal, Proposition 3.1.1 implies that there is a language $L \in \mathcal{L}$ that uniquely maximizes inclusion depth given σ. So $\Psi^{\mathcal{L}}_{\text{fast}}(\sigma) = L$ and $\Psi(\sigma) = ?$. Now let $T \supset \sigma$ be any text for L extending σ. It is easy to see that L remains the language that uniquely maximizes inclusion depth on any data sequence σ' with $\sigma \subseteq \sigma' \subset T$. So $\text{CONV}(\Psi^{\mathcal{L}}_{\text{fast}}, T) \leq |\sigma|$. On the other hand, clearly

$\mathrm{CONV}(\Psi, T) > |\sigma|$ since $\Psi(\sigma) = ?$. Thus $\Psi^{\mathcal{L}}_{\mathrm{fast}}$ identifies $\mathcal{L}$ uniformly faster than Ψ and in general faster than any other SMC-optimal learner. □

3.4 Gold Learning Paradigm with Set Learners

In Chapter 2, we define a learner to be a function that outputs either a single language or the question mark "?". Alternatively, we can define a learner to output a set of languages.

Definition 3.4.1 (Set Learner)**.** A **set learner** for a language collection $\mathcal{L}$ is a mapping of SEQ into $2^{\mathcal{L}}$.

For convenience of exposition, we shall call a learner that outputs a single language or the question mark a **regular learner**. Compared to a regular learner, a set learner takes a different approach when facing model uncertainty. One can argue that a set of languages is a more informative output than the question mark.

A set learner Ψ **identifies** a language L in a language collection $\mathcal{L}$ if it cofinitely outputs $\{L\}$, and Ψ identifies $\mathcal{L}$ if it identifies every $L \in \mathcal{L}$. A set learner Ψ **changes its mind** at some nonempty finite sequence $\sigma \in \mathrm{SEQ}$ if $\Psi(\sigma) \not\subseteq \Psi(\sigma^-)$. In other words, a set learner changes its mind if it changes its conjecture, unless that conjecture is just a strengthening of the previous one. With the modified definition of mind change, we can consider the mind change complexity and the mind change optimality of set learners.

Using the technique in Theorem 3.1.1, we can easily prove the following lemma.

Lemma 3.4.1. *Accumulation order is a lower bound of the mind change complexity for a set learner.*

Proposition 3.4.1. *Let $\mathcal{L}$ be a language collection of finite accumulation order. Then $\mathcal{L}$ is identifiable by a set learner in n mind changes if and only if* $\mathrm{acc}(\mathcal{L}) = n$.

Proof. Let Ψ be a SMC-optimal regular learner for $\mathcal{L}$. We construct a set learner Ψ' and prove that it not only realizes the same mind change bound, but also is SMC-optimal.

The learner Ψ' is constructed as follows.

1. $\Psi'(\sigma) = \Psi(\sigma)$ if $\Psi(\sigma) \neq ?$;
2. $\Psi'(\sigma)$ contains the set of languages with the highest accumulation order if $\Psi(\sigma) = ?$.

Since $\mathcal{L}$ has finite accumulation order, the set of languages with the highest accumulation order is always defined.

Clearly the Ψ' identifies $\mathcal{L}$ as Ψ does. Moreover Ψ' changes its mind only if the accumulation order drops, this is exactly when Ψ changes its mind.

Hence we proved that $acc(\mathcal{L})$ is an upper bound on the mind change complexity for a set learner. Combined with Lemma 3.4.1, we proved the proposition. □

In the previous proof, we see a natural SMC-optimal set-learner for language collections of finite accumulation order. In chapter 6 and 7, we shall consider approximation to this SMC-optimal set-learner. For now, we will turn back to study regular learners.

3.5 Accumulation Order and Structural Complexity

Our final section relates accumulation order to other well-known learning-theoretic concepts that describe the structure of a learning problem.

3.5.1 Elasticity

We show that the concept of elasticity provides a sufficient condition for a language collection $\mathcal{L}$ to have a bounded accumulation order, which by Theorem 3.1.1 implies that $\mathcal{L}$ is identifiable with a bounded number of mind changes.

A class of languages $\mathcal{L}$ has **infinite elasticity** if there exist an infinite sequence of strings $(s_i)_{i\in\mathbb{N}}$ and a infinite sequence of languages $(L_i)_{i\in\mathbb{N}}$, where $L_i \in \mathcal{L}$, such that for each $i \in \mathbb{N}$, $\{s_0, \ldots, s_i\} \subseteq L_i$ but $s_{i+1} \notin L_i$. A class of languages has **finite elasticity** if it does not have infinite elasticity [105][66, Definition 7]. For example the language collection LINEAR_n has finite elasticity because if vector $\vec{v}_{i+1}$ is not in a linear subspace L_i, then $\vec{v}_{i+1}$ is independent of any subset $\{\vec{v}_0, \vec{v}_1, \ldots, \vec{v}_i\}$. It is not hard to see that finite thickness implies finite elasticity [105], so T_n and P_1 have finite elasticity.

We use $D(\mathcal{L})$ to denote all finite subsets of languages in $\mathcal{L}$. A subset $\mathcal{P}$ of a topological space is **perfect** if $\mathcal{P}$ has no isolated points [55, page 78].

Lemma 3.5.1. *Let $\mathcal{P}$ be a perfect nonempty set of languages. Then $\mathcal{P}|d$ is also nonempty and perfect for every finite subset $d \in D(\mathcal{P})$.*

To illustrate, the language collection FIN which comprises all finite subsets of $\mathbb{N}$ is perfect and nonempty. Since no finite subset d entails a single language in FIN (i.e., $\mathrm{card}(\mathsf{FIN}\,|d) > 1$), we have that $\mathsf{FIN}\,|d$ is nonempty and perfect.

The next proposition gives a topological condition sufficient to establish that a language collection $\mathcal{L}$ has infinite elasticity, namely that $\mathcal{L}$ contain a subset that is perfect in the language topology for $\mathcal{L}$. If $\mathcal{L}$ has a perfect subset, the derivation procedure from Section 3.1.1 terminates with a nonempty perfect kernel, and $\mathcal{L}$ has no bounded accumulation order, which by Theorem 3.1.1 is equivalent to the statement that $\mathcal{L}$ is not identifiable with an ordinal mind change bound. Contrapositively, if $\mathcal{L}$ has finite elasticity, then $\mathcal{L}$ is identifiable with an ordinal mind change bound.

Proposition 3.5.1. *Let $\mathcal{L}$ be a collection of languages.*

1. *If $\mathcal{L}$ contains a nonempty perfect subset $\mathcal{P} \subseteq \mathcal{L}$, then $\mathcal{L}$ has infinite elasticity.*
2. *If $\mathcal{L}$ has finite elasticity, then $\mathcal{L}$ has a bounded accumulation order and hence $\mathcal{L}$ is identifiable with a bounded number of mind changes.*

Proof. Part 1: If $\mathcal{P} \neq \emptyset$ is perfect, then $\mathcal{P}$ is infinite and so there are infinitely many languages $L \in \mathcal{P}$ such that $L \neq \bigcup \mathcal{P}$. Choose a nonempty language $L_0 \neq \bigcup \mathcal{P}$ and strings $s_0 \in L_0$ and $s_1 \in \bigcup \mathcal{P} - L_0$. Let $\mathcal{P}_1 := \mathcal{P}|\{s_0, s_1\}$. Then by Lemma 3.5.1, $\mathcal{P}_1$ is a nonempty perfect set. So there is nonempty $L_1 \in \mathcal{P}_1$ such that $L_1 \neq \bigcup \mathcal{P}_1$, and we may choose a string $s_2 \in \bigcup \mathcal{P}_1 - L_1$. Continuing this process indefinitely, we obtain two sequences $(L_i)_{i\in\mathbb{N}}$ and $(s_i)_{i\in\mathbb{N}}$ such that for each $i \in \mathbb{N}$, $\{s_0, \ldots, s_i\} \subseteq L_i$ but $s_{i+1} \notin L_i$. In other words, $\mathcal{L}$ has infinite elasticity.

Part 2: Suppose that $\mathcal{L}$ has finite elasticity. Then by the contrapositive of Clause 1, the only perfect subset of $\mathcal{L}$ is the empty set. Since the derivation procedure from Definition 3.1.1 terminates with a perfect subset of $\mathcal{L}$, it thus terminates with the empty set, so $\mathcal{L}$ is scattered and has bounded accumulation order by Corollary 3.1.1. □

The proposition implies that LINEAR_n and all sub-collections of PATTERN are identifiable with a bounded number of mind changes.

If a language has infinite elasticity, then it also has infinite thickness. It is known that, for indexed language families, finite elasticity is a sufficient condition for effective learnability [105, 66]. A sequence of nonempty languages $\{L_i\}$ constitutes an indexed family just in case

there exists a computable function f such that for each $i \in N$ and for each $x \in N$, we have $f(i,x) = 1$ if $x \in L_i$ and $f(i,x) = 0$ otherwise [5, Sec. 2][43, Ex. 4.7]. Figure 3.2 illustrates the relationship among these structural concepts.

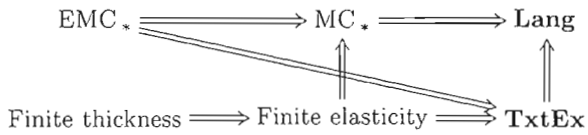


Figure 3.2: Relations between various computable and noncomputable identifiability concepts. $\mathbf{EMC}_*$ denotes language collections identifiable by a computable learner with a bounded number of mind changes. $\mathbf{MC}_*$ denotes language collections with bounded accumulation orders, or equivalently, identifiable by a noncomputable learner with a bounded number of mind changes. Following [43], we use **Lang** to denote all language collections identifiable by noncomputable learners and use **TxtEx** to denote all language collections identifiable by computable learners. The notation $\Rightarrow_{+\text{ indexed family}}$ indicates that the implication holds only for indexed language collections.

3.5.2 Intrinsic Complexity

Next we consider the relationship between weak and strong reducibility, intrinsic complexity [31, 44], and accumulation order. Our basic result is that if language collection $\mathcal{L}_1$ is reducible to $\mathcal{L}_2$, then $\text{acc}(\mathcal{L}_2) \geq \text{acc}(\mathcal{L}_1)$. In this sense reducibility agrees with accumulation order—and hence mind change complexity—as a comparison of the complexity of different learning problems.

Definition 3.5.1 ([42, 44, 31]).

1. An **enumeration operator** is a computable function that maps SEQ into SEQ.

2. An infinite sequence G is **admissible** for a text T if G converges to an index (or grammar) of the language $L = \text{content}(T)$.

3. Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two classes of languages. Then $\mathcal{L}_1$ is **weakly reducible to** $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \leq_{weak} \mathcal{L}_2$, if there exist two enumeration operators Θ and Ξ such that for every text T_1 for $\mathcal{L}_1$,

   (a) $\Theta(T_1) = \bigcup_n \Theta(T_1[n])$ is a text for $\mathcal{L}_2$.

(b) for every admissible sequence G for $\Theta(T_1)$, the sequence $\Xi(G) = \bigcup_n \Xi(G[n])$ is admissible for T_1.

We say that operators Θ and Ξ **witness** $\mathcal{L}_1 \leq_{weak} \mathcal{L}_2$.

4. Language collection $\mathcal{L}_1$ is **strongly reducible to** $\mathcal{L}_2$, denoted by $\mathcal{L}_1 \leq_{strong} \mathcal{L}_\epsilon$, if there exists Θ and Ξ such that

   (a) Θ and Ξ witness $\mathcal{L}_1 \leq_{weak} \mathcal{L}_2$, and

   (b) for every language $L_1 \in \mathcal{L}_1$, there exists a language $L_2 \in \mathcal{L}_2$ such that $L_2 = \text{content}(\Theta(T))$ for every text T for L_1.

The following proposition relates accumulation order to reducibility.

Proposition 3.5.2. *Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two language collections such that $\mathcal{L}_1 \leq_{weak} \mathcal{L}_2$ is witnessed by operators Θ and Ξ.*

1. *Let L and L' be two distinct languages in $\mathcal{L}_1$, and let T and T' be texts for L and L', respectively. Then* $\text{content}(\Theta(T)) \neq \text{content}(\Theta(T'))$. *(Thus, texts from distinct languages are mapped to texts from distinct languages.)*

2. *Let T be a text for some $L_1 \in \mathcal{L}_1$, and let $L_2 = \text{content}(\Theta(T))$. Then* $\text{acc}_{\mathcal{L}_2}(L_2) \geq \text{acc}_{\mathcal{L}_1}(L_1)$.

3. *If $\mathcal{L}_1 \leq_{weak} \mathcal{L}_2$, then* $\text{acc}(\mathcal{L}_1) \leq \text{acc}(\mathcal{L}_2)$. *Therefore if $\mathcal{L}_2$ is identifiable with mind change bound α, so is $\mathcal{L}_1$.*

Proof. Clause 1: For contradiction, assume $\text{content}(\Theta(T)) = \text{content}(\Theta(T')) = L_2 \in \mathcal{L}_2$. If G is an admissible sequence for $\Theta(T)$, then G is also an admissible sequence for $\Theta(T')$. Therefore $\Xi(G)$ is admissible for both T and T', which is impossible.

Clause 2: The proof is by transfinite induction on $\text{acc}_{\mathcal{L}_2}(L_2)$. Assume the claim hold for all cases where $\text{acc}_{\mathcal{L}_2}(L_2) = \beta < \alpha$, and suppose that $\text{acc}_{\mathcal{L}_2}(L_2) = \alpha$.

For contradiction, assume that $\text{acc}_{\mathcal{L}_1}(L_1) = \gamma > \alpha$. Since $\Theta(T)$ is a text for L_2, by Lemma 3.1.1, there exists a time n such that L_2 uniquely has the highest accumulation order α in $\mathcal{L}_2|\Theta(T)[n]$. Let m be a time such that (a) $\Theta(T[m]) \supseteq \Theta(T)[n]$. Since T is a text for L_1 and $\text{acc}_{\mathcal{L}_1}(L) > \alpha$, there is a language $L_1' \in \mathcal{L}_1|T[m]$ such that $\text{acc}_{\mathcal{L}_1}(L_1') = \alpha$. Let T' be a text for L_1' that extends $T[m]$; then by Clause 1 we have that $\text{content}(\Theta(T')) \neq L_2$.

Let us write L_2' for content($\Theta(T')$). Clearly content($\Theta(T'[m])$) $\subseteq$ content($\Theta(T')$) $= L_2'$, so $L_2' \in \mathcal{L}_2|\Theta(T'[m])$. Since $T[m] = T'[m]$ we have (b) $L_2' \in \mathcal{L}_2|\Theta(T[m])$. Combining (a) and (b) we have (c) $L_2' \in \mathcal{L}_2|\Theta(T)[n]$.

Since L_2 is the only language in $\mathcal{L}_2|\Theta(T)[n]$ with the accumulation order α, the language L_2' must has an accumulation order $\beta < \alpha$ in $\mathcal{L}_2$. Therefore, $\text{acc}_{\mathcal{L}_1}(L_1') = \alpha > \text{acc}_{\mathcal{L}_2}(L_2') = \beta$, which contradicts the induction hypothesis and establishes the inductive step.

Clause 3: Immediate consequence of Clause 2. □

The above proposition gives us a necessary condition for reducibility, which we illustrate in the following examples. As in [44], SINGLE denotes the class of all singleton languages. It is easy to see that acc(COINIT) $= \omega$ but acc(SINGLE) $= 0$, therefore COINIT $\not\leq_{weak}$ SINGLE, as shown in [44].

If $\mathcal{L}_1$ is not scattered (i.e., has no mind change bound) and $\mathcal{L}_2$ is scattered (i.e., has a mind change bound), then Proposition 3.5.2 implies that $\mathcal{L}_1$ is not weakly reducible to $\mathcal{L}_2$. Since the class of all finite languages FIN is not scattered (cf. Section 3.1.1), it follows that FIN $\not\leq_{weak}$ COINIT, as established by [44].

If Θ and Ξ witness $\mathcal{L}_1 \leq_{strong} \mathcal{L}_2$, then Θ induces a function f_Θ that maps $\mathcal{L}_1$ into $\mathcal{L}_2$ as follows: for a language $L \in \mathcal{L}_1$, choose any text T for L, and assign $f(L) =$ content($\Theta(T)$). The definition of strong reducibility guarantees that f_Θ is well-defined. We show that f_θ is a continuous one-one function in our topology. A function $f : X \to Y$ is **continuous** if for every point $x \in X$ and every neighbourhood V of $f(x)$ in Y, there exists a neighborhood U of x in X, such that $f(U) \subseteq V$.[5] For two language collections $\mathcal{L}_1$ and $\mathcal{L}_2$, this means that $f : \mathcal{L}_1 \to \mathcal{L}_2$ is continuous if for every language $L_1 \in \mathcal{L}_1$ and every finite subset $D_2 \subseteq f(L_1)$, there is a finite subset $D_1 \subseteq L_1$ such that $\{f(L) : L \in \mathcal{L}_1|D_1\} \subseteq \mathcal{L}_2|D_2$.

Lemma 3.5.2. *Suppose Θ and Ξ witness $\mathcal{L}_1 \leq_{strong} \mathcal{L}_2$. Then f_Θ defined above is a continuous one-one function.*

The proof is left to the reader.

Lemma 3.5.2 connects strong reducibility with many basic results in point-set topology. As an illustration, we apply standard theorems in topology to immediately derive that strong reducibility respects accumulation order without the need for the construction of Proposition 3.5.2.

[5] This definition is equivalent to the condition that $f^{-1}(V)$ is open in X for every open set V of Y.

Proposition 3.5.3. *Let $f : X \mapsto Y$ be a continuous one-one function, and let $A \subseteq X$ and $x \in X$.*

1. *If $x \in A^{(1)}$, then $f(x) \in f(A)^{(1)}$ (i.e., $f(A^{(1)}) \subseteq [f(A)]^{(1)}$).*

2. *If $\mathrm{acc}(Y)$ is defined, then $\mathrm{acc}(X)$ is also defined and moreover $\mathrm{acc}(X) \leq \mathrm{acc}(Y)$.*

Proof. Clause 1 is Theorem 2.3 of [11]. Clause 2 follows easily by transfinite induction. □

Therefore we can establish the following result from standard topological results.

Corollary 3.5.1. *Suppose Θ and Ξ witness $\mathcal{L}_1 \leq_{strong} \mathcal{L}_2$. Then $\mathrm{acc}(\mathcal{L}_1) \leq \mathrm{acc}(\mathcal{L}_2)$.*

Thus if $f_\Theta : \mathcal{L}_1 \to \mathcal{L}_2$ is onto and $(f_\Theta)^{-1} : \mathcal{L}_2 \to \mathcal{L}_1$ is continuous, then $\mathrm{acc}(\mathcal{L}_1) = \mathrm{acc}(\mathcal{L}_2)$; in topological terminology, homeomorphic language collections have the same accumulation order.

3.6 Summary

We applied the classic topological concept of accumulation order to characterize the mind change complexity of a learning problem: A language collection $\mathcal{L}$ is identifiable by a learner (not necessarily computable) with α mind changes if and only if the accumulation order of $\mathcal{L}$ is at most α. This reveals the characteristic property of strongly mind change optimal learners: They output languages with maximal accumulation order. Thus analyzing the accumulation order of a learning problem is a powerful guide to constructing mind change efficient learners. We illustrated these results in the learning problem of identifying a linear space. For learning linear subspaces, the natural method of conjecturing the least subspace containing the data is the only mind change optimal learner that does not "procrastinate" (i.e., never outputs ? or an inconsistent conjecture). This procedure reproduces exactly the inference procedure that the particle physics community has followed to arrive at the set of conservation laws found in the current standard model of particle physics. In general, mind change optimal learners maximize the simplicity of the output language.

Chapter 4

Mind Change Optimal Learning of Patterns

> A mathematician, like a painter or poet, is a maker of patterns.
>
> G. H. Hardy

In this chapter we consider further computational issues and illustrate how our analysis of mind change complexity can aid the design of mind change efficient learning algorithms in specific problems. As it turns out, Angluin's well-known pattern languages bring out a number of general points about constructing SMC-optimal learning algorithms.

It is straightforward to computationally implement the learners Ψ_{CO} and Ψ_{LIN}. These learners have the feature that whenever they produce a conjecture L on data σ, the language L is a subset of every other languages in $\mathcal{L}|\sigma$. Formally, we say L is the $\subseteq$**-minimum** at σ if L is a subset of every other language in $\mathcal{L}|\sigma$. It follows from Clause 2 of Lemma 3.1.1 that a $\subseteq$-minimum also maximizes accumulation order, so Ψ_{CO} and Ψ_{LIN} always output the language uniquely having the highest accumulation order and hence by Proposition 3.2.1 they are both SMC-optimal. For a language collection $\mathcal{L}$ like COINIT and LINEAR, if we can compute the $\subseteq$-minimum, an SMC-optimal learning algorithm for $\mathcal{L}$ can be constructed on the model of Ψ_{CO} and Ψ_{LIN}. However, these conditions are much stronger than necessary in general. In general, it suffices that we can *eventually* compute a $\subseteq$-minimum along any text. In particular, we can make a learner output ? when it is computationally impossible or too complex to find an $\subseteq$-minimum consistent language. We illustrate this point by specifying

SMC-optimal learning algorithms for P_1 and T_n, two subclasses of languages defined by Angluin's well-known patterns [4, p.48].

4.1 Patterns

Let X be a set of variable symbols (e.g., $x_1, x_2, \ldots$) and let Σ be a finite alphabet of at least two constant symbols (e.g., $0, 1, \ldots, n$). A **pattern**, denoted by p, q etc., is a finite non-null sequence over $X \cup \Sigma$ (e.g., $\mathtt{x_1 0 x_1}$ or $\mathtt{x_1 x_2 x_2}$). We use $var(p)$ to denote the set of distinct variables in p and use $\#var(p)$ to denote the number of distinct variables in p. A pattern p is **canonical** if $var(p) = \{x_1, x_2, \ldots, x_{var(p)}\}$ and their first occurrence (from left to right) is in that order. For example, the pattern $\mathtt{x_1 2 x_2 x_1}$ is canonical, but patterns $\mathtt{x_2 1 x_4}$ and $\mathtt{x_2 x_1}$ are not. We use PATTERN to denote the set of all canonical patterns. A **substitution** θ replaces a variable in a pattern p by another pattern uniformly. For example, $\theta = [\mathtt{x_2 x_3}/x_1]$ maps the pattern $\mathtt{x_1 x_1}$ to the pattern $\mathtt{x_2 x_3 x_2 x_3}$. Substitutions give rise to a partial order over all patterns. Following [79, 80], we say that a pattern q **subsumes** a pattern p, denoted by $p \preceq q$, if there is a substitution θ such that $p = q\theta$. The **language generated by a pattern** p, denoted by $L(p)$, is the set $\{q \in \Sigma^* : q \preceq p\}$. The **length** of a pattern p, denoted by $|p|$, is the number of symbols occurred in p. The set of strings of the same length as a given pattern p plays an important role in the proofs below; we denote this set by $S(p) \equiv \{s \in L(p) : |s| = |p|\}$. We observe that for an alphabet Σ, the size of $S(p)$ is given by $|S(p)| = |\Sigma|^{\#var(p)}$.

To discuss effective learning we have to take care of some technicalities. First, the output of a learning algorithm are descriptions of languages instead of languages themselves. Therefore, we extend our notation in Section 2.1 by replacing languages and language collections by language descriptions and classes of language descriptions. For example, in pattern identification problem, we use PATTERN to denote both the class of all canonical patterns and the language collection it generates; we use $\mathrm{acc}_{\mathrm{PATTERN}}(p)$ to denote the accumulation order of $L(p)$ in the language collection denoted by PATTERN. As another example, we use PATTERN$\,|S$ to denote both languages consistent with the evidence set S and the patterns that generate them. It should be clear from the context whether we are referring to a language or its description by a pattern that generates the language.

4.2 Mind Change Optimal Identification of One-Variable Patterns

If a pattern contains exactly one distinct variable (i.e., $\#var(p) = 1$), then it is a **one-variable pattern**. For one-variable patterns, we usually omit the subscript for the variable (e.g., x01 or 0x00x1). Following [4], we denote the set of all one-variable patterns by P_1. Angluin described an algorithm that, given a finite set S of strings as input, finds the set of one-variable patterns descriptive of S, and then (arbitrarily) selects one with the maximum length [4, Th.6.5]. A one-variable pattern p is **descriptive of a sample** S if $S \subseteq L(p)$ and for every one-variable pattern q such that $S \subseteq L(q)$, the language $L(q)$ is not a proper subset of $L(p)$ [4, p.48]. To illustrate, the pattern 1x is descriptive of the samples $\{10\}$ and $\{10, 11\}$, the pattern x0 is descriptive of the samples $\{10\}$ and $\{10, 00\}$, and the pattern x is descriptive of the sample $\{10, 00, 11\}$.

We give an example (summarized in Figure 4.1) to show that Angluin's algorithm is not an SMC-optimal learner. Let x be the target pattern and consider a text $T = \langle 10, 00, 11, 0, \ldots \rangle$ for $L(\mathtt{x})$. As mentioned above, we write $P_1|S$ for the set of one-variable patterns consistent with a sample S. Then $P_1|\{10\} = \{1\mathtt{x}, \mathtt{x}0, \mathtt{x}\}$, $P_1|\{10, 00\} = \{\mathtt{x}0, \mathtt{x}\}$, $P_1|\{10, 11\} = \{1\mathtt{x}, \mathtt{x}\}$ and $P_1|\{10, 00, 11\} = \{\mathtt{x}\}$. The accumulation orders of these languages are determined as follows:

1. $\mathrm{acc}_{P_1}(L(\mathtt{x})) = 0$ since $L(\mathtt{x})$ is isolated; so $\mathrm{acc}_{P_1}(\langle 10, 00, 11\rangle) = 0$.
2. $\mathrm{acc}_{P_1}(L(1\mathtt{x})) = 1$ since $P_1|\{10, 11\} = \{1\mathtt{x}, \mathtt{x}\}$; so $\mathrm{acc}_{P_1}(\langle 10, 11\rangle) = 1$.
3. $\mathrm{acc}_{P_1}(L(\mathtt{x}0)) = 1$ since $P_1|\{10, 00\} = \{\mathtt{x}0, \mathtt{x}\}$; so $\mathrm{acc}_{P_1}(\langle 10, 00\rangle) = 1$.

Also, we have $\mathrm{acc}_{P_1}(\langle 10\rangle) = 1$. Since for $T[1] = \langle 10\rangle$, the one-variable patterns 1x and x0 are both descriptive of $\{10\}$, an Angluin-style learner M_A conjectures either 1x or x0; suppose $M_A(\langle 10\rangle) = 1\mathtt{x}$. Now let c_A be any mind change counter for M_A. Since 1x is consistent with $\langle 10\rangle$, SMC-optimality requires that $c_A(\langle 10\rangle) = \mathrm{acc}_{P_1}(\langle 10\rangle) = 1$. The next string 00 in T refutes 1x, so M_A changes its mind to x0 (i.e., $M_A(T[2]) = \mathtt{x}0$), and $c_A(\langle 10, 00\rangle) = 0$. However, M_A changes its mind again to pattern x on $T[3] = \langle 10, 00, 11\rangle$, so c_A is not a mind change counter for M_A, and M_A is not SMC-optimal. In short, after the string 10 is observed, it is possible to identify the target one-variable pattern with one more mind change, but M_A requires two.

The issue with M_A is that M_A changes its mind on sequence $\langle 10, 00\rangle$ even though $\mathrm{acc}_{P_1}(\langle 10\rangle) = \mathrm{acc}_{P_1}(\langle 10, 00\rangle) = 1$. Intuitively, a mind change optimal learner has to wait until the data decide between the two patterns 1x and x0. As Proposition 3.2.1 indicates, we can design an SMC-optimal learner M for P_1 by "procrastinating" with ? until there is a pattern with the highest accumulation order. For example on the text T described above, our SMC-optimal learner M makes the following conjectures: $M(\langle 10\rangle) =?$, $M(\langle 10, 00\rangle) = \mathtt{x0}$, $M(\langle 10, 00, 11\rangle) = \mathtt{x}$ (see Figure 4.1).

Text T :	10	00	11	0	$\cdots$
Stage n :	1	2	3	4	$\cdots$
Patterns consistent with $T[n]$:	1x, x0, x	x0, x	x	x	$\cdots$
Patterns descriptive of $T[n]$:	1x, x0	x0	x	x	$\cdots$
Accumulation order of $T[n]$:	1	1	0	0	$\cdots$
Output of Angluin's learner M_A :	1x	x0	x	x	$\cdots$
Output of a UMC-optimal learner M :	?	x0	x	x	$\cdots$

Figure 4.1: An illustration of why Angluin's learning algorithm for one-variable patterns is not strongly mind change optimal.

The general specification of the SMC-optimal learning algorithm M is as follows. For a terminal $a \in \Sigma$ let $p^a \equiv p[\mathtt{a}/x]$. The proof of [4, Lemma 3.9] shows that if q is a one-variable pattern such that $L(q) \supseteq \{p^a, p^b\}$ for two distinct terminals a, b, then $L(q) \supseteq L(p)$. So if for a pattern p consistent with data σ, the data contain $\{p^a, p^b\}$, then $L(p)$ is a $\subseteq$-minimum for $P_1|\sigma$ and hence has the highest accumulation order for σ. Thus an SMC-optimal learning algorithm M can proceed by waiting until the data feature p^a and p^b for some pattern p. More precisely, define M as follows.

1. Set $M(\Lambda) :=?$.
2. Given a sequence σ with $S := \mathrm{content}(\sigma)$, check (*) if there is a one-variable pattern p consistent with σ such that $S \supseteq \{p^a, p^b\}$ for two distinct terminals a, b. If yes, output $M(\sigma) := p$. If not, set $M(\sigma) :=?$.

Since there are at most finitely many patterns consistent with σ, the check (*) is effective.

In fact, (*) and hence M can be implemented so that computing $M(\sigma)$ takes time linear in $|\sigma|$. Outline: Let $m = \min\{|s| : s \in S\}$. Let S^m be the set of strings in S of length m.

Define $p_S(i) := a$ if $s(i) = a$ for all $s \in S^m$, and $p_S(i) := \mathtt{x}$ otherwise for $1 \leq i \leq m$. For example, $p_{\{10,11,111\}} = \mathtt{1x}$ and $p_{\{10,01\}} = \mathtt{x}$. Then check for all $s \in S$ if $s \in L(p_S)$. For a one-variable pattern, this can be done in linear time because $|\theta(\mathtt{x})|$, the length of $\theta(\mathtt{x})$, must be $\frac{|s|-term(p_S)}{|p_S|-term(p_S)}$ where $term(p_S)$ is the number of terminals in p_S. For example, if $s = 111$ and $p_S = \mathtt{1x}$, then $|\theta(\mathtt{x})|$ must be 2. If p_S is consistent with S, then there are distinct $a, b \in \Sigma$ such that $\{p^a, p^b\} \subseteq S$. Otherwise no pattern p of length m is consistent with S and hence (*) fails.

It is worth noting that sometimes the mind change efficient learner M_{P_1} may take longer to converge than the Angluin-style learner M_A. For example, let T be a text for the pattern $\mathtt{1x}$ such that $T(0) = 10$ and $T(1) = 11$; then we can verify that the Angluin-style learner M_A in Figure 4.1 converges at time 0, but a mind change efficient learner does not converge until time 1. In general, an Angluin-style learner will converge to the correct one-variable pattern at least as soon as a SMC-optimal learner and strictly sooner on some texts. Thus, the Angluin-style learner dominates the SMC-learner with respect to convergence time in the sense of [62] and [49].

4.3 Mind Change Optimal Identification of Fixed-Length Patterns

Following [67], for each positive integer n, we write T_n to denote the set of canonical patterns of length n. We apply the concept of accumulation order to design a mind change efficient algorithm that identifies T_n for a fixed n. The first step is to find an easily computable, closed-form expression for the accumulation order of a pattern in T_n.

Lemma 4.3.1. *Fix a positive integer $n > 0$, and let p be a pattern in T_n. Then $\mathrm{acc}_{T_n}(p) = n - \#var(p)$, where $\#var(p)$ is the number of distinct variables in p.*

Proof. We prove the claim by downward induction.

Base case: $\#var(p) = n$. Then p is the most general pattern $\mathtt{x_1 x_2 \ldots x_n}$; thus $\mathrm{acc}_{T_n}(p) = 0$.

Inductive step: Assume $\mathrm{acc}_{T_n}(q) = n - \#var(q)$ for all q with $\#var(q) > k$. Consider a pattern p with $\#var(p) = k$. Let $r \in T_n$ be another pattern of length n. If $\#var(r) < k$, then $|S(r)| < |S(p)|$ so $S(p) \not\subseteq S(r)$. Angluin shows that $S(p) = S(r)$ implies $L(p) = L(r)$ [4, Lm. 3.2]. So if $\#var(r) = k$ and $L(p) \neq L(r)$, then $S(p) \neq S(r)$ and so $S(p) \not\subseteq S(r)$ since

$|S(p)| = |S(r)|$. So in either case, $S(p) \not\subseteq S(r)$. As there are only finitely many patterns of length n, this implies that there exists a finite subset $S \subseteq L(p)$ such that $L(r) \neq L(p)$ implies that $\#var(r) > k$ for every pattern $r \in T_n|S$. By the induction assumption, it follows that (1) $\text{acc}_{T_n}(p) \leq n - k$.

Second, since $\#var(p) < n$, it is easy to see that there exists a pattern r such that $\#var(r) = \#var(p) + 1$ and $q \succeq p$; thus $L(p) \subseteq L(q)$. This implies that (2) $\text{acc}_{T_n}(p) \geq n - (k-1) + 1 = n - k$. Combining the above two inequalities (1) and (2), we have $\text{acc}_{T_n}(p) = n - k = n - \#var(p)$. □

To illustrate, the lemma implies that $\text{acc}_{T_n}(\mathtt{x_1x_2 \ldots x_n}) = 0$, $\text{acc}_{T_n}(\mathtt{x_10x_2 \ldots x_{n-1}}) = 1$, and $\text{acc}_{T_n}(\mathtt{x_1x_1 \ldots x_1}) = n - 1$.

Lemma 4.3.1 allows us to design a strongly mind change optimal learner as follows. First we observe that every data sequence σ is topped for the language collection T_n. this is because T_n is finite. For a finite set of ordinals $\{\alpha_1, \alpha_2, \cdots, \alpha_n\}$, its supremum is its maximum. Thus Condition 2 of Proposition 3.2.1 holds vacuously. Condition 1 requires a strongly mind change optimal learner to output ? or the pattern that uniquely tops the given data sequence σ. For a given data sequence σ, we can enumerate the finitely many patterns $T_n|\sigma$ of length n that are consistent with the strings in σ. Then we simply check if any pattern p in $T_n|\sigma$ uniquely maximizes $n - \#var(p)$ or equivalently minimizes $\#var(p)$.

In principle, closed form expressions for the accumulation order of a pattern p in the one-variable pattern space P_1 and in the general pattern space PATTERN, such as Lemma 4.3.1 provides for T_n, would yield mind change optimal learners for these language collections. Finding closed form expressions for acc_{P_1} and $\text{acc}_{\text{PATTERN}}$ are currently open problems [59].

4.4 Summary

We illustrated strongly mind change optimal learning of one-variable and fixed-length patterns. We showed that Angluin's algorithm for learning one-variable pattern is not strongly mind change optimal. We described a different SMC-optimal algorithm for this problem that has linear update time. As we have seen, mind change optimality imposes strong constraints on learners. This means that the theory provides an inductive principle that can be used to design optimal learning algorithms for problems of interest. An analysis in terms of the principle can validate existing inference procedures, as in the case of learning

conservation laws (see [88]) or lead to the development of new ones, as with one-variable patterns and with Bayes net structures (see Chapter 7).

In sum, strong mind change optimality guides the construction of learning algorithms by imposing strong and natural constraints; and the analytical tools we established for solving these constraints in preceding chapters reveal significant aspects of the fine structure of learning problems.

Chapter 5

Mind Change Optimal Learning of Bayes Net Structure

> Decisions made in real time are never perfect.
>
> CIA Deputy Director Noah Vosen (in *The Bourne Ultimatum*)

In this chapter, we model learning the structure of a Bayes net as a language learning problem in the Gold paradigm. This makes it possible to apply identification criteria such as mind change bounds [43, Chapter 12.2][81], text-efficiency (minimizing time or number of data points before convergence) [73, 37], and more importantly the inductive principle of mind-change optimality [60] (see previous chapters in this thesis). Bayes nets are one of the most prominent knowledge representation formalisms [75, 76, 47, 25]. They are widely used to define probabilistic models in a graphical manner. A Bayes net model consists of a structure with parameters. The structure is a directed acyclic graph (DAG) whose edges link the variables of interest. The parameters are conditional probability tables that specify the distribution of a child variable given an instantiation of its parents.

We base our model of Bayes net structure learning on an approach known as "constraint-based" learning [24]. Constraint-based learning views the Bayes net structure as a specification of conditional dependencies of the form $X \not\perp\!\!\!\perp Y|\mathbf{S}$, where X and Y are variables of interest and $\mathbf{S}$ is a set of variables disjoint from $\{X, Y\}$. (Read $X \not\perp\!\!\!\perp Y|\mathbf{S}$ as "variable X is

dependent on variable Y given values for the variables in the set $\mathbf{S}$".) For example, a conditional dependence statement represented by a Bayes net may be "Wetness is dependent on Rain given the Season" (see Figure 5.1 below). In this view, a Bayes net structure is a syntactic representation of a dependency relation [75, Section 3.3]. A dependency relation meets the mathematical definition of a language in the sense of Gold's paradigm, where the basic "strings" are dependence statements of the form "$X \not\perp\!\!\!\perp Y|\mathbf{S}$".

We show that in this learning model, the mind change complexity of learning a Bayes net graph for a given set of variables $\mathbf{V}$ is $\binom{|\mathbf{V}|}{2}$—the maximum number of edges in the graph. Our analysis leads to a characterization of Bayes net learning algorithms that are mind-change optimal. Intuitively, mind-change optimality means that a learner that is efficient with respect to mind changes minimizes the number of mind changes not only globally in the entire learning problem, but also locally in subproblems after receiving some evidence [60]. We show that mind-change optimal BN learners are exactly those that meet the following condition: A mind-change optimal learner may output the vacuous conjecture '?' (for "no guess"). But if the learner does conjecture a graph G, the graph must be the only one that satisfies the observed dependencies *with a minimum number of edges.*

To examine the speed of convergence to a correct Bayes net structure, we apply Gold's notion of dominance in convergence time [37, page 462]. There is a fastest mind-change optimal learner whose convergence time dominates that of all other mind-change optimal learners. The fastest learner is defined as follows: If there is more than one Bayes net pattern G (see Definition 5.2.1) that satisfies the observed dependencies with a minimum number of edges, output ? (for "no guess"). If there is a unique pattern G that satisfies the observed dependencies with a minimum number of edges, output G. Thus standard identification criteria in the Gold paradigm lead to a natural and novel algorithm for learning Bayes net structure.

We also examines the computational complexity of the fastest mind-change optimal learner for Bayes net structure: assuming that P = RP, computing its conjectures is NP-hard.

The chapter is organized as follows. The next section shows the connection and distinction between our model and related work. It is followed by basic definitions from Bayes net theory. Section 5.3 presents and discusses our model of Bayes net structure learning from dependency data as a language learning problem. Then we analyze the mind change

complexity of Bayes net structure learning. Section 5.4 characterizes the mind-change optimal learning algorithms for this problems and describes the fastest mind-change optimal learner. The final two sections define the problem of computing the output of the fastest mind-change optimal learner and show that the problem is NP-hard.

5.1 Related work

Constraint-based algorithms for learning Bayes net structure are a well-developed area of machine learning. Introductory overviews are provided in [24], [70, Chapter 10]. The Tetrad system from Carnegie Mellon University [85] includes a number of constraint-based methods for learning Bayes nets; the theory for these methods is presented in [93]. The papers [61, 18] describe some other constraint-based algorithms. While constraint-based algorithms share the view of a Bayes net graph as defining a dependency relation, a fundamental difference between existing constraint-based approaches and our model is that the existing methods assume access to an oracle that returns an answer for every query of the form "does $X \not\perp\!\!\!\perp Y|\mathbf{S}$ hold"? So in learning theory terms, existing constraint-based methods are best seen as query algorithms for learning from an oracle [6]. Our model in contrast corresponds to the situation of a learner whose evidence grows incrementally over time. Another difference is that existing constraint-based methods presuppose that their oracle indicates whether two variables are conditionally dependent and whether they are conditionally independent. In language learning terms, the constraint-based method has access to both positive data (dependencies) and negative data (independencies). In our analysis, the learner receives positive data (dependencies) only. We discuss the relationship between the oracle-based and our sequential on-line data model further in Section 5.5. To our knowledge, our work is the first analysis of Bayes net learning in the Gold language learning paradigm.

A Bayes net that satisfies a set of given dependencies $\mathcal{D}$ is said to be an I-map for $\mathcal{D}$ [75, page 119]. In these terms, we show the NP-hardness of the following problem: for a given set of dependencies $\mathcal{D}$ represented by an oracle O (Section 5.5), decide whether there is a unique edge minimal I-map G for $\mathcal{D}$, and if so, output G. Bouckaert proved that the problem is NP-hard without the uniqueness condition [13, Lemma 4.5]. However, Bouckaert's proof cannot be adapted for our uniqueness problem, which requires a much more complex reduction. To our knowledge, this is the first NP-hardness result for deciding the existence of a uniquely optimal Bayes net structure for any optimality criterion.

5.2 Bayes Nets: Basic Concepts and Definitions

We employ notation and terminology from [76], [75] and [93]. We consider Bayes net for a set of variables $\mathbf{V} = \{X_1, X_2, \ldots, X_n\}$ where each X_i has a finite number of values or states. A **Bayes net structure** $G = \langle \mathbf{V}, \mathbf{E} \rangle$ for a set of variables $\mathbf{V}$ is a directed acyclic graph (DAG) with node set $\mathbf{V}$. A **Bayes net** (BN) is a pair $\langle G, \theta_G \rangle$ where θ_G is a set of parameter values that specify the probability distributions of children conditional on instantiations of their parents. (Informally, a Bayes net is a directed acyclic graph (DAG) in which nodes represent random variables and edges represent direct dependencies between variables.) A Bayes net $\langle G, \theta_G \rangle$ defines a joint probability distribution over $\mathbf{V}$.

5.2.1 Colliders and d-separation

Two nodes X, Y are **adjacent** in a Bayes net G if G contains an edge $X \rightarrow Y$ or $Y \rightarrow X$. An (undirected) **path** in a graph G is a sequence of nodes such that every two consecutive nodes in the sequence are adjacent in G.

Definition 5.2.1. Let G be a directed graph and p be a path in G. Then a node X is a **collider** on p if X is an interior node on p and X's left and right neighbors on p both have edges pointing to X. The collider X is **shielded** if its right and left neighbours are adjacent; otherwise X is **unshielded** on path p. Removing from G the arrowheads that do not participate in an unshielded collider yields a partially directed graph that is called the **pattern** of G, often denoted pattern(G) or $\pi(G)$.

As long as there is no ambiguity, we shall use G to denote both graphs and patterns. Fig. 5.1 shows a Bayes net from [76, page 15]. In this network, node `wet` is a collider on the path `sprinkler` – `wet` – `rain`; in contrast, node `wet` is not a collider on the path `sprinkler` – `wet` – `slippery`. The pattern of the network has the same skeleton, but contains only two edges that induce the collider **wet**.

Definition 5.2.2 (d-separation)**.** Let G be a Bayes net over variables $\mathbf{V}$.

1. Two nodes X and Y are **d-separated** by a set of nodes $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y\}$, written $(X \perp\!\!\!\perp Y | \mathbf{S})_G$, if for every path p connecting X and Y,

   (a) there exists some collider Z on p such that $\mathbf{S}$ does not contain Z or any descendant of Z, or

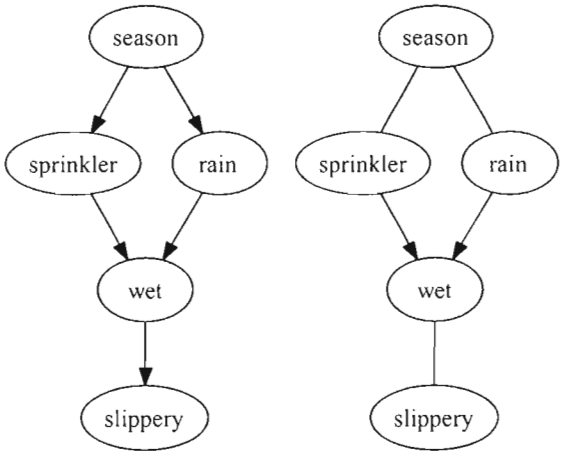


Figure 5.1: Sprinkler network and its pattern.

(b) set $\mathbf{S}$ contains a node in p which is not a collider on p.

2. Two nodes X and Y are **d-connected** by a set of nodes $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y\}$, written $(X \not\perp\!\!\!\perp Y|\mathbf{S})_G$, if X and Y are not d-separated by $\mathbf{S}$. The d-connectedness relation, or **dependency relation**, for a graph is denoted by $\mathcal{D}_G$, that is, $\langle X, Y, \mathbf{S}\rangle \in \mathcal{D}_G$ iff $(X \not\perp\!\!\!\perp Y|\mathbf{S})_G$.

For example, in Fig. 5.1, we have $(\texttt{sprinkler} \perp\!\!\!\perp \texttt{rain}|\{\texttt{season}\})_G$, but $(\texttt{sprinkler} \not\perp\!\!\!\perp \texttt{rain}|\{\texttt{season}, \texttt{wet}\})_G$. Verma and Pearl proved that two Bayes nets G_1 and G_2 represent the same dependency relation iff they have the same pattern (i.e., $\mathcal{D}_{G_1} = \mathcal{D}_{G_2}$ iff PATTERN(G_1) = PATTERN(G_2) [101, Theorem. 1]). Thus we use a pattern as a syntactic representation for a Bayes net dependency relation. The **statement space** over a set of variables $\mathbf{V}$, denoted by $\mathcal{U}_{\mathbf{V}}$, contains all conditional dependency statements of the form $(X \not\perp\!\!\!\perp Y|\mathbf{S})$, where X, Y are distinct variables in $\mathbf{V}$ and $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y\}$.

5.2.2 The Markov Condition and I-maps

Let P be a joint distribution over variables $\mathbf{V}$. If $\mathbf{X}, \mathbf{Y}$ and $\mathbf{S}$ are three disjoint sets of variables, then $\mathbf{X}$ and $\mathbf{Y}$ are **stochastically independent given** S, denoted $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{S})_P$ if $P(\mathbf{X}, \mathbf{Y}|\mathbf{S}) = P(\mathbf{X}|\mathbf{S}) \cdot P(\mathbf{Y}|\mathbf{S})$ whenever $P(\mathbf{S}) > 0$. The distribution P satisfies the **composition principle** if $(\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{S})_P$ whenever $(X \perp\!\!\!\perp \mathbf{Y}|\mathbf{S})_P$ for all variables $X \in \mathbf{X}$.

A Bayes net structure G is an **I-map** of distribution P if $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{S})_P$ implies $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{S})_G$ for all variable sets $\mathbf{X}, \mathbf{Y}$ and variable sets $\mathbf{S}$ disjoint from $\mathbf{X}, \mathbf{Y}$. A basic result in Bayes net theory states that for a given Bayes net structure G and joint distribution P, there is a parametrization θ_G such that P is the joint distribution over $\mathbf{V}$ defined by $\langle G, \theta \rangle$ if and only if G is an I-map of P [70, Theorem 1.4,1.5].

In constraint-based Bayes net learning, it is common to assume that the probability distribution generating the data of interest has a faithful Bayes net representation [93, Theorem 3.2], [76, Chapter 2.4].

Definition 5.2.3. Let $\mathbf{V}$ be a set of variables, G a Bayes net over $\mathbf{V}$, and P a joint distribution over $\mathbf{V}$. Then G is **faithful to** P if $(\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{S})_P$ in $P \iff (\mathbf{X} \not\perp\!\!\!\perp \mathbf{Y}|\mathbf{S})_G$ in G.

Assuming faithfulness, the dependencies in the data can be exactly represented in a Bayes net or a pattern, which is the assumption in our language learning model. It is easy to see that a graph G is faithful to a distribution P if and only if G is faithful with respect to variable pairs, that is, if $(X \not\perp\!\!\!\perp Y|\mathbf{S})_P$ in $P \iff (X \not\perp\!\!\!\perp Y|\mathbf{S})_G$ in G for all variables X, Y. Therefore constraint-based methods focus on conditional dependencies of the form $(X \not\perp\!\!\!\perp Y|\mathbf{S})$, which is the approach we follow throughout the chapter.

Next we introduce our model of Bayes net structure learning, which associates a language collection $\mathcal{L}_\mathbf{V}$ to a given set of variables $\mathbf{V}$ of interest; the language collection $\mathcal{L}_\mathbf{V}$ comprises all dependency relations defined by Bayes net structures.

As Gold's paradigm does not specify how linguistic data are generated for the learner, our model does not specify how the observed dependencies are generated. In practice, a Bayes net learner obtains a random sample S and applies a suitable statistical criterion to decide if a dependency $X \not\perp\!\!\!\perp Y|\mathbf{S}$ holds. One way in which data for our model can be generated from random samples is the following (we discuss another interpretation in Section 5.5): For every triple $X \not\perp\!\!\!\perp Y|\mathbf{S}$ with $\{X, Y\} \cap \mathbf{S} = \emptyset$, a statistical test is performed with $X \perp\!\!\!\perp Y|\mathbf{S}$ as the null hypothesis. (For small numbers of variables, this is a common procedure in statistics called "all subsets variable selection" [107, page 59].) If the test rejects the null hypothesis, the dependency $X \not\perp\!\!\!\perp Y|\mathbf{S}$ is added to the dependency data; otherwise no conclusion is drawn. This is similar to how many constraint-based systems answer queries to a dependency oracle: given a query "Does $X \not\perp\!\!\!\perp Y|\mathbf{S}$ holds", the system performs a statistical test, and answers "yes" if the test rejects the hypothesis $X \perp\!\!\!\perp Y|\mathbf{S}$, and "no" otherwise. The assumption that this procedure yields correct results is called the assumption of valid statistical testing

[24, Section 6.2]. Compared to this assumption, our model is more realistic in two respects. First, the model assumes only that dependency information is available, but does not require independence data. In fact, many statisticians hold that no independence conclusion should be drawn when a statistical significance test fails to reject an independence hypothesis [34]. Second, our model does not assume that the dependency information is supplied by an oracle all at once, but allows that more information may become available as the sample size increases.

Since the set of dependency relations $\mathcal{L}_{\mathbf{V}}$ constitutes a language collection in the sense of the Gold paradigm, we can employ standard identification criteria to analyze this learning problem. We begin by applying a fundamental result in Bayes net theory to determine the mind change complexity of the problem.

5.3 The Mind Change Complexity of Learning Bayes Net Structure

Since we are considering Bayes nets with finitely many variables, the statement space $\mathcal{U}_{\mathbf{V}}$ is finite, so the language collection $\mathcal{L}_{\mathbf{V}}$ containing all Bayes net-dependency relations is finite and therefore $\mathcal{L}_{\mathbf{V}}$ has finite thickness. Hence we have the following corollary.

Observation 2. *Let* $\mathbf{V}$ *be a set of variables. There exists a learner* Ψ *that identifies* $\mathcal{L}_{\mathbf{V}}$ *with mind change bound* $k \iff$ *the inclusion depth of* $\mathcal{L}_{\mathbf{V}}$ *is at most* k*, that is,* $\mathrm{ID}(\mathcal{L}_{\mathbf{V}}) \leq k$.

The observation shows that the mind change complexity of learning a Bayes net depends on the inclusion depth of the space of dependency relations that can be represented in a Bayes net. A fundamental result in Bayes net theory allows us to determine the inclusion depth of a dependency relation in $\mathcal{L}_{\mathbf{V}}$.

An edge $A \rightarrow B$ is **covered** in a DAG G if the parents of B are exactly the parents of A plus A itself (e.g., edge $A \rightarrow B$ is covered in Figure 5.2). The operation that reverses the direction of the arrow between A and B is a **covered edge reversal**.

Theorem 5.3.1 (Meek-Chickering). *Let* G *and* H *be two Bayes nets over the same set of variables* $\mathbf{V}$. *Then* $\mathcal{D}_G \subseteq \mathcal{D}_H \iff$ *the Bayes net* H *can be transformed into the Bayes net* G *by repeating the following two operations: (1) covered edge reversal, and (2) single edge deletion.*

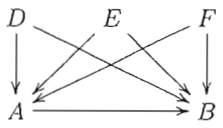


Figure 5.2: A simple network with a covered edge. Edge $A \rightarrow B$ is covered, but $D \rightarrow A$ is not covered.

The theorem was conjectured by Meek [65] and proven by Chickering [19, Theorem 4].

Corollary 5.3.1. *Let G be a Bayes net over a set of variables $\mathbf{V}$. Then the inclusion depth of the Bayes net-dependence relation $\mathcal{D}_G$ equals the number of missing edges in G, and we take the inclusion depth of a Bayes net to be the inclusion depth of its dependence relation. In particular, the totally disconnected network has inclusion depth $\binom{|\mathbf{V}|}{2}$; a complete network has the inclusion depth* 0.

Proof. We use downward induction on the number of edges n in graph G. Let $N = \binom{|\mathbf{V}|}{2}$. Base case: $n = N$. Then G is a complete graph, so $\mathcal{D}_G$ contains all dependency statements in the statement space $\mathcal{U}_{\mathbf{V}}$, and therefore G has 0 inclusion depth. Inductive step: Assume the hypothesis for $n+1$ and consider a graph G with n edges. Add an edge to G to obtain a Bayes net G' with $n+1$ edges that is a super-graph of G'. The definition of d-separation implies that $\mathcal{D}_G \subset \mathcal{D}'_G$. By inductive hypothesis, there is an inclusion chain $\mathcal{D}'_G \subset \cdots \subset D_{G_{N-(n+1)}}$ consisting of Bayes net dependency relations. Hence the inclusion depth of G is at least $N - (n+1) + 1 = N - n$.

To show that the inclusion depth of G is exactly $N - n$, suppose for contradiction that $\mathrm{ID}(\mathcal{D}_G) > N - n$. Then there is an inclusion chain $\mathcal{D}_G \subset \mathcal{D}_{H_1} \subset \mathcal{D}_{H_2} \subset \cdots \subset \mathcal{U}_{\mathbf{V}}$ of length greater than $N - n$. So the inclusion depth of $\mathcal{D}_{H_2}$ is at least $N - (n+1)$ and the inclusion depth of $\mathcal{D}_{H_1}$ is at least $N - n$. Hence by inductive hypothesis, the number of edges in H_2 is at most $n+1$ and in H_1 at most n. So at least two of the graphs G, H_1, H_2 have the same number of edges. Without loss of generality, assume that H_1 and H_2 have the same number of edges. Since $\mathcal{D}_{H_1} \subset \mathcal{D}_{H_2}$, Theorem 5.3.1 implies that H_1 can be obtained from H_2 with covered edge reversals. But covered edge reversals are symmetric, so we also have that $\mathcal{D}_{H_2} \subseteq \mathcal{D}_{H_1}$, which contradicts the choice of H_1 and H_2. So $\mathrm{ID}(\mathcal{D}_G) = N - n$, which completes the inductive proof. □

The corollary characterizes the inclusion depth of the Bayes net dependence relation $\mathcal{D}_G$ for a graph G in terms of a simple syntactic feature of G, namely the number of edges.

Together with Proposition 3.3.1, the corollary implies that the mind change complexity of identifying a Bayes Net structure over variables $\mathbf{V}$ from dependency data is given by the maximum number of edges over $\mathbf{V}$.

Theorem 5.3.2. *For any set of variables* $\mathbf{V}$*, the inclusion depth of* $\mathcal{L}_{\mathbf{V}}$ *is* $\binom{|\mathbf{V}|}{2}$*. So the mind change complexity of identifying the correct Bayes Net structure from dependency data is* $\binom{|\mathbf{V}|}{2}$*.*

The next section characterizes the Bayes net learning algorithms that achieve optimal mind change performance.

5.4 Mind Change Optimal Learners for Bayes Net Structure

The complexity of a learning problem is a lower bound on the best possible performance by a learning algorithm. A goal of learning theory is to design algorithms that achieve the best possible performance. This section analyzes mind-change optimal algorithms for identifying Bayes net structure. The intuition underlying mind-change optimality is that a learner that is efficient with respect to mind changes minimizes mind changes not only globally in the entire learning problem, but also locally in subproblems after receiving some evidence [60]. We formalize this idea as in [60, Definition 2.3]. If a mind change bound exists for $\mathcal{L}$ given σ, we write $\mathbf{MC}_{\mathcal{L}}(\sigma)$ for the least k such that $\mathcal{L}$ is identifiable with k mind changes given σ. For example, given a sequence σ of dependencies, let $G = (\mathbf{V}, E)$ be a Bayes net that covers (satisfies) the dependencies in σ with a minimum number of edges. Then the mind change complexity $\mathbf{MC}_{\mathcal{L}_{\mathbf{V}}}(\sigma)$ is $\binom{|\mathbf{V}|}{2} - |E|$.

Applying Corollary 3.3.1 to Bayes net learners yields the following corollary.

Corollary 5.4.1. *Let* Ψ *be a Bayes net learner that identifies the correct Bayes net structure for a set of variables* $\mathbf{V}$*. The learner* Ψ *is SMC-optimal* $\iff$ *for all dependency sequences* σ*, if the output of* Ψ *is not ?, then* Ψ *outputs the unique edge-minimal pattern for the dependencies* $\mathcal{D} = \text{content}(\sigma)$*.*

It is easy to implement a slow SMC-optimal Bayes net learner. For example, for a given set of dependencies $\mathcal{D}$ it is straightforward to check if there is a pattern G that covers exactly those dependencies (i.e., $\mathcal{D}_G = \mathcal{D}$). So an SMC-optimal learner could output a pattern G if there is one that matches the observed dependencies exactly, and output ? otherwise. But

such a slow learner will require exponentially many dependency statements. We have seen in Chapter 3 that there is a fastest SMC-optimal learner for each language collection with finite inclusion depth.

Applying Observation 1 to identifying Bayes net-dependency relations leads to the following algorithm for identifying a Bayes net pattern.

Corollary 5.4.2. *Let* $\mathbf{V}$ *be a set of variables. For a given sequence of dependencies* σ*, the learner* $\Psi^{\mathbf{V}}_{\text{fast}}$ *outputs ? if there is more than one edge-minimal pattern that covers the dependencies in* σ*, and otherwise outputs the unique edge-minimal pattern* P *consistent with* σ*. The learner* $\Psi^{\mathbf{V}}_{\text{fast}}$ *is SMC-optimal and identifies the correct pattern uniformly faster than any other SMC-optimal Bayes net structure learner.*

Example 5.4.1. Let $\mathbf{V} = A, B, C, D$. If

$$\begin{aligned}\sigma = (&B \not\perp\!\!\!\perp D, B \not\perp\!\!\!\perp D|A, B \not\perp\!\!\!\perp D|C,\\ &C \not\perp\!\!\!\perp D, C \not\perp\!\!\!\perp D|A, C \not\perp\!\!\!\perp D|B,\\ &B \not\perp\!\!\!\perp C|D, B \not\perp\!\!\!\perp C|\{A, D\}),\end{aligned}$$

then $\Psi^{\mathbf{V}}_{\text{fast}}$ outputs the graph shown in Figure 5.3(a). Now suppose after some time, we have accumulated more data, and obtain the following data sequence

$$\begin{aligned}\sigma' = (&B \not\perp\!\!\!\perp D, B \not\perp\!\!\!\perp D|A, B \not\perp\!\!\!\perp D|C,\\ &C \not\perp\!\!\!\perp D, C \not\perp\!\!\!\perp D|A, C \not\perp\!\!\!\perp D|B,\\ &B \not\perp\!\!\!\perp C|D, B \not\perp\!\!\!\perp C|\{A, D\}\\ &A \not\perp\!\!\!\perp B, A \not\perp\!\!\!\perp B|C, A \not\perp\!\!\!\perp B|D,\\ &A \not\perp\!\!\!\perp C, A \not\perp\!\!\!\perp C|B, A \not\perp\!\!\!\perp C|D, B \not\perp\!\!\!\perp C),\end{aligned}$$

then the output of $\Psi^{\mathbf{V}}_{\text{fast}}$ is shown in Fig 5.3(b).

We have seen that the criteria of mind-change optimality and convergence speed determine a unique, natural and novel method for learning Bayes net structure. The next section analyzes the run-time complexity of this method; we show that computing the output of the learner is NP-hard (assuming that RP = P).

Figure 5.3: Intermediate outputs of the fast SMC-optimal learner. Figure (a) shows the output on the data sequence σ in Example 5.4.1; figure (b) shows the output on the data sequence σ'

5.5 Complexity Model for Constraint-Based Bayes Net Learners

This section describes the standard approach of analyzing the complexity of constraint-based learners in the Bayes net literature. We also state some known results from complexity theory that we require as background.

As with any run-time analysis, an important issue is the representation of the input to the algorithm. The most straightforward approach for our learning model would be to take the input as a list of dependencies, and the input size to be the size of that list. However, in practice constraint-based learners do not receive an explicitly enumerated list of dependencies, but rather they have access to a dependency oracle (cf. Section 5.3). Enumerating relevant dependencies through repeated queries is part of the computational task of a constraint-based learner. Accordingly, the standard complexity analysis takes a dependency oracle and a set of variables as the input to the learning algorithm (e.g., [21, Definition 12],[13]. The oracle is assumed to be represented syntactically in a reasonably concise way. For example, it may be a Turing Machine that computes the characteristic function of a given dependency relation $\mathcal{D}$. To facilitate comparison with the related literature, we follow the approach of representing the set of dependencies in the data through an oracle. We remark that the NP-hardness result for the learner $\Psi^{\mathbf{V}}_{\text{fast}}$ still holds if the input are enumerated dependencies (see [90]).

Definition 5.5.1. A dependency oracle O for a variable set $\mathbf{V}$ is a Turing Machine that takes as input dependency queries from the statement space $\mathcal{U}_{\mathbf{V}}$ and returns, in constant time, either "yes" or "?" (meaning "unknown").

The dependency relation associated with oracle O is given by $\mathcal{D}_O = \{X \not\perp\!\!\!\perp Y|\mathbf{S} \in \mathcal{U}_{\mathbf{V}} : O \text{ returns "yes" on input } X \not\perp\!\!\!\perp Y|\mathbf{S}\}$. We note that our model of learning Bayes net structure can be reformulated in terms of a sequence of oracles: Instead of a complete sequence of dependency statements for a dependence relation $\mathcal{D}_G$, the learner could be presented with a sequence of dependency oracles $O_1, \ldots, O_n, \ldots$ such that $\mathcal{D}_{O_i} \subseteq \mathcal{D}_{O_{i+1}}$ and $\bigcup_{i=1}^{\infty} O_i = \mathcal{D}_G$. The notion of a one-sided dependency oracle makes clear the relation to the standard constraint-based learning model which presupposed the existence of a two-sided oracle: (1) a dependency oracle assumes valid statistical testing only for dependence findings, not independence conclusions, and (2) our sequential data model envisions the dependency model becoming more informative as more data is obtained, whereas the standard model considers a completely informative oracle at a single point in time.

We will reduce the problem of computing the output of the fastest mind change optimal learner $\Psi_{\text{fast}}^{\mathbf{V}}$ to deciding the existence of a unique exact cover by 3-sets; this problem is defined as follows.

Definition 5.5.2. Unique Exact Cover by 3-Sets (UX3C)

Instance A finite set X with $|X| = 3q$ and a collection C of 3-element subsets of X.

Question Does C contain a unique *exact cover* for X, that is, a subcollection $C' \subseteq C$ such that every element of X occurs in exactly one member of C'?

We will apply the following result, which is well-known in complexity theory. The class RP comprises the decision problems that can be decided in polynomial time with a randomized algorithm [74, Definition 11.1].

Proposition 5.5.1. *A polynomial time algorithm for* UX3C *yields a polynomial time algorithm for the satisfiability problem* SAT *provided that* $P = RP$. *So* UX3C *is NP-hard.*

The proof is based on the famous theorem of Valiant and Vazirani which gives a probabilistic reduction of SAT to UNIQUE SAT. Standard reductions (such as those in [74, 33]) show that UNIQUE SAT reduces to UX3C.

5.6 Computational Complexity of Fast Mind Change-Optimal Identification

Computing the conjectures of the fastest SMC-optimal learner $\Psi^{\mathbf{V}}_{\text{fast}}$ poses the following computational problem.

Definition 5.6.1. UNIQUE MINIMAL I-MAP

Input A set of variables $\mathbf{V}$ and a dependency oracle O for $\mathbf{V}$.

Output If there is a *unique* DAG pattern P that covers the dependencies in O with a minimal number of edges, output P. Otherwise output ?.

This is a function minimization problem; the corresponding decision problem is the following.

Definition 5.6.2. UNIQUE I-MAP

Instance A set of variables $\mathbf{V}$, a dependency oracle O for $\mathbf{V}$, and a bound k.

Question Is there a DAG pattern P such that: P covers the dependencies in O, every other DAG pattern P covering the dependencies in O has more edges than P, and P has at most k edges?

Clearly an efficient algorithm for the function minimization problem yields an efficient algorithm for UNIQUE I-MAP. We will show that UNIQUE I-MAP is NP-hard, assuming that $P = RP$.

Theorem 5.6.1. *Assuming that* $P = RP$, UX3C$\leq_P$UNIQUE I-MAP$\leq_P$UNIQUE MINIMAL I-MAP. *So* UNIQUE MINIMAL I-MAP *is NP-hard.*

Proof Outline. We transform UX3C to UNIQUE I-MAP.

Let an arbitrary instance of UX3C be given by a set $\mathbf{X}$ such that $|\mathbf{X}| = 3q$ and a collection $\mathbf{C}$ of 3-element subsets of $\mathbf{X}$. We construct a set of variables $\mathbf{V}$, a dependence oracle O for $\mathbf{V}$, and a bound k such that $\mathbf{C}$ has an exact subcover for $\mathbf{X}$ if and only if there is a unique I-map over $\mathbf{V}$ for O with at most k edges.

First, suppose $\mathbf{X} = \{x_1, \ldots, x_{3q}\}$ and $\mathbf{C} = \{c_1, \ldots, c_m\}$; the variables in V belong to one of the following types:

1. A **root variable** R;
2. A **member variable** X_j for each point $x_j \in \mathbf{X}$, where $1 \leq j \leq 3q$.
3. A **set variable** C_i for each set $c_i \in \mathbf{C}$, where $1 \leq i \leq m$.

Second, construct the following procedure M as the dependency oracle O. Given a dependency query $V_1 \not\perp\!\!\!\perp V_2 | \mathbf{S}$?,

1. M returns "yes" if one of the following conditions is satisfied:
   (a) $V_1 = C_i$ is a set variable, $V_2 = X_j$ is a member variable, and $x_j \in c_i$.
   (b) $V_1 = X_j$ and $V_2 = X_k$ are both member variables, and $\mathbf{S}$ contains a set variable C_i such that $c_i \in \mathbf{C}$ contains both x_j and x_j.
   (c) $V_1 = R$ is the root variable, and V_2 is a member variable, and $\mathbf{S}$ is the empty set $\emptyset$.
   (d) $V_1 = R$ is the root variable, and $V_2 = X_j$ is a member variable, and $C_i \notin \mathbf{S}$ for all c_i containing x_j.
2. M returns "?" otherwise.

Finally, we let the bound $k = 3m + q$.

Intuitively, the reduction makes sure that a minimal I-map has the form shown in Figure 5.4. Clause 1a enforces edges connecting C_i and X_j. Clause 1b enforces the arrows from X_j and X_k to C_i. Clause 1c enforces for an edge connecting the root variable R and a set variables C_i, the arrow is always pointing to R. Clause 1d requires that every member variable is d-connected to the root variable. It allows that the d-connection is of the form $X - C - R$ where x is in c. The intuition is that this is the most edge-efficient way to connect the member variables to the root variable because if a set c contains three variables x_1, x_2, and x_3, adding a single edge $C \rightarrow R$ d-connects all three member variables X_1, X_2, and X_3 at once. The formal verification of this intuition can be found in [90].

□

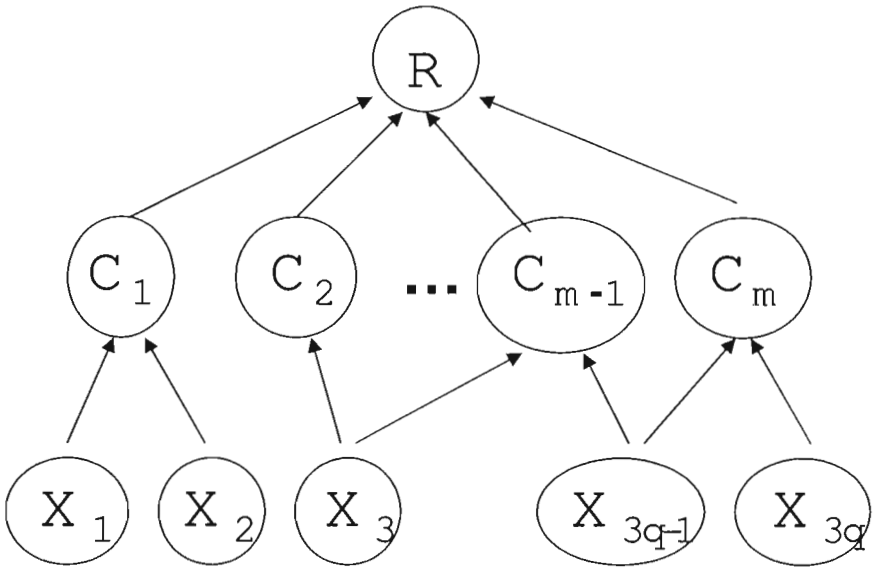


Figure 5.4: The basic graph for the NP-hardness proof. A set cover of size q corresponds to q edges of the form $C \rightarrow R$.

5.7 Summary

This chapter applied learning-theoretic analysis to a practically important learning problem: identifying Bayes net structure. We presented a learning model of this task in which learning is based on conditional dependencies between variables of interest. This model fits Gold's definition of a language learning problem, so identification criteria from Gold's paradigm apply. We considered mind-change optimality and text efficiency (or dominance in convergence time). The mind change complexity of identifying a Bayes net over variable set $\mathbf{V}$ is $\binom{|\mathbf{V}|}{2}$, the maximum number of edges in a graph with node set $\mathbf{V}$. There is a unique mind-change optimal learner $\Psi^{\mathbf{V}}_{\mathsf{fast}}$ whose convergence time dominates that of all other mind-change optimal learners. This learner outputs a Bayes net G if G is the unique graph satisfying the observed dependencies with a minimum number of edges; otherwise $\Psi^{\mathbf{V}}_{\mathsf{fast}}$ outputs ? for "no guess". In many language learning problems, it its plausible to take the mind change complexity of a hypothesis as a form of simplicity [60, Section 4]. Our results establish that the notion of simplicity corresponding to mind change complexity of a Bayes net graph G is the inclusion depth of G, which we showed is measured by the number of edges in G. In these terms, the fastest mind-change optimal learner outputs a uniquely simplest Bayes net consistent with the dependency data if there is one, and outputs "no guess" otherwise. Using the number of edges as simplicity criterion to guide learning appears to be a new idea in the Bayes net literature.

We have seen that determining whether there is a uniquely simplest (edge-minimal) Bayes net for a given set of dependencies is NP-hard. The NP-hardness result implies that a practical, polynomial-time application of the fastest mind-change optimal learner must use search heuristics. Many Bayes net learning algorithms are based on an optimality measure or score (see Section 6.2); exact optimization of this measure is typically NP-hard [21], so the learning algorithm employs heuristic search instead. Research in this area has developed various strong local search methods for finding optimal Bayes nets [13, 70]. These methods can be applied to search for Bayes nets that optimize our simplicity criterion as well. The result would be a practical, novel constraint-based Bayes net learner that has a learning-theoretic foundation. We shall discuss the implications in following chapters.

In sum, applying Gold-style identification criteria leads to a fruitful analysis of Bayes net learning. We gained insights into the structure of the hypothesis space (determining its mind change complexity or inclusion depth), which leads to a natural new notion of

simplicity for Bayes nets (simpler graphs have fewer edges) that can guide learning with a theoretical foundation.

Chapter 6

From Theory to Practice: Approximating an MC-Optimal Bayes Net learner

> There is no magic about numeric methods, and many ways in which they can break down. They are a valuable aid to the interpretation of data, not sausage machines automatically transforming bodies of numbers into packets of scientific fact.
>
> F.H.C. Marriott

In the previous chapter, the inductive principle of mind change optimality leads to fast mind change-optimal identification of Bayes net structure. Directly implementing the learning strategy is NP-hard. From this chapter on, we shall study approximations to fast mind change-optimal identification.

6.1 The Principle of I-map Learning

In Chapter 3.4, we have seen that for problems with finite accumulation order, every SMC-optimal regular learner (a learner that outputs either a single language or ?) corresponds

to an SMC-optimal set learner. That set learner always outputs the set of languages with the highest accumulation order. Hence we can reformulate our problem as follows:

Problem 6.1.1. *Given a list of dependencies, find the* set *of edge-minimal graphs that are consistent with the observed dependencies.*

This task apparently is no easier than deciding uniqueness, but it suggests a further approximation: looking for *some* minimal I-map that satisfies the observed dependencies. We call this the principle of I-map learning.

Problem 6.1.2. *Given a list of dependencies, find an edge-minimal graph that is consistent with the dependencies.*

The difference between problems 6.1.1 and 6.1.2 is somewhat analogous to the difference between model averaging and model selection.

In practice, we do not have a list of dependencies, instead we have a sample s from which a set of dependencies $\mathcal{D}(S)$ can be inferred by a statistical testing strategy T. Hence we have the following approximation:

Problem 6.1.3. *Given a sample s, find an edge-minimal graph that is consistent with $\mathcal{D}(s,T)$, a set of dependencies inferred from s using testing strategy T.*

For this problem, two decisions are critical in algorithm design: how to get $\mathcal{D}(s,T)$ and how to find an edge-minimal graph.

Assuming that we use statistical testing to infer dependencies, there are more than one ways to get $\mathcal{D}(s,T)$. First, we may run a constraint-based algorithm (e.g., the PC algorithm described later in the chapter) over the sample, and then collect the dependencies resulted. Second, we can bound the size of conditioning set (hence bound the time complexity of statistical tests) to a constant k, and then exhaustively test all conditional independence statements. Experiments show that the second approach is more robust and often returns a set of dependencies that better reduces underfitting.

To find an edge-minimal graph consistent with $\mathcal{D}(s,T)$, a common strategy is to use a score-based heuristic. First, we can search with a score based on the number of edges. Second, we can use the GES search heuristic with BDeu score (described later). Experiments show that GES search with BDeu score can be modified into a better performed algorithm (see Chapter 7).

When talking about approximation, a natural question is whether existing algorithms qualify as good solver for Problem 6.1.3. However, we shall see that most of the existing algorithms, given a sample of small to medium size, do not output a graph that is consistent with known the dependencies that can be easily inferred from the sample.

In learning theory, a **consistent learner** is a learner that always outputs a hypothesis that is consistent with the data[1]. For a finite hypothesis space, inconsistency is a source of unnecessary mind changes (see Proposition 3.2.1). In the previous chapter, dependencies are modeled as data points presented to a learner, while a Bayes net structure is modeled as a collection of such points. Therefore the principle of I-map learning requires a learner to output a minimal consistent hypothesis; this behavior is aligned with some other algorithms (e.g., the closure algorithm for intersection-closed concept classes [9]).

More importantly, the principle of I-map learning respects Pearl's original definition of Bayes net [75, page 119], which is stated here.

Definition 6.1.1 (Pearl's definition of Bayes net)**.** Given a probability distribution P on a set of variables $\mathbf{U}$, a DAG $D = (\mathbf{U}, \mathbf{E})$ is called a **Bayesian network** of P if D is a minimal I-map of P.

Empirical evidence in this chapter shows that two typical algorithms for learning Bayes net structure, the GES algorithm with the BDeu score [40] and the PC algorithm [93], often produce inconsistent hypotheses, and hence violating the principle of I-map learning. Since these two algorithms represent two most influential methodologies for learning Bayes net structure, a new method is needed to address the inconsistency.

6.2 Underfitting in Score-base Learning of Bayes Net Structure

In score-based learning, a scoring function $S(\cdot, \cdot)$ maps the given sample d and a candidate structure G to a real number $S(\mathrm{d}, G)$. Hence the learning problem is converted to an optimization problem in which a structure with the maximum score is sought.

[1]Unfortunately, consistency has a very different meaning in statistics. A point estimator is said to be consistent if it converges to a constant when sample size goes to infinity (see [16, Section 5.5]). In this thesis, we will not use consistency in this sense.

6.2.1 Structure Learning as Model Selection

A typical scoring function tries to optimize the predictive power of the resulting structure. Hence it assigns higher scores to structures that both fit the sample well and are parsimonious (to suppress over-fitting). Therefore, most score functions in structure learning derive from model selection criteria such as BIC, AIC, Cross Validation, and so on.

A Bayes net structure G defines a family of distributions $f_{(G,\theta)}$, where θ is an instantiation of parameters over G. In model selection, we look for a pair of G and θ to minimize the discrepancy between $f_{(G,\theta)}$ and the true distribution (in the parlance of model selection, the operating model) g. With only a (finite) sample D from g, the discrepancy can only be estimated (for example, by taking average discrepancy of possible instantiations of g). Although different model selection criteria should be used for different definitions of discrepancies [58, Chapter 10], they all look for the balance between data fitness and parsimoniousness of models. That is, they respect a syntactic notion of Occam's Razor to avoid overfitting[2] [29, Section 9.6.5].

Many scoring functions are special cases of the Bayesian score.

Definition 6.2.1 (Bayesian score)**.** Let G be a Bayes net structure and d be a sample. Then the Bayesian score of G is the joint probability (or density in the continuous case)

$$P(\mathrm{d}, G) = P(G) \int_{\Theta} P(\mathrm{d}, \Theta|G) = P(G) \int_{\Theta} P(\mathrm{d}|\Theta, G) \cdot P(\Theta|G),$$

where $P(G)$ is the prior distribution of G, and $P(\Theta|G)$ is prior distribution of Θ given the structure G.

Machine learning researchers impose various assumptions on $P(\mathrm{d}|\Theta, G)$ and $P(\Theta|G)$; Heckerman et. al. list 7 such assumptions in [40]. These assumptions lead to the widely adopted BDeu score [40, Theorem 5].

Definition 6.2.2 (BDeu score)**.** Let $\mathbf{V}$ be a set of random variables and G_c be a complete Bayes net structure over $\mathbf{V}$ (i.e., every two nodes in G_c are adjacent). Let N' be a Dirichlet equivalent sample size for model G_c. The **BDeu score** of a Bayes net structure G over V is defined to be

$$P(\mathrm{d}, G) = P(G) \cdot \prod_{i=1}^{n} \prod_{j=1}^{q_i} \frac{\Gamma(N'_{ij})}{\Gamma(N'_{ij} + N_{ij})} \cdot \prod_{k=1}^{r_i} \frac{\Gamma(N'_{ijk} + N_{ijk})}{\Gamma(N'_{ijk})}$$

[2] This is in contrast to maximizing accumulation order, which can be understood a semantic notion of Occam's Razor.

where q_i is the number of states for node i's parents, r_i is the number of states for node i, N_{ijk} is the number of occurrences of that specific instantiation, $N_{ij} := \sum_k N_{ijk}$, $N'_{ijk} := \frac{N'}{q_i \cdot r_i}$, and $N'_{ij} := \frac{N'}{q_i}$.

A typical setup for $P(G)$ and N' is $P(G) := 0.001^k$, where k is the number of free parameters in G, and $N' := 10$ (see [19, Section 6]).

The problem of optimizing the BDeu score is NP-hard in general [20, 21]; hence a search heuristic is used. The Greedy Equivalence Search (GES) is a heuristic designed to search over the space of dependence-equivalent Bayes net structures. The GES search with BDeu score is implemented in most Bayes learning software packages, such as Tetrad [77], the BNT for Matlab [69], and Weka [14]. We shall give a more detailed description of the GES search in later chapters. For now, it suffices to know that it is a popular score-based search algorithm.

6.2.2 Underfitting in Score-based Learning

Maximizing a scoring function alone often leads to Bayes nets that underfit data. Hence maximizing score often sacrifices the principle of I-map learning.

Most scoring functions take the form of the likelihood regularized with the number of parameters in a Bayes net. For example, the BIC score, which approximates the Bayesian score [70, p 457], has the form

$$\text{BIC}(\text{d}, G) := \log(P(\text{d}|\hat{\Theta}, G)) - \frac{k}{2} \log m,$$

where $\hat{\Theta}$ is the maximal-likelihood estimation of Θ, m is the size of sample d, and k is the number of parameters in Θ. Therefore, an algorithm based on such type of scoring functions is biased towards Bayes net structures with *minimal parameters*. Note that the principle of I-map learning requires an algorithm, in contrast, to be biased towards *I-maps* with **minimal edges**. Hence two potential problems of score-based learning are:

1. Minimizing the number of parameters is not always aligned with minimizing the number of edges.

2. An underfitting structure often has higher score than an I-map, although it does not fit dependencies because it has fewer number of parameters.

Later experiments illustrate both problems.

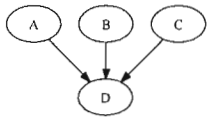


Figure 6.1: A Bayes net with fewer edges than parameters.

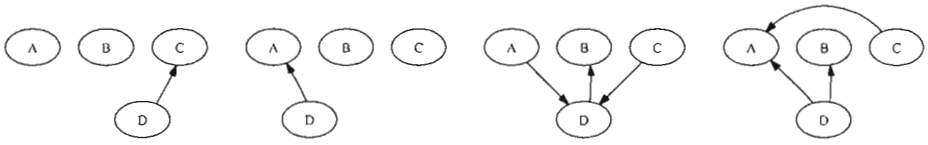
P(D=0)	$(A,B)=(0,0)$	$(A,B)=(1,0)$	$(A,B)=(0,1)$	$(A,B)=(1,1)$
$C=0$	0.2920	0.0813	0.5247	0.5586
$C=1$	0.0382	0.8159	0.3206	0.7733

Table 6.1: The conditional probability of variable D given variables A, B, and C of the Bayesian network in Figure 6.1.

Parameters vs. Edges

Let G be the Bayesian network shown in Figure 6.1. The variables A, B, and C denote three fair coins. The conditional probability table for D given A, B, C is shown in Table 6.1. This Bayes net obviously has a large number of parameters (9) and a relatively small number of edges (4).

We randomly draw samples of various sizes from G, and the networks of the highest BDeu score (with structure priors 0.001^k and the equivalent sample size 10, where k is the number of parameters, see [19, Section 6]) are shown in Figure 6.2. None of them has the same DAG pattern as G. The parameters in Table 6.1 do not entail extra independencies for the target distribution; hence every I-map of the distribution should contain all edges in G. Therefore none of the resulting graphs with the highest BDeu score in Figure 6.2 is an I-map. This simple experiment shows that a score-based method is not designed to respect the principle of I-map learning and it often fails to return even an I-map.



(a) 50 data points (b) 100 data points (c) 150 data points (d) 200 data points

Figure 6.2: Structures with the highest BDeu score over the variable set $\{A, B, C, D\}$ on four samples generated from the Bayesian network in Figure 6.1. In the BDeu score used, the structure prior is 0.001^k, where k is the number of parameters in the structure; the equivalent sample size is 10. These are the settings used by Chickering [19] and Tetrad [77].

Underfitting of GES+BDeu

In most implementations of score-based learning, greedy search is taken; therefore the Bayes net structure returned may not have the highest score. The existence of greedy search does not help reduces the problem of underfitting. We again consider the GES algorithm with the BDeu score (GES+BDeu). This is the default combination used in various Bayes net structure learning packages, such as Tetrad [77] and BNT [69].

We use an experiment to demonstrate the underfitting problem in GES+BDeu. We randomly generated Bayes nets over a set of binary variables, and then generated random samples of various sizes from these networks and feed them to GES+BDeu. For the priors of GES+BDeu, we used the default setting in Tetrad. We here only describe the case of 10 variables; other settings return similar results. The number of edges in the resulting networks are box-plotted in Figures 6.6, 6.7, 6.8, and 6.9. The ratios of the number of parameters are box-plotted in Figures 6.10, 6.11, 6.12, and 6.13. (The number of parameters is used by the MDL score [84, 38] to measure the complexity of a model. For the sake of comparison, we also show in the figures the results of the PC algorithm and our IGES algorithm, which will be introduced in the following chapter.) These figures clearly indicate that GES+BDeu underfits small-to-medium sized samples.

6.3 Underfitting in Constraint-based Learning of Bayes Net Structure

Earlier works by Spirtes, Glymour, Scheines [93], and Pearl [76] have shown that it is possible to recover Bayes nets from observational data using conditional independence information. This is the so-called constraint-based approach of learning Bayes net structure. The challenge of implementing constraint-based learning lies mainly at orienting arrows using noisy independence information. In this section, we briefly review the assumptions and mechanisms underlying current constraint-based algorithms, in particular the PC algorithm. We point out a problem in the way the PC estimates independence information.

Before moving on to specific algorithms, we introduce a concept that helps us understand constraint-based learning.

Definition 6.3.1 (d-separating collection)**.** Let $G = (\mathbf{V}, A)$ be a Bayes net, and X, Y be two nodes in G.

1. The **complete d-separating collection** of X and Y, denoted $\mathcal{S}^K_{XY}$, consists of all sets $\mathbf{S} \subseteq \mathbf{V} \setminus \{X, Y\}$ such that X and Y are d-separated by $\mathbf{S}$ in G.

2. A nonempty subset of the complete d-separating collection will be called a **partial d-separating collection**, often written $\mathcal{S}_{XY}$.

3. A partial d-separating collection consists of all d-separating subsets of neighbor$(X) \cup$ neighbor$(Y) \setminus \{X, Y\}$ is called the **standard d-separating collection**, denoted $\mathcal{S}^S_{XY}$.

Relation among a complete d-separating collection, a standard d-separating collection, and an (arbitrary) partial d-separating collection is shown in Figure 6.3.

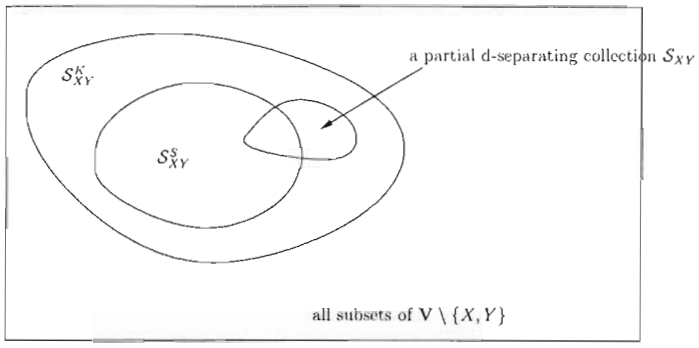


Figure 6.3: The relations among the complete d-separating collection ($\mathcal{S}^K_{XY}$), the standard d-separating collection ($\mathcal{S}^S_{XY}$), and an (arbitrary) partial d-separating collection ($\mathcal{S}_{XY}$ in the figure).

6.3.1 Constraint-based Learning Algorithms

We shall outline the PC algorithm. Irrelevant details of the algorithms are omitted so that the main problem can be revealed.

The PC Algorithm

The PC algorithm is considered by many the most popular constraint-based causal discovery algorithm; the algorithm has been implemented in Tetrad [77] and the Bayes Net Toolbox for Matlab [69]. A full description can be found at [93, page 84-85]. A brief outline of the PC algorithm is as follows.

Inputs: A sample from a distribution generated by the Bayes net structure.

Outputs: A Bayes net pattern.

Stage 1: Fast Adjacency Search.

1. Start with a complete undirected graph G over the set V of variables.
2. Repeat the following step: For each edge $X - Y$ in G, estimate a partial d-separating collection $\hat{\mathcal{S}}_{XY}$ using conditional independence test. The estimation is designed such that $\hat{\mathcal{S}}_{XY}$ either contains one d-separating set or none.
3. If $\hat{\mathcal{S}}_{XY}$ is nonempty, remove the edge $X - Y$ from G.

The justification for this stage is that a nonempty partial d-separating collection $\mathcal{S}_{XY}$ of X, Y implies non-adjacency between X and Y [82, Proposition 1].

The resulting graph G is undirected and called a **skeleton**. At the same time, we obtain a singleton estimation $\hat{\mathcal{S}}_{XY}$ for each pair of nonadjacent nodes X, Y in G.

Stage 2: Collider Identification. For each unshielded triple $\langle X, Y, Z \rangle$ in the skeleton, direct the path $X - Y - Z$ as $X \to Y \leftarrow Z$ if Y is contained in the only set in $\hat{\mathcal{S}}_{XZ}$.

The justification for this stage is that, with the faithfulness assumption, a d-separating set must contain Y if it is not a collider on $X - Y - Z$, and it must not contain Y if it is not a collider on $X - Y - Z$ (see Definition 5.2.2).

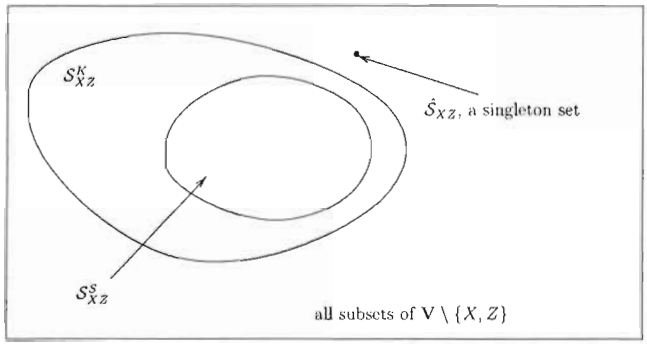


Figure 6.4: Estimation of a singleton d-separating collection by the PC algorithm $\hat{\mathcal{S}}_{XZ}$. Since it has only one element, noise may cause it to fall completely out of $\mathcal{S}^K_{XZ}$.

For the PC algorithm, Stage 1 (fast adjacency search) affects the skeleton of the final graph, whereas Stage 2 affects the direction of arrows in the resulting DAG patterns.

6.3.2 Underfitting with the PC Algorithm

The performance of Stage 1 in the PC algorithm is directly related to the accuracy of partial d-separating collections $\hat{\mathcal{S}}_{XY}$. Since $\hat{\mathcal{S}}_{XY}$ contains only one element, it matters only whether that element of $\hat{\mathcal{S}}_{XY}$ falls in the complete d-separating collection $\mathcal{S}^K_{XY}$ (see Figure 6.4).

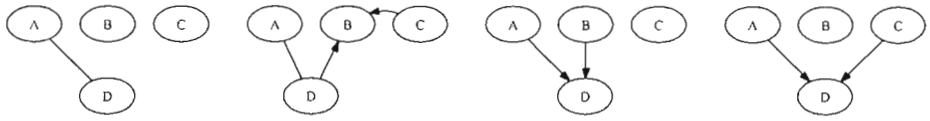
The PC algorithm uses Fisherian significance test of conditional independence to estimate the d-separating collection $\hat{\mathcal{S}}_{XY}$. Let X, Y be two nonadjacent nodes in the resulting skeleton and d be the training sample. The d-separating collection estimate $\hat{\mathcal{S}}_{XY}$ is generated by as follows:

$\hat{\mathcal{S}}_{XY} \leftarrow \emptyset$.
for all $S \subseteq \text{neighbor}(X) \cup \text{neighbor}(Y)$ **do**
  Let the null hypothesis H_0 be $X \perp\!\!\!\perp Y|S$.
  Set the significance level α with a small number (e.g., 0.05 in Tetrad [77]).
  Calculate a test statistic and its P-value p.
  if $p > \alpha$ **then**
    Add S into $\hat{\mathcal{S}}_{XY}$ (accepting $X \perp\!\!\!\perp Y|S$) and exit.
  end if
end for

Fisherian significance test To understand why the above procedure leads to inaccurate estimation of $\mathcal{S}_{XY}$, we briefly review Fisherian significance test.

In a significance test, a distribution of a test statistics (e.g., X^2 in χ^2-test of conditional independence) exists assuming the null hypothesis. If the value of the statistics on a sample is extreme under the distribution, then "we shall say that in this case the deviations from expectation are clearly significant" (quote from [30, Chapter 4]). In the context of conditional independence test, if the P-value goes under the significance level α that we have assigned beforehand, then the independence is very unlikely to be true. In other words, it happens only with probability at most α. On the other hand, the P-value being above the significance level α does not mean that the independence is likely to be true (for an in-depth discussion, see [90]). Therefore if we use the procedure previously described to estimate $\hat{\mathcal{S}}_{XY}$, it often falls off $\mathcal{S}^K_{XY}$ because we assume a null hypothesis to be true when we are really just not sure of the falsity of the opposite.

To summarize, unorthodox use of significance test in the PC algorithm leads to many incorrect independencies and hence missing edges (underfitting) in the constructed network.



(a) 50 data points (b) 100 data points (c) 150 data points (d) 200 data points

Figure 6.5: Structures returned by the PC algorithm on four random samples generated from the Bayesian network in Figure 6.1. The significance level used in the PC algorithm is 0.05.

Examples of Markov Condition Violation

We apply the PC algorithm over a sequence of samples generated the network in Figure 6.1. We observe similar results as in Figure 6.2; that is, none of the resulting networks is an I-map of the distribution.

For randomly generated networks and samples, results similar to GES+BDeu are shown in Figures 6.6, 6.7, 6.8, 6.9, 6.10, 6.11, 6.12, and 6.13. These figures clearly indicate the underfitting problem caused by the PC algorithm[3].

6.4 Summary

With the NP-hardness of exact implementation of fast mind change optimal learner, a natural approximation is to find *any one* minimal I-map consistent with the observed dependencies. This requires a learner to be consistent, that is, being an I-map of observed dependencies.

Score-based learning and constraint-based learning are two most common approaches for learning Bayesian network structures. Although they are based on different interpretation of Bayesian network structures, they share a common problem: underfitting the target distribution on small-to-medium samples. In other words, they do not always output an I-map of observed dependencies.

The following chapter will study a learning algorithm that combines the two approaches to approximate mind change optimality.

[3]In a boxplot, the middle black dot denotes the median, the box prescribes the upper and lower quantile (containing 50% or so data), and the dashed appendages encode adjacent values. For complete explanation of box plots, see [96].

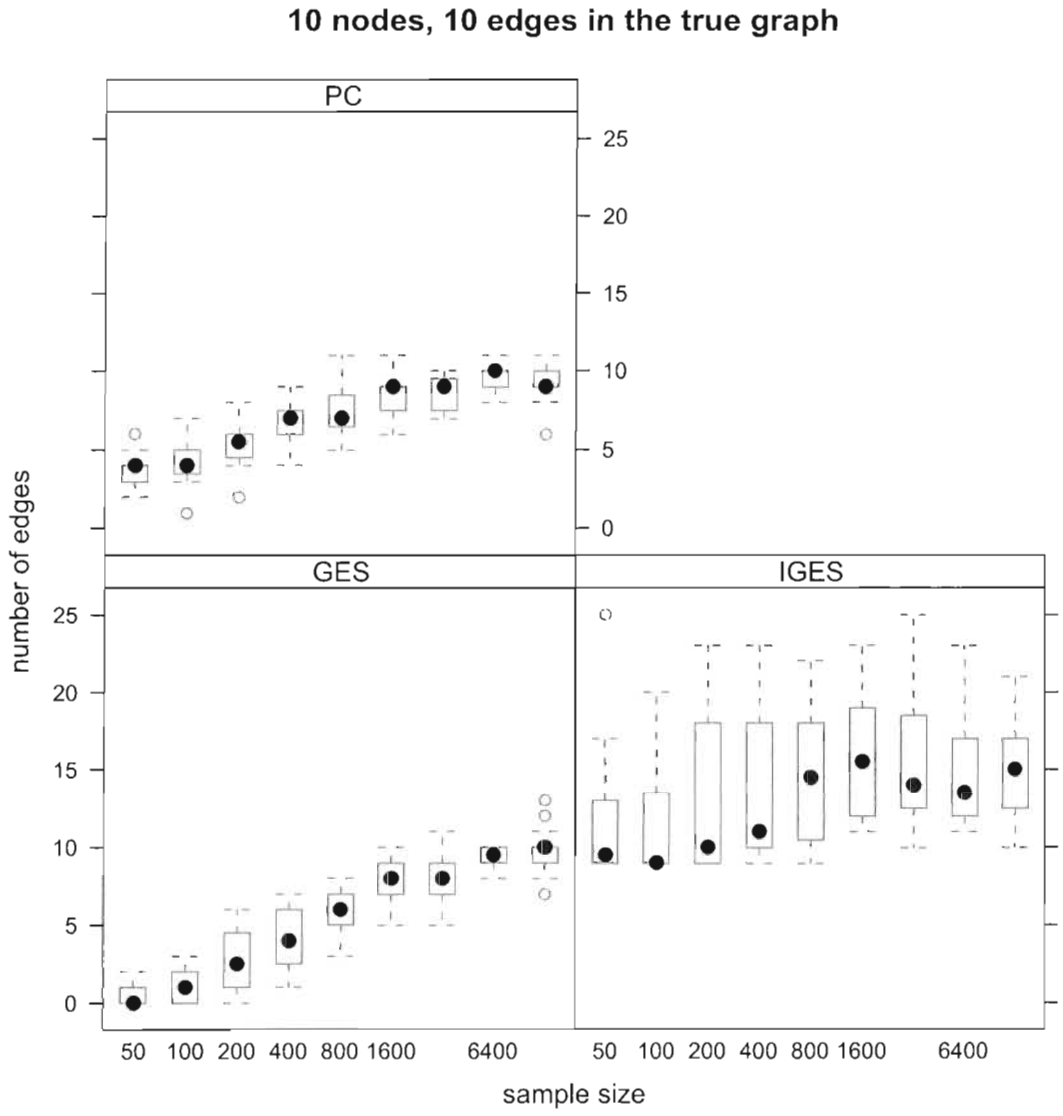


Figure 6.6: Number of edges in resulting patterns when true patterns have 10 edges. For each predefined sample size, up to 10 Random DAGs are generated over 10 nodes. When the sample sizes are small, the GES algorithm and the PC algorithm produce far fewer edges compared to the true graph. The IGES algorithm, in contrast, reduces the underfitting problem.

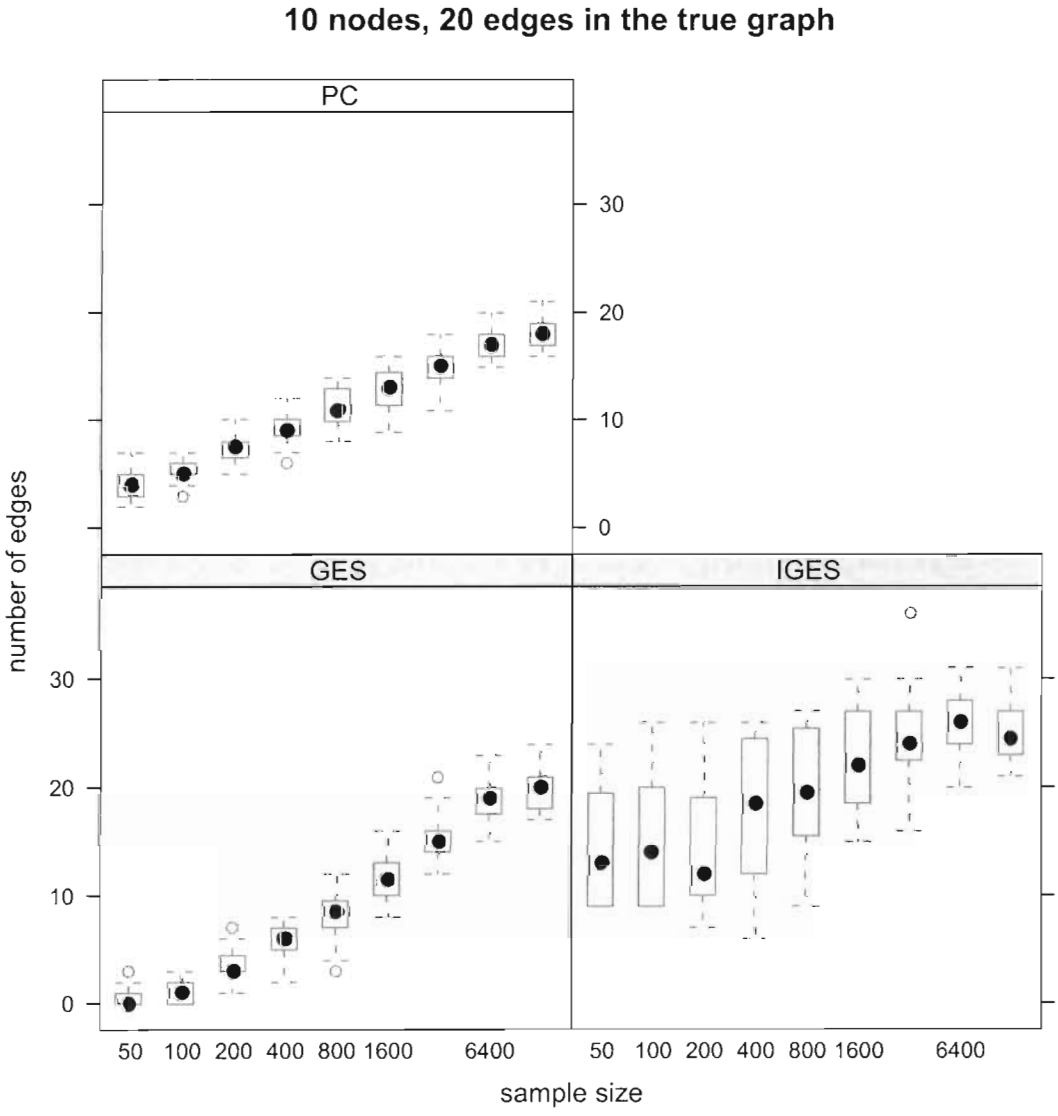


Figure 6.7: Number of edges in resulting patterns when true patterns have 20 edges. For each predefined sample size, up to 10 Random DAGs are generated over 10 nodes. When the sample sizes are small, the GES algorithm and the PC algorithm produce far fewer edges compared to the true graph. The IGES algorithm, in contrast, reduces the underfitting problem.

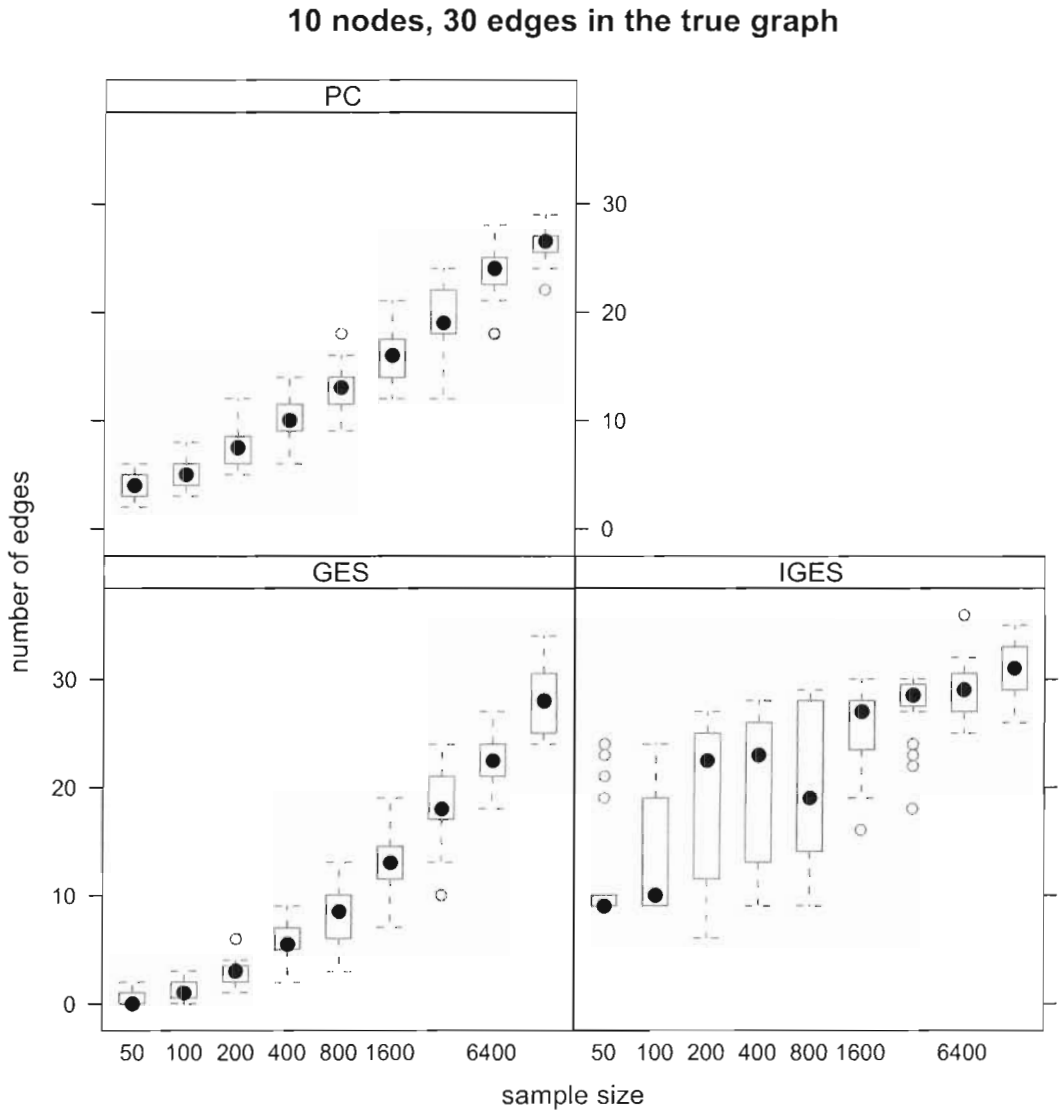


Figure 6.8: Number of edges in resulting patterns when true patterns have 30 edges. For each predefined sample size, up to 10 Random DAGs are generated over 10 nodes. When the sample sizes are small, the GES algorithm and the PC algorithm produce far fewer edges compared to the true graph. The IGES algorithm, in contrast, reduces the underfitting problem.

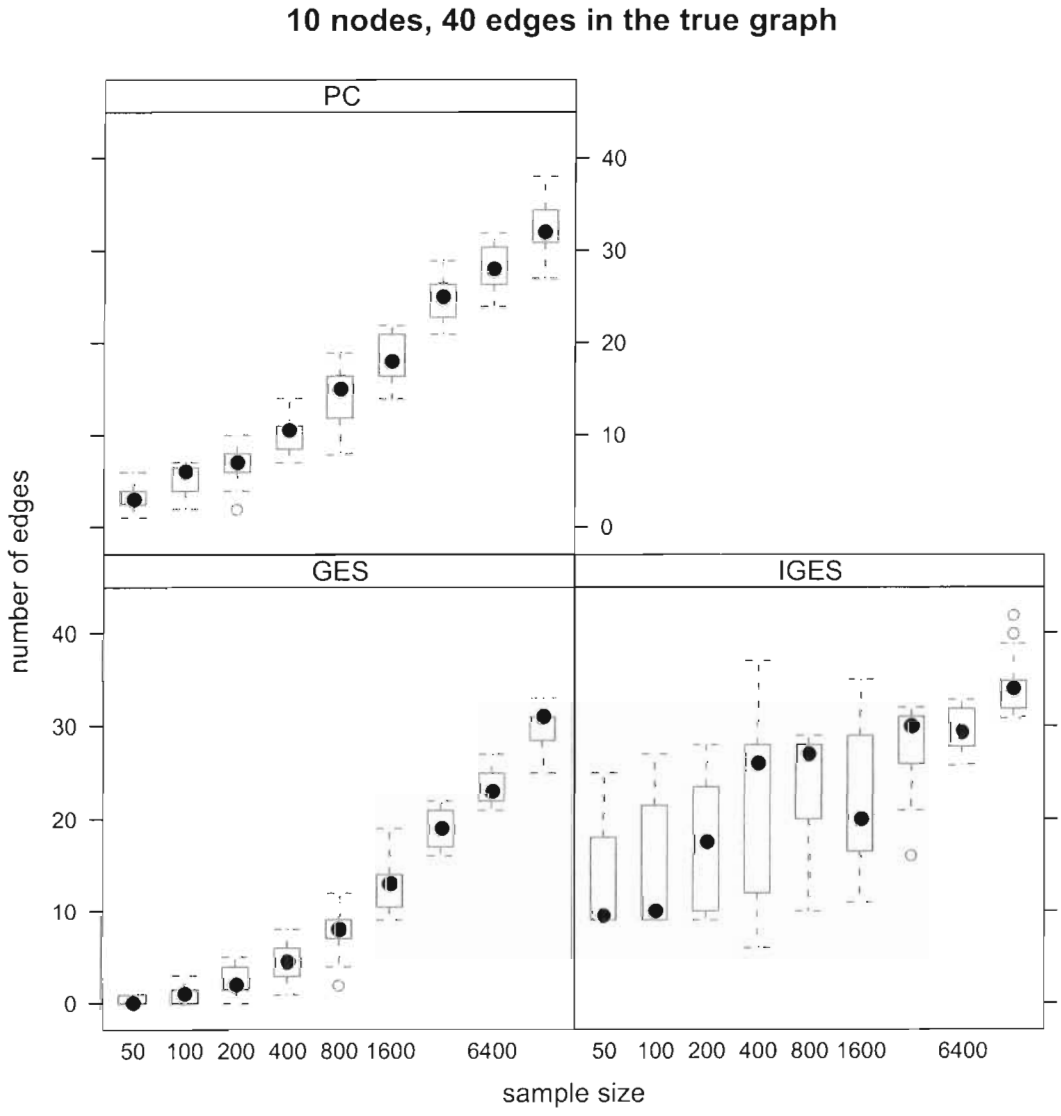


Figure 6.9: Number of edges in resulting patterns when true patterns have 40 edges. For each predefined sample size, up to 10 Random DAGs are generated over 10 nodes. When the sample sizes are small, the GES algorithm and the PC algorithm produce far fewer edges compared to the true graph. The IGES algorithm, in contrast, reduces the underfitting problem.

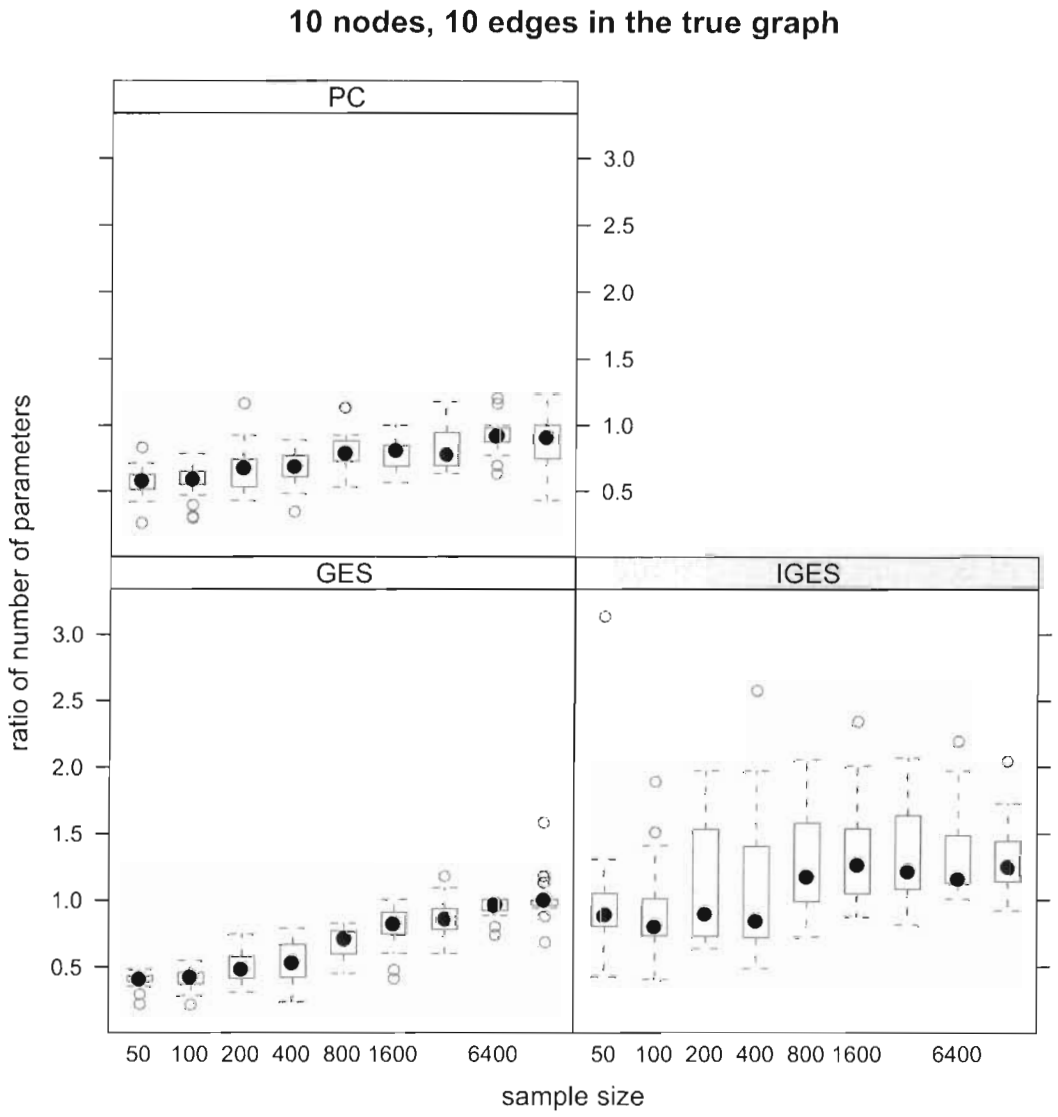


Figure 6.10: Ratio of the number of parameters in resulting patterns compared to the true number of parameters. For each predefined sample size, up to 10 random target patterns are generated. Target patterns have 10 edges over 10 nodes.

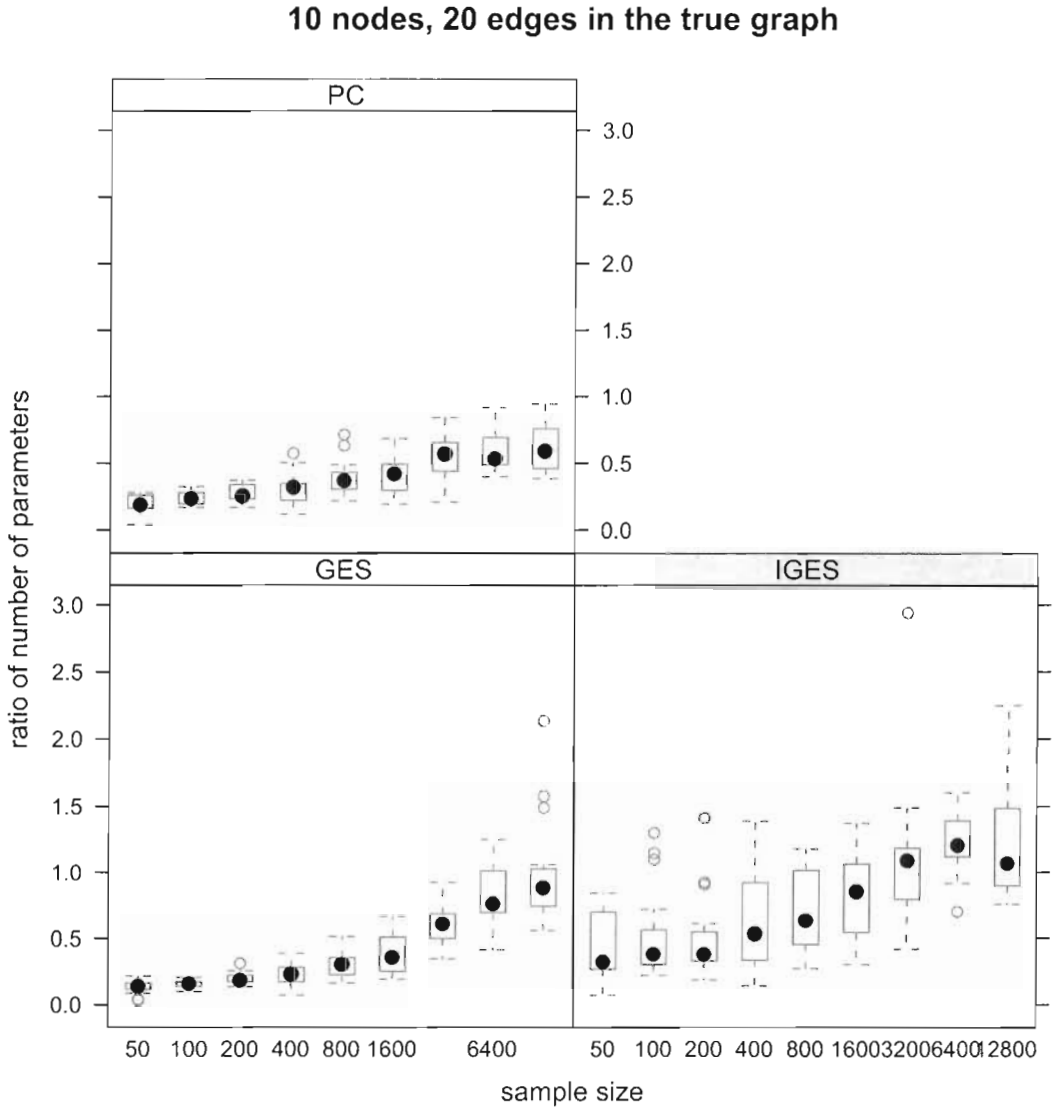


Figure 6.11: Ratio of the number of parameters in resulting patterns compared to the true number of parameters. For each predefined sample size, up to 10 random target patterns are generated. Target patterns have 20 edges over 10 nodes.

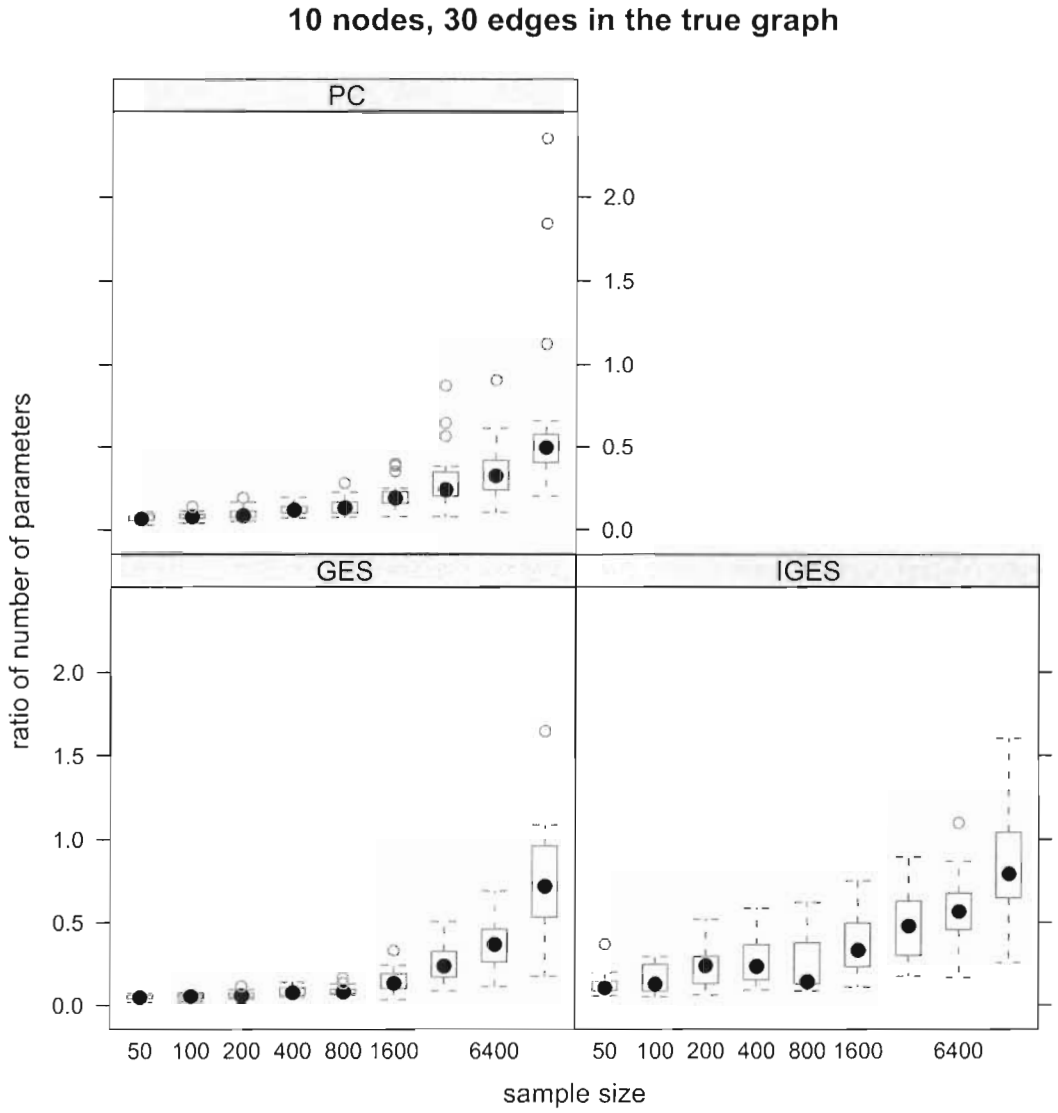


Figure 6.12: Ratio of the number of parameters in resulting patterns compared to the true number of parameters. For each predefined sample size, up to 10 random target patterns are generated. Target patterns have 30 edges over 10 nodes.

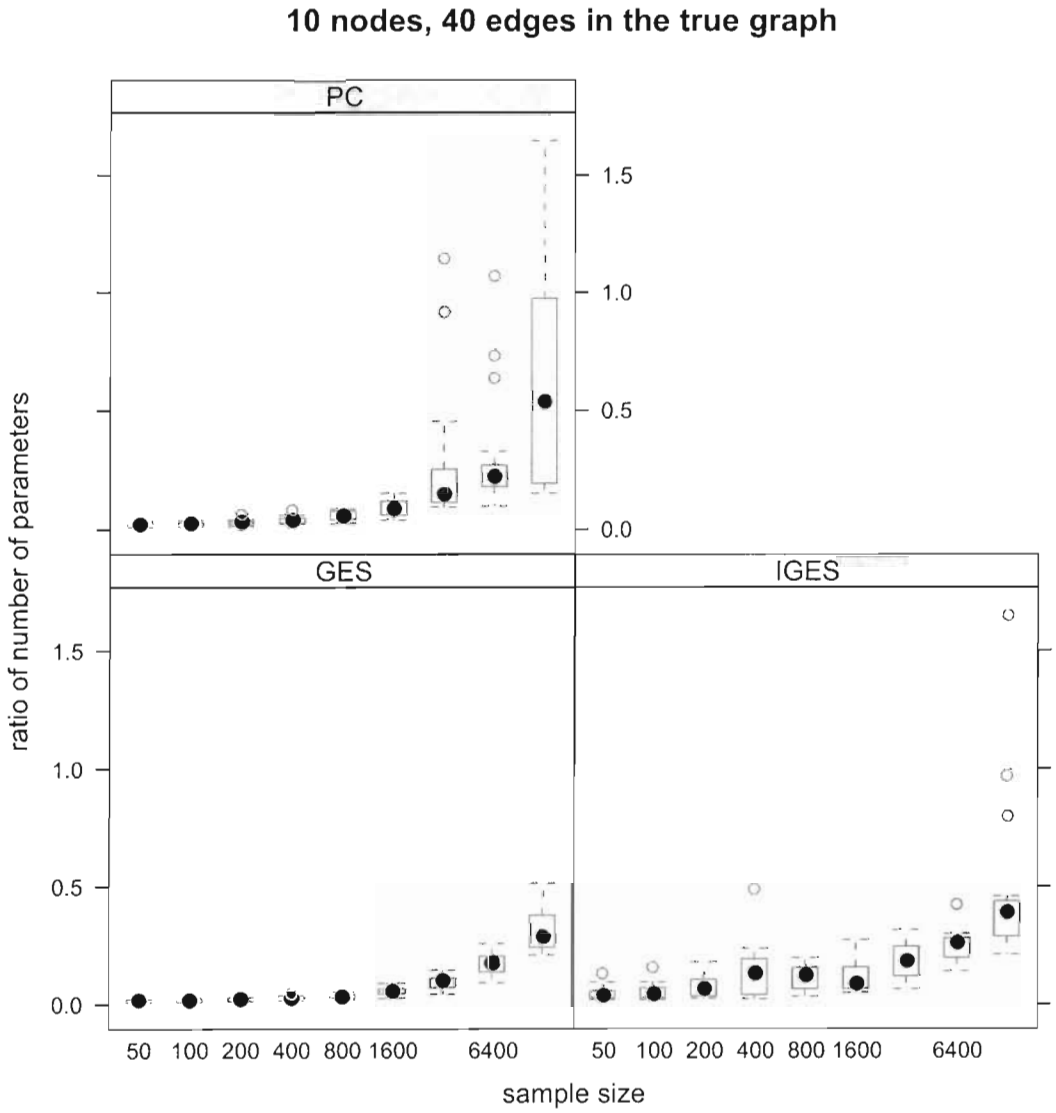


Figure 6.13: Ratio of the number of parameters in resulting patterns compared to the true number of parameters. For each predefined sample size, up to 10 random target patterns are generated. Target patterns have 40 edges over 10 nodes.

Chapter 7

Learning Bayes Nets by Adding Dependency Constraints

> All of these ways are usually too weird for normal use, but useful only on occasion. That's another way to say that they are the perfect hack.
>
> from *Perl Hacks*

In the previous chapters, we have seen that given a sample of small-to-medium size, both constraint-based methods and score-based methods may output a network that violates the Markov condition. Hence they neither incidentally nor accidentally get close to being a candidate for a mind change optimal Bayes net learner. However, by leveraging the ideas from the two methods, we reach a way to apply the principle of I-map learning. In this chapter, we propose a new criterion that combines information from statistical tests with a score-based search; such methods were termed "hybrid" by [36, p.50]. The basic idea is to combine conditional dependency constraints and a score function into a single criterion for evaluating a Bayes net structure G given sample d: maximize the score $S(G, \mathrm{d})$ given the constraint that G must satisfy the dependencies detected by a suitable statistical test. For the ease of comparison, we shall adopt more terms from machine learning than ones from learning theory. We hope the trivial details will not blur the central idea of the method, which is to emulate a mind change optimal Bayes net learner.

As we have seen, in Bayes net theory, a graph structure that entails a set of dependencies

$\mathcal{D}$ is called an I-map for $\mathcal{D}$. Thus our hybrid criterion corresponds to the constrained optimization problem $\max\{S(G, \mathrm{d}) : G$ is an I-map of conditional dependencies $\mathcal{D}\}$. We have shown evidence that GES+DBeu often underfits the true Bayes net structure. Several previous studies have also suggested the tendency of many score-based methods to learn graphs that are sparser than the target structure [27, 1, 19]. The motivation for adding dependency constraints is to correct the tendency of these score-based methods towards underfitting the data with overly sparse models. We also found that dependency information helps the learner to correctly orient edges; as shown later.

We need to address two main issues to apply our hybrid learning criterion: (1) choosing a statistical test for identifying the set of dependencies $\mathcal{D}$ appropriate for a given data sample d, and (2) computing a Bayes net structure that optimizes (at least approximately) the score given the dependencies $\mathcal{D}$. In this chapter, we follow a simple exhaustive query strategy that applies a statistical test (here, χ^2) to all conditional independencies of the form $X \perp\!\!\!\perp Y | \mathbf{S}$, where X and Y are distinct variables, $\mathbf{S}$ is a set of variables disjoint from X and Y, and the size of $\mathbf{S}$ is bounded by a constant k. The constant k is a parameter of the system; our experiments indicate that even with a fairly small bound of $k = 3$, the system detects helpful dependency constraints, at reasonable computational cost. Note that our approach relies only on *dependencies* detected by the statistical test, not independencies. In statistical tests of independence, the null hypothesis is independence, so we follow the test when it rejects the null hypothesis. We do not, however, accept the null hypothesis when the test fails to reject. To motivate this, observe that (at small to medium sample sizes) a rejection is a quite reliable indicator that the null hypothesis is false, but failure to reject is a less reliable indicator that the null hypothesis is true. Since we treat the output of the statistical test as a constraint for the score-based search, it is important that this information is correct.

Once a set of conditional dependencies is found, the next question is how to apply the score criterion computationally to guide a learning algorithm. As computing a Bayes net structure that optimizes a given score function is difficult [21], it is necessary to use a local search heuristic. We provide a general schema for adapting any local search algorithm to perform this constrained search with dependencies. Adapting a local search procedure for constrained score optimization raises two main issues: 2a) carrying out the local constrained search, and 2b) choosing an initial graph as the starting point. For our implementation, we

chose to adapt the state-of-the-art GES search procedure [65, 19] for constrained optimization; we refer to the resulting procedure as IGES (for "I-map + GES"). Under mild standard assumptions, GES and IGES provably converge to the same output in the sample size limit when IGES is given correct dependency information. The GES procedure searches in two phases over patterns that represent equivalence classes of directed acyclic graphs (DAGs). In the first phase, the search expands an initial sparse pattern until it reaches a local maximum in the score function. In the second phase, the GES search prunes the pattern through edge deletions until it reaches a local maximum. Our adaptation of GES search continues expanding the pattern in the first phase until it arrives at an I-map of the dependencies $\mathcal{D}$. (Thus the difference is that if a pattern P is not an I-map of $\mathcal{D}$, then IGES may add an edge to P even if this operation moves the search to a lower scoring pattern.) The IGES procedure has the same pruning phase as GES except that the search considers only patterns that are I-maps of $\mathcal{D}$. For the starting point of the constrained search, we use the output of the unconstrained search — because the starting pattern tends to be close to a constrained local optimum, this leads to a fast search. That is, we view the constrained search as a post-processor for the unconstrained search. The main effect of our post-processing is to expand an insufficiently complex model by fitting the additional dependency constraints observed in the data. In a sense, this is dual to common post-processing procedures in Machine Learning where an overly complex structure is pruned. Thus I-map learning is implemented as separate module in addition to the original score-based search. Experiments have shown that on small to medium sample sizes, maximizing the BDeu score and related scores tends to select Bayes net structures that are sparser than the target graph generating the data. For such structures, the main effect of our post-processing is to expand an insufficiently complex model by fitting the additional dependency constraints observed in the data. This contrasts with common post-processing procedures in Machine Learning where an overly complex structures is pruned. Figure 7.1 shows the components of our algorithm.

For evaluation, we performed a number of experiments comparing GES search based on the well-established BDeu score function [40] with and without dependency constraints. We present simulation results where the target graphs are randomly generated Bayes net structures and compare the graphs learned with and without dependency constraints to the target graph.

These experiments illustrate how, for small to medium sample sizes, adding dependency

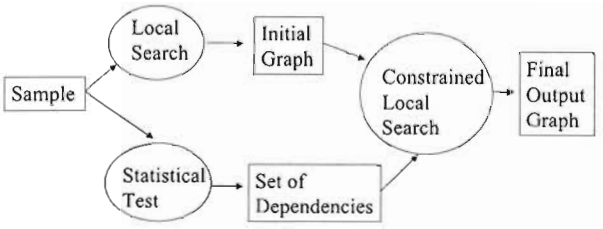


Figure 7.1: Constrained Optimization as Postprocessing. A local search procedure applied to a sample produces an initial graph. A statistical significance test is used to produce a set of conditional dependencies. The local search procedure, constrained to satisfy these dependencies, outputs a final graph.

constraints corrects some of the underfitting tendency of parameter-count based score functions. In order to fit the observed statistically significant dependencies, constrained learning tends to add adjacencies and arrowheads missing from the unconstrained search model. (Arrowheads may be missing when we search for pattern graphs; see Sections 2.1 and 7.3.) In our experiments, the constrained search model has lower KL-divergence than the unconstrained search model, which is evidence that the additional structure is important for the distribution defined by the target model.

7.1 Related Work

There are many constraint-based algorithms that employ statistical tests to reconstruct Bayes net graph structure [93, 61, 18]. Many of these methods use the "single link deletion" strategy [106]: if a significance test does not reject a null hypothesis $X \perp\!\!\!\perp Y|\mathbf{S}$, infer a conditional independence and mark variables X and Y as nonadjacent. As we do not infer independence from failure to reject, our approach does not rely on the single edge deletion strategy. Many statisticians recommend against inferring the truth of the null hypothesis when the null hypothesis is not rejected [41]; our use of statistical tests follows this recommendation and is more conservative than the use of tests in previous constraint based algorithms. For more discussion of independence testing in constraint based algorithms, see [70, p.593], [93, Sec.5.6]. There are several previous Bayes net learning algorithms that may be classified as hybrid algorithms (e.g., [27, 32]). While these algorithms do consider statistical measures (*e.g.*, mutual information), they do not incorporate the outcome of

a statistical test as a constraint that the learned model must satisfy. Combining such constraints with score-based search is based on the principle of I-map learning, and is a novel feature of our system. A number of previous studies have observed the tendency of score-based methods to learn graphs that are sparser than the target structure [27, 1, 19]. Our experimental methodology mainly follows that of previous investigations.

Several theoretical results have been established about the computational complexity of learning Bayes nets that satisfy a given set of dependencies. Bouckaert shows that finding an edge-minimal or parameter-minimal I-map for a given dependence oracle is NP-hard [13, Lemma. 4.5]. Our previous work [90] has shown that learning unique I-map given dependencies is also NP-hard. To our knowledge, there has been no work on heuristic algorithms for computing an edge-minimal I-map for a given set of dependencies. Chickering et al. prove that minimizing a score is NP-hard even if the learner has access to an independence oracle [21]. This problem specification is different from constrained optimization because the goal is not to use dependency information as an aid to finding a graph that optimizes the score with respect to all possible network structures, but to find a graph that optimizes the score with respect to the network structures that satisfy the dependencies. Moreover, our motivation for employing information about dependencies is not to speed up the computation of a graph with an optimal score, but to improve the learned model (see Figure 7.2).

7.2 Algorithm Design for Constrained Score Optimization

This section discusses the major design choices in our system, following the architecture outline of Figure 7.1.

7.2.1 Use of Statistical Tests for Detecting Conditional Dependencies

I-map learning requires a statistical significance test for conditional independence hypotheses of the form $X \perp\!\!\!\perp Y|\mathbf{S}$. As with constraint based methods, the test can be chosen to suit the type of available data and application domain. We used the traditional χ^2 test for categorical data. Since I-map learning treats the results of the statistical test as "hard" constraints, it is important that the decisions of the test be as reliable as possible, even on small to medium sample sizes. To this end, our system follows two principles for applying the significance test. (1) Take "rejection of the independence null hypothesis" as indicating

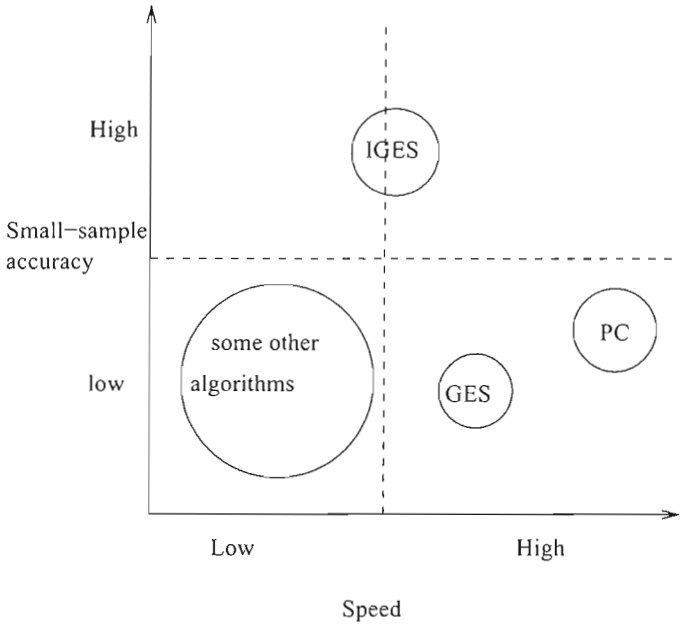


Figure 7.2: Relation of algorithms in terms of speed and accuracy with small-to-median samples. Both the GES and the PC algorithm run fast, but output incorrect networks. Depending on the number of constraints added, the IGES algorithm may achieve improve accuracy, at the cost of extra running time.

dependencies, but draw no conclusion from failure to reject.

The significance level controls a trade-off between reliability and informativeness: a lower significance level implies more reliable rejections of the null hypothesis, but also fewer conclusions to guide the search (see Figure 7.4). We follow previous constraint based methods [17] and use a fixed significance level for statistical testing; our experiments set a standard significance level of 5%. (2) Require a minimum sample coverage for the χ^2 test: the number of samples in each cell C_i must be at least $m \times p_i$, where p_i is the probability of cell C_i according to the null hypothesis, and m is the pre-assigned minimum sample size. This suggests that the χ^2 distribution should be a reliable approximation to the distribution of the test statistic [28, Ch.9.1]. If the sample coverage condition is not met, we draw no conclusion from the outcome of the test. (A similar sample coverage condition for the G^2 test is discussed in [93, Section 5.5]. The authors, however, assume dependency when the conditional is not met.)

If the number of variables is small, it is feasible to exhaustively apply the statistical

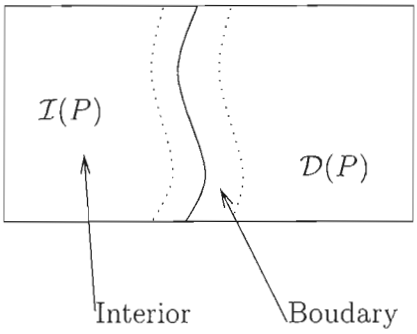


Figure 7.3: Statement space of distribution P consists of $\mathcal{I}(P)$ and $\mathcal{D}(P)$. Given a small sample, the membership of some statements are more uncertain than others. In particular, the larger the conditioning set, the more uncertain its membership is. Therefore, the estimation of the boundary is statistically unstable.

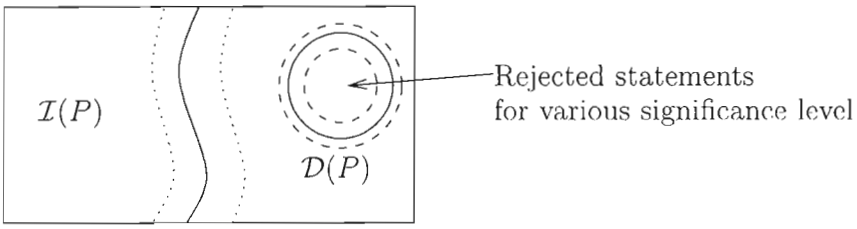


Figure 7.4: Rejections by independence test provide a reliable subset of dependencies. The lower the significance level of each test, the smaller but the more reliable the subset of $\mathcal{D}(P)$ is obtained.

test to all possible conditional independence statements. This corresponds to a procedure known as "all subsets variable selection" in model selection. The exhaustive testing strategy quickly becomes infeasible as the number of variables grows [107, p.59]. We use two restrictions: (1) Test only pairwise conditional independence statements of the form $X \perp\!\!\!\perp Y|\mathbf{S}$ for single variables X and Y. Testing only pairwise independence statements can be theoretically justified if we assume the composition property (Section 2.1), which implies that independence $\mathbf{X} \perp\!\!\!\perp \mathbf{Y}|\mathbf{S}$ holds if and only if $X \perp\!\!\!\perp Y|\mathbf{S}$ for each pair of variables $X \in \mathbf{X}$ and $Y \in \mathbf{Y}$. Chickering and Meek [22] shows that the composition property holds for a large class of probability distributions. (2) Exhaustively test the pairwise conditional independence statements, but only for conditioning sets $\mathbf{S}$ whose size is bounded by a constant k. In our experiments, we set the bound $k = 3$. One advantage to generating the dependency constraints independently of a local search strategy is that they serve to check errors made

by the heuristic search. Our simulations provide evidence that even this limited correlation analysis yields constraints that improve the quality of the model learned in score-based search. As a side benefit, the statistically significant conditional correlations between variables are often in themselves of interest to a user. Other possible strategies for choosing a set of conditional independencies to test include the following:

1. Statistical Queries From Constraint-Based Methods: constraint based methods perform a sequence of statistical tests, so one possibility is to run a constraint based algorithm on the sample first and then apply constrained optimization with the set of dependencies detected by the constraint based method. Some researchers have applying a constraint based method to obtain an initial graph G, which serves as the starting point for a score-based search [94]. With this approach, it is natural to apply constrained optimization such that the score-based search not only starts with the output of the constraint based method, but also utilizes the conditional dependencies discovered by the constraint based method's statistical tests as constraints for score optimization. Our experiments so far do not indicate that this combination of constraint based and score-based methods leads to better result than our simpler approach, but further investigation seems warranted.

2. Search-Based Hypothesis Choice: Several hybrid methods compute statistics such as conditional mutual information to measure the strength of association between two variables; the statistics to be computed are based on the graph constructed at the current state of the local search [27] This approach can be adapted for I-map learning. For example, suppose the current graph G contains an edge $A - B$ and the search algorithm considers deleting the edge to move to a graph G'. Let M be the Markov blanket of A in G' and suppose that $B \notin M$. Then if a test indicates that $A \not\perp B|M$, a dependency not covered in G', we could direct the search not to delete the edge $A - B$. An attractive feature of constrained optimization on this approach is that although the decision to test a certain conditional independence is made locally depending on the state of the search, if a dependency is detected, it becomes part of a global cache that constrains the search at all future stages. A disadvantage of search-based hypothesis choice is that errors made by the local search algorithm may propagate to choosing less than optimal dependencies to test. In contrast, if the dependency constraints are generated independently of the local search, they can serve to check its errors.

From the perspective of classical statistics, statistical tests should be done without any knowledge about the samples. Since the above two approaches dynamically select which hypotheses to test based (directly or indirectly) on the data observed so far, they raise the difficulty of interpreting the test results.

7.2.2 Heuristic Search Algorithm with Dependency Constraints

Assume that a set of statistically significant dependencies $\mathcal{D}$ of the form $X \not\perp Y|\mathbf{S}$ have been found by the procedures of the previous section, and that a sample d is given. We describe a general schema for adapting any local hill-climbing search procedure L with score function $score(G, \mathrm{d})$ to perform constrained optimization of the score that honors the set of dependencies $\mathcal{D}$. We refer to the constrained version of the L search procedure as IL search (for I-map + L). If the current state of the search is a graph G, a local search procedure L moves to the highest scoring graph G' in a neighborhood $\mathsf{nbdh}(G)$ provided that $score(G', \mathrm{d}) > score(G, \mathrm{d})$. The *neighborhood constrained by dependencies* $\mathcal{D}$ is defined as follows. A graph G' is a member of $\mathsf{nbdh}^{\mathcal{D}}(G)$ if

1. $G' \in \mathsf{nbdh}(G)$ and $(\mathcal{D}(G') \cap \mathcal{D}) \supseteq (\mathcal{D}(G) \cap \mathcal{D})$, and
2. $score(G', \mathrm{d}) > score(G, \mathrm{d})$ or
   $(\mathcal{D}(G') \cap \mathcal{D}) \supset (\mathcal{D}(G) \cap \mathcal{D})$.

The first clause requires that a candidate graph G' for constrained optimization must be a candidate graph in the original search space, and that it must cover at least as many of the given dependencies $\mathcal{D}$ as the current graph G. The second clause stipulates that a candidate graph G' must make progress, in that G' has a higher score or covers more of the given dependencies. From a current graph G, the IL search moves to the neighboring candidate graph $G' \in \mathsf{nbdh}^{\mathcal{D}}(G)$ with maximum score. Note that IL search may move to a graph with lower score G' if G' covers more dependencies and all the neighbors of G have a lower score than G. The IL search terminates with graph G when there are no more candidate graphs, that is, when $\mathsf{nbdh}^{\mathcal{D}}(G) = \emptyset$. Given the modified definition of neighborhood, this schema can be extended in an obvious way to local search strategies more complex than hill climbing.

The next observation asserts that adapted local search finds a local optimum for our hybrid criterion, provided that the basic operations of the local search procedure make it

possible to reach an I-map for any set of dependencies $\mathcal{D}$. This is the case if single edge addition is one of the local operations; all local search algorithms that we know consider single edge additions.

Observation 3. *Let L be a local search procedure for score S that has single edge addition as one of its basic operations. Then on any sample d, the constrained local search IL with dependencies $\mathcal{D}$ terminates with a local score optimum G that is an I-map for $\mathcal{D}$. That is, if G' is a neighbor in* $\mathsf{nbdh}(G)$*, then G' is not an I-map of $\mathcal{D}$, or $score(G', d) \leq score(G, d)$.*

Proof. If a current graph G does not cover the dependencies $\mathcal{D}$, the constrained IL search can apply an operation (*e.g.*, add an edge to G) to increase the set of covered dependencies $\mathcal{D}$, so Clause 2 applies. So the search covers more and more of the dependencies in $\mathcal{D}$ until it reaches an I-map of $\mathcal{D}$. Then Clause 1 implies that all subsequent candidates are I-maps of $\mathcal{D}$, and the search continues until no neighboring I-map of $\mathcal{D}$ has a higher score. Then it terminates with a local score optimum given $\mathcal{D}$. □

For our system we adapt the GES (Graph Equivalence Search) local search algorithm. GES is a state-of-the-art Bayes net search strategy that satisfies optimality guarantees in the large sample limit and has been extensively evaluated [19]. The general conclusion from experimental evaluation is that GES performs very well in finds high-scoring high-quality graph models (albeit at increased computation time), arguably better than any other standard deterministic local search algorithm. Since our goal is to investigate whether adding dependency constraints improves the quality of learned models, we want to employ a high-quality score-based method such as GES. We describe GES only in sufficient detail to indicate how we adapt GES search; for a full description see [19]. The GES algorithm searches the space of patterns in two phases. During the growth phase, GES adds an edge to a current pattern π, subject to several conditions, until reaching a local score maximum. The growth phase terminates with a pattern π if no valid edge addition to π increases the score. The definition of d-separation implies that adding an edge to a pattern π leads to a pattern π' that covers strictly more conditional dependencies. So during the growth phase of the GES algorithm, the set of covered dependencies increases monotonically. During the subsequent shrink phase, GES deletes an edge from a current pattern π, subject to several conditions, until reaching a local score maximum. GES is particularly natural for I-map learning because the second part of Condition 1 of IL search is always satisfied during its growth phase. Applying the IL schema defines the constrained IGES, which continues

adding edges to a pattern until it reaches an I-map of $\mathcal{D}$ and deletes edges only if the result is still an I-map of $\mathcal{D}$. The algorithm is as follows:

Inputs: A sample from a distribution generated by the Bayes net structure.

Outputs: A Bayes net pattern.

Dependencies Generation: Generate a set of dependencies $\mathcal{D}$ using statistical testing.

Growth Phase:

1. Start with the Bayes net pattern π over the set V of variables with no edges.
2. Repeat the following step: replace π with maximum-score pattern $\pi' \in \mathsf{nbdh}^{+}(\pi)$ if either $score(\pi') > score(\pi)$ or $\mathcal{D}(\pi') \cap \mathcal{D} \supset \mathcal{D}(\pi) \cap \mathcal{D}$.
3. Go to the shrink phase.

Shrink Phase:

1. Repeat the following step: replace π with maximum-score pattern $\pi' \in \mathsf{nbdh}^{-}(\pi)$ if $score(\pi') > score(\pi)$ and $\mathcal{D}(\pi') \supseteq \mathcal{D}$.
2. Output the pattern when the search stops.

The next observation relates the outputs of GES and IGES.

Observation 4. *Let P be any joint probability distribution over the variables $\mathbf{V}$. Suppose that $\mathcal{D}$ is a set of conditional dependencies that hold in P and let d be a given sample.*

1. *If the growth phase of GES terminates with an I-map π of P, then the growth phase of IGES with dependencies $\mathcal{D}$ terminates with the same pattern π.*
2. *If the shrink phase of GES terminates with an I-map π of P, then the shrink phase of IGES with dependencies $\mathcal{D}$ terminates with the same pattern π.*

Proof. Clause 1: Suppose that on some sample d, the growth phase of the GES produces a sequence $\pi_1, .., \pi$ of patterns where π is an I-map of P. Then each pattern π_{i+1} is a score-maximizing neighbor of π_i, so IGES also reaches π. Since the pattern π is an I-map of P, it is also an I-map of $\mathcal{D}$, so the IGES terminates with π, as required. Clause 2: Suppose that on some sample d, the shrink phase of the GES goes through a sequence $\pi_1, .., \pi$ of

patterns where π is a parameter-optimal I-map of P. Suppose for contradiction that any of the patterns π_i is not an I-map of the dependencies $\mathcal{D}$. Since the shrink phase deletes edges, the set of covered dependencies monotonically decreases, and so no pattern $\pi_{i+1}, ..., \pi$ is an I-map of $\mathcal{D}$. But then π is not an I-map of P, which contradicts our supposition. This shows that all patterns in the sequence $\pi_1, .., \pi$ are I-maps of $\mathcal{D}$, and also local score-maximizers. So the IGES follows the sequence $\pi_1, .., \pi$, as required. □

Chickering [19] proved that if the scoring function is consistent, and the generating distribution P satisfies the composition property, then the following statements hold with probability 1 in the sample size limit: (1) the growth phase of GES yields an I-map of the generative distribution P, and (2) if P has a perfect I-map, and the shrink phase of GES starts with an I-map of P, then it terminates with the perfect I-map of P. [1] Observation 4 together with Chickering's results implies that we can expect IGES to behave like GES when the sample size is large. The motivation for taking into account statistically significant dependencies is that we expect fitting these dependencies to speed up convergence to a correct graph structure. The next section presents evidence from simulation studies validating this expectation.

7.3 Evaluation

Our experiments investigate the BDeu score function. The BDeu score has a Bayesian theoretical foundation, and has been shown to be a competitive score for learning Bayes net models [40, 19] (see also Chapter 6.2). The BDeu score requires specifying a prior equivalent sample size and a structure prior. We followed [27] and used a uniform structure prior in order to maximize the impact of the data-driven likelihood term in the BDeu score and thereby minimize underfitting [27, Section 5]. In additional, we also used the default structure prior 0.001^k in Tetrad [77] and [19], where k is the number of parameters in that structure. For the convenience of exposition, we simply refer the two settings as the structure prior 1 and the structure prior 0.001. Our simulations examine models with binary

[1] The assumptions are that the score function is consistent and the generative distribution P satisfies the composition property. Chickering's theorems actually guarantee somewhat stronger properties of the result of GES than the ones we stated. We note that the convergence guarantees still hold if we replace the assumption that the statistical test returns only correct dependencies by the weaker assumption that the dependencies are correct with probability 1 in the sample size limit.

variables, as in [1]. The methodology follows previous studies of Bayes net learning such as [1, 94, 27, 82]. Our code is written in Java and uses many of the tools in the Tetrad package [77].

7.3.1 Evaluation Criteria

Following [27], We consider both topological criteria for the resulting DAGs and distributional criteria for the fitted model.

As in [27], [94], [82], we consider the following set of topological criteria for the resulting graph.

1. number of added edges(E^+),
2. number of removed edges (E^-),
3. number of added arrowheads (A^+),
4. number of removed arrowheads (A^-),
5. number of added unshielded colliders (C^+),
6. number of removed unshielded colliders (C^-).

To aid interpretation of the experimental results, we combine false positives and false negatives using the F-measure from information retrieval [104, p.146], which is defined as

$$\frac{2(\text{True Positive})}{2(\text{True Positive}) + (\text{False Positive}) + (\text{False Negative})}$$

.

As in other Bayes net learning studies (*e.g.*, [27, 1]), the distributional criterion considered is the Kullback-Leibler (KL) divergence of the fitted model to the true distribution [54]. Given an operating or target distribution f that generates the training sample, and a DAG G inferred from the sample, let $\hat{f}_G$ be the fitted distribution (with MLE estimation of parameters). Then the KL divergence of $\hat{f}_G$ to f is defined as $\text{KLD}(f, \hat{f}_G) = \mathbb{E}_f(\log f/\hat{f}_G)$ where $\mathbb{E}_f$ denotes the expectation with respect to distribution f. Our simulations use an exact method to compute KL divergence. Note that fitting a DAG with more edges or parameters does not necessarily define a distribution with smaller KL divergence, because the parameter estimation in a complex model often has large bias [107].

7.3.2 Simulation Results

The target models considered were randomly generated networks with number of nodes n varying from 4 to 10. We used Tetrad's random DAG generating functions `GraphUtils.createRandomDagC()` to build the networks with $3n$ or $\binom{n}{2}$ edges, whichever is smaller. For each graph, samples of various sizes (ranging from 100 to 12800) were drawn and presented to both the IGES algorithm and the GES algorithm with the BDeu score. To reduce the variance in performance measurement, we repeated the experiment 10 times, resulting in 10 random graphs for each combination of sample size and node count, and took the average of all measures. The sampling procedure is as follows.

for all n such that $4 \leq n \leq 10$ **do**
  for $i = 1$ to 10 **do**
    Generate a DAG with n nodes and $\min(3n, \binom{n}{2})$ edges.
    From the DAG, create a Bayes net G_i with binary variables and randomly generated conditional probability tables.
    Let $m = 100$.
    repeat
      Draw m random instantiations from G_i.
      Double the value of m (i.e., $m \leftarrow 2 \times m$).
    until $m > 20000$.
  end for
end for

Figures 7.5-7.13 show the KL-divergence and F-measures for adjacency, arrow and collider identification. On small-to-medium size training samples, the IGES algorithm produces structures with smaller KL-divergence and larger F-measures, which indicate a better agreement with the target graph structure compared to the GES algorithm. Table 7.1 shows typical values for topological measurements in the experimental results.

Our postprocessing approach runs GES twice, the second time with dependencies. A potential explanation of the learning improvement is that just iterating GES by itself leads to better results, independently of adding the dependency constraint. To test this hypothesis, we also compared a twice iterated GES to single GES and to GES with post-processing by IGES. The double-run GES sometimes gives different results from the single-run GES, but no clear improvement is discernible; however, GES with post-processing by IGES leads to

Alg	m	E^+	E^-	A^+	A^-	C^+	C^-
GES	100	2.4	19	6.8	20.3	5.5	17.5
IGES	100	3	17.3	7.7	19	6.2	17.3
GES	200	1.8	17.3	6.9	19.4	5.6	17.1
IGES	200	3.2	14	9.5	17.7	6.1	17.1
GES	400	1.2	13.9	8.5	17.7	6.3	16.6
IGES	400	3.1	10	10.9	16.4	7.1	16.3
GES	800	2.1	12.1	10.7	16.7	8.2	16.6
IGES	800	3.7	7.8	10	14.9	6.3	16.1
GES	1600	2.9	8	10.6	12.2	8.1	14.4
IGES	1600	3.6	5.3	7.8	12.1	6.2	14.3
GES	3200	3.8	6	15.8	14.1	12.3	15
IGES	3200	4	3.9	10.5	9.8	6.5	13.9
GES	6400	4.6	3.1	17.5	13.1	10.6	16.1
IGES	6400	3.9	3.5	10.5	9.1	6.4	12.6
GES	12800	5.9	2.3	18.4	12.6	13.7	16.2
IGES	12800	5.2	2.9	15.3	12.6	8.1	16
GES	25600	7	1.4	18.1	11.3	11.2	16
IGES	25600	7	1.2	15.2	10.5	9.8	15.1

Table 7.1: Measures of structural errors for the randomly generated graphs. Here $|V|$ denotes the number of variables, m the sample size; the other symbols are defined in the text. E^+ the number of added edges, E^- the number of removed edges, A^+ the number of added arrows, A^- the number of removed arrows, C^+ the number of added unshielded colliders, C^- the number of removed unshielded colliders. The IGES algorithm tends to have fewer false negatives, at the cost of more false positives. As the sample size increases, the IGES and the GES algorithm produce similar measures. For each sample size, 10 samples are generated.

an improvement over the double GES similar to the improvement over the single GES.

The run-time difference between GES and IGES is mainly due to the number of independence tests performed by IGES. The number of independence tests with n variables is $\sum_{i=0}^{k} \binom{n}{2} \cdot \binom{n-2}{i}$, which is bounded by $\frac{n^2(n-2)^{k-1}}{(k-1)!} = O(n^{k+1})$ [93, Ch.5.4.2.1]. For 10 variables, the runtime of the IGES and the GES algorithms on some typical sample sizes are shown in Table 7.2.

sample size	500	1000	5000
IGES	11.7	15.0	26.9
GES	0.3	0.6	2.3

Table 7.2: Runtime (in seconds) of the IGES and the GES algorithms on some random samples. The samples are drawn from a random network of 10 nodes.

For off-line analysis of a dataset, the increased run-time seems acceptable for the expected improvement in the quality of the learned model. There is considerable room to improve the efficiency of our current implementation for time-critical applications; for example, during the grow phase of the IGES algorithm, it is not necessary to perform a statistical test $X \perp\!\!\!\perp Y|\mathbf{S}$ if the current graph already contains an adjacency $X - Y$. The most important step towards a fast application of our hybrid criterion would be a query strategy for generating a feasibly small yet informative set of dependency constraints.

Our experiments provide a proof of concept that motivates further research into combining score functions with statistical tests. They show that even a very high quality search+score algorithm often fails to fit statistically significant dependencies on small-to-medium sample sizes, and that adding these dependencies as a constraint typically speeds up convergence to the target structure.

7.4 Summary

This chapter introduced a new criterion for learning Bayes nets: find the graph G that maximizes a given score, subject to the constraint that G cover the dependencies detected by a statistical test. This is a hybrid criterion that combines the basic idea behind constraint based approaches—to treat the output of a statistical test as constraints on the learned structure—with the search+score framework. The practical motivation for the hybrid criterion is that many standard scoring criteria based on parameter counts tend to produce

overly sparse graphs; our criterion selects expanded graphs that fit the observed statistically significant correlations.

We showed how to adapt a generic local search+score procedure for the constrained optimization required by the hybrid criterion. In the case of GES search, the resulting IGES search procedure provably maintains the convergence and optimality guarantees associated with GES (provided that the conditional dependencies accepted by the statistical test hold in the target distribution). We presented evidence from simulation studies (from both real-world structures and synthetic ones) with the well-established BDeu criterion that fitting dependencies even with relatively small conditioning sets (at most 3 variables) leads to better learning, as evaluated both by distributional and topological criteria. Most model selection criteria penalize networks with more parameters. For Bayes net structures with discrete variables, the number of parameters is large compared to the number of variables, and thus the penalization term tends to lead to underfitting. We expect that our hybrid approach will be effective for score functions that penalize parameter counts.

In sum, our hybrid criterion is a bridge between the two main frameworks for Bayes net learning, score-based and constraint-based. It appears to be a principled and effective way to address underfitting tendencies in Bayes net learners.

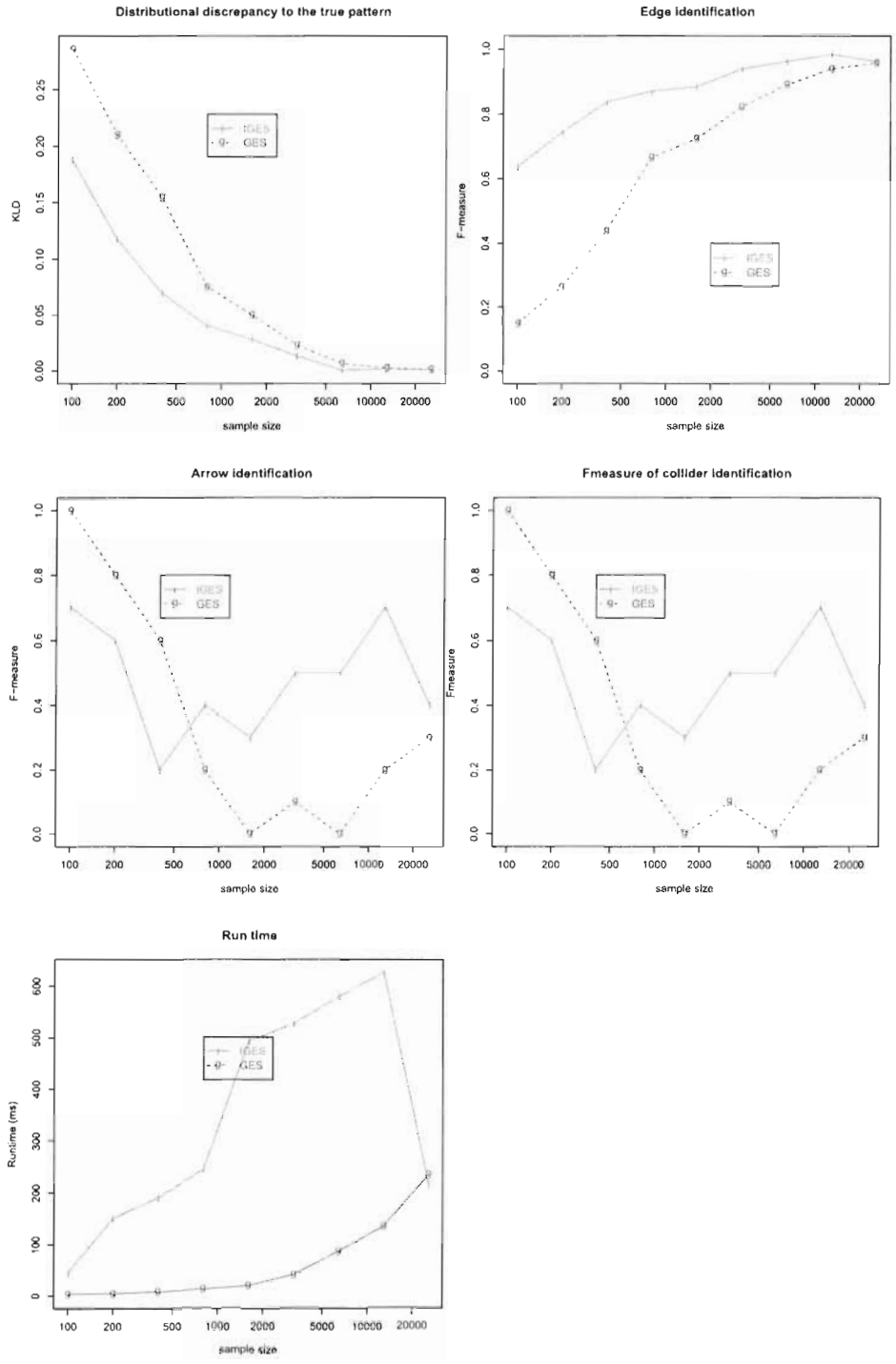


Figure 7.5: Evaluation measures of learned graphs from target graphs with exactly 5 nodes and up to 10 edges. The structure prior is set to 0.001, the sample prior to 10. Average is taken over every 10 samples.

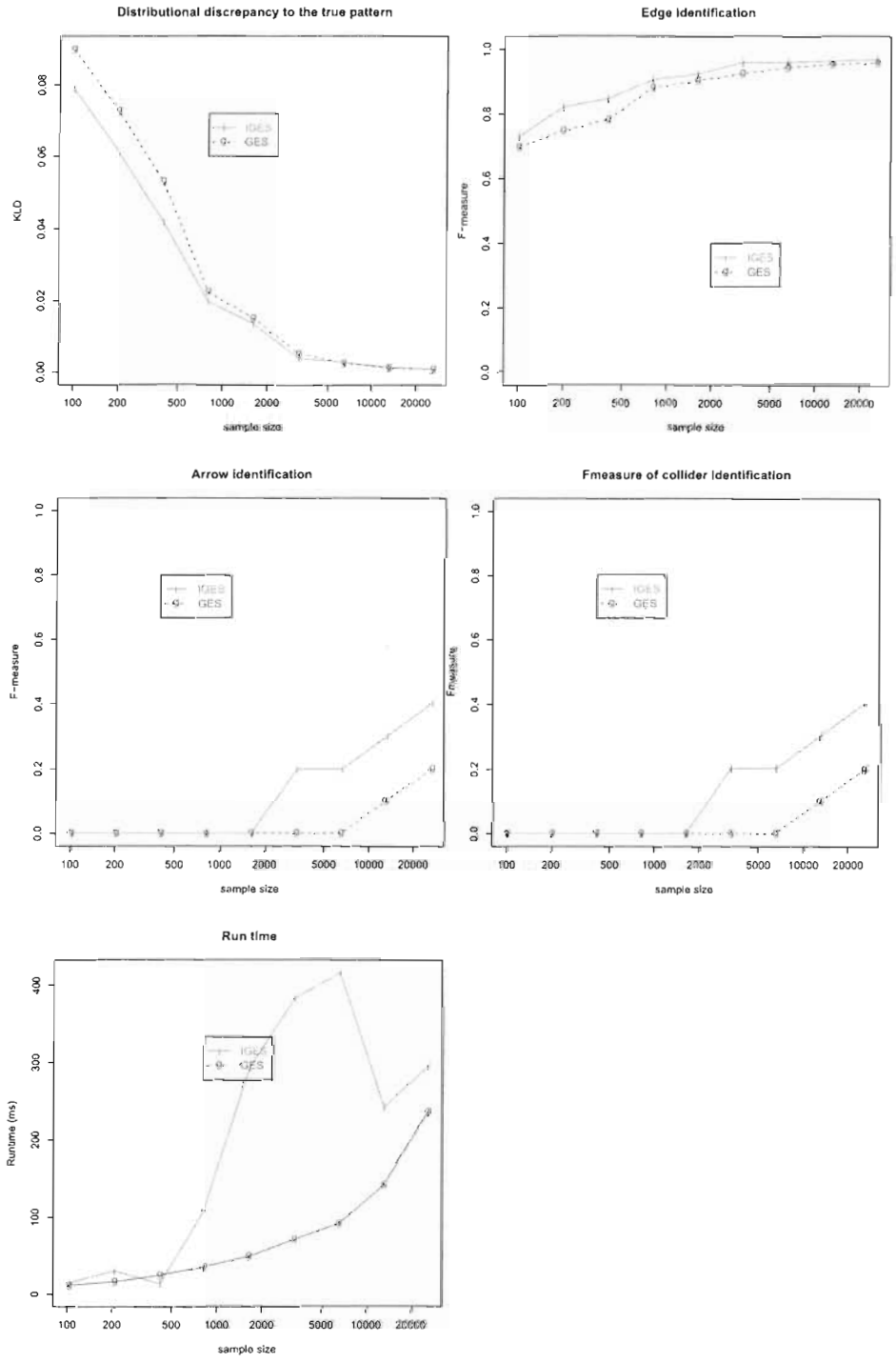


Figure 7.6: Evaluation measures of learned graphs from target graphs with exactly 5 nodes and up to 10 edges. The structure prior is set to 1, the sample prior to 10. Average is taken over every 10 samples.

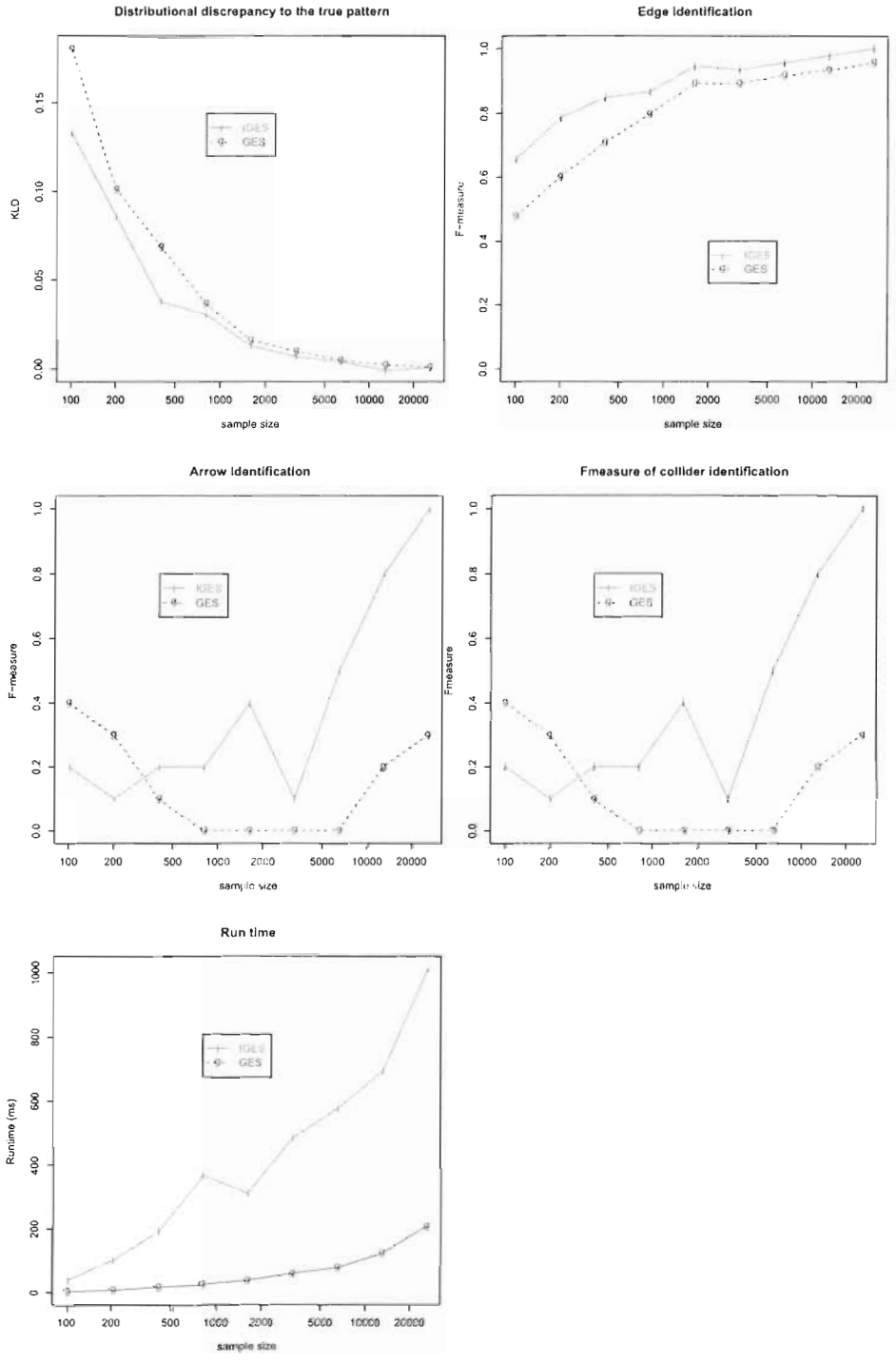


Figure 7.7: Evaluation measures of learned graphs from target graphs with exactly 5 nodes and up to 10 edges. The structure prior is set to 1, the sample prior to 1. Average is taken over every 10 samples.

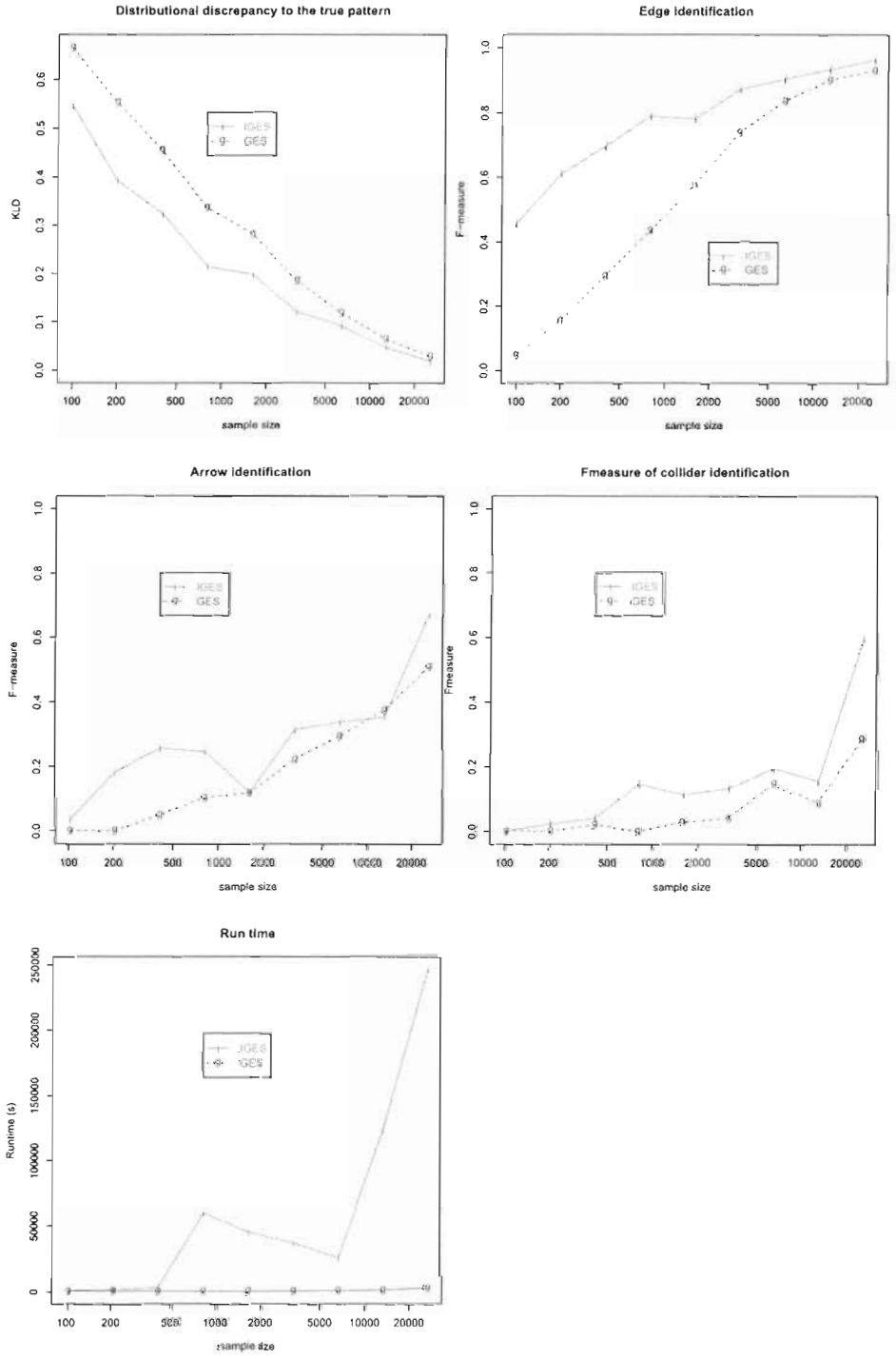


Figure 7.8: Evaluation measures of learned graphs from target graphs with exactly 8 nodes and up to 24 edges. The structure prior is set to 0.001, the sample prior to 10. Average is taken over every 10 samples.

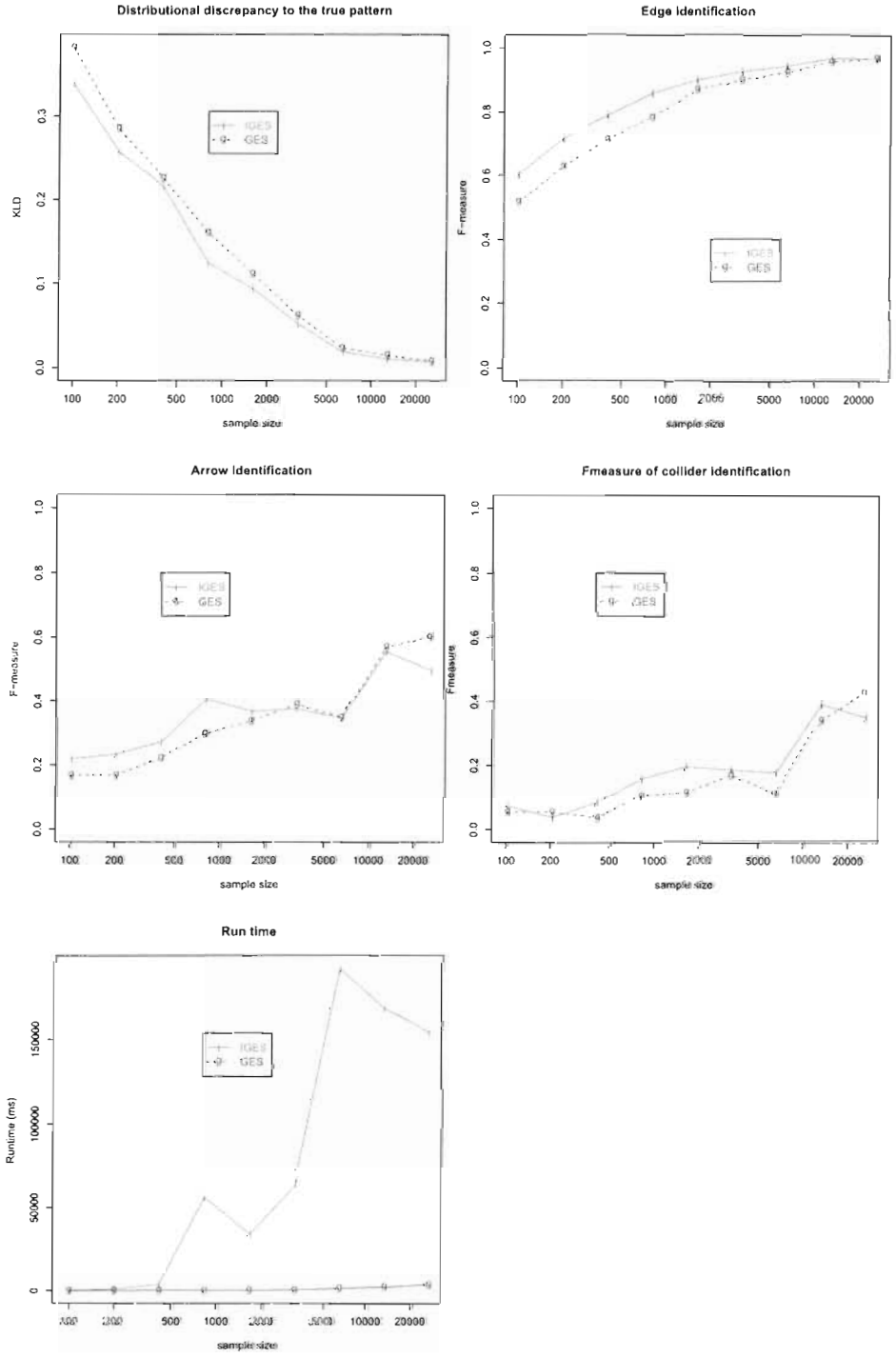


Figure 7.9: Evaluation measures of learned graphs from target graphs with exactly 8 nodes and up to 24 edges. The structure prior is set to 1, the sample prior to 10. Average is taken over every 10 samples.

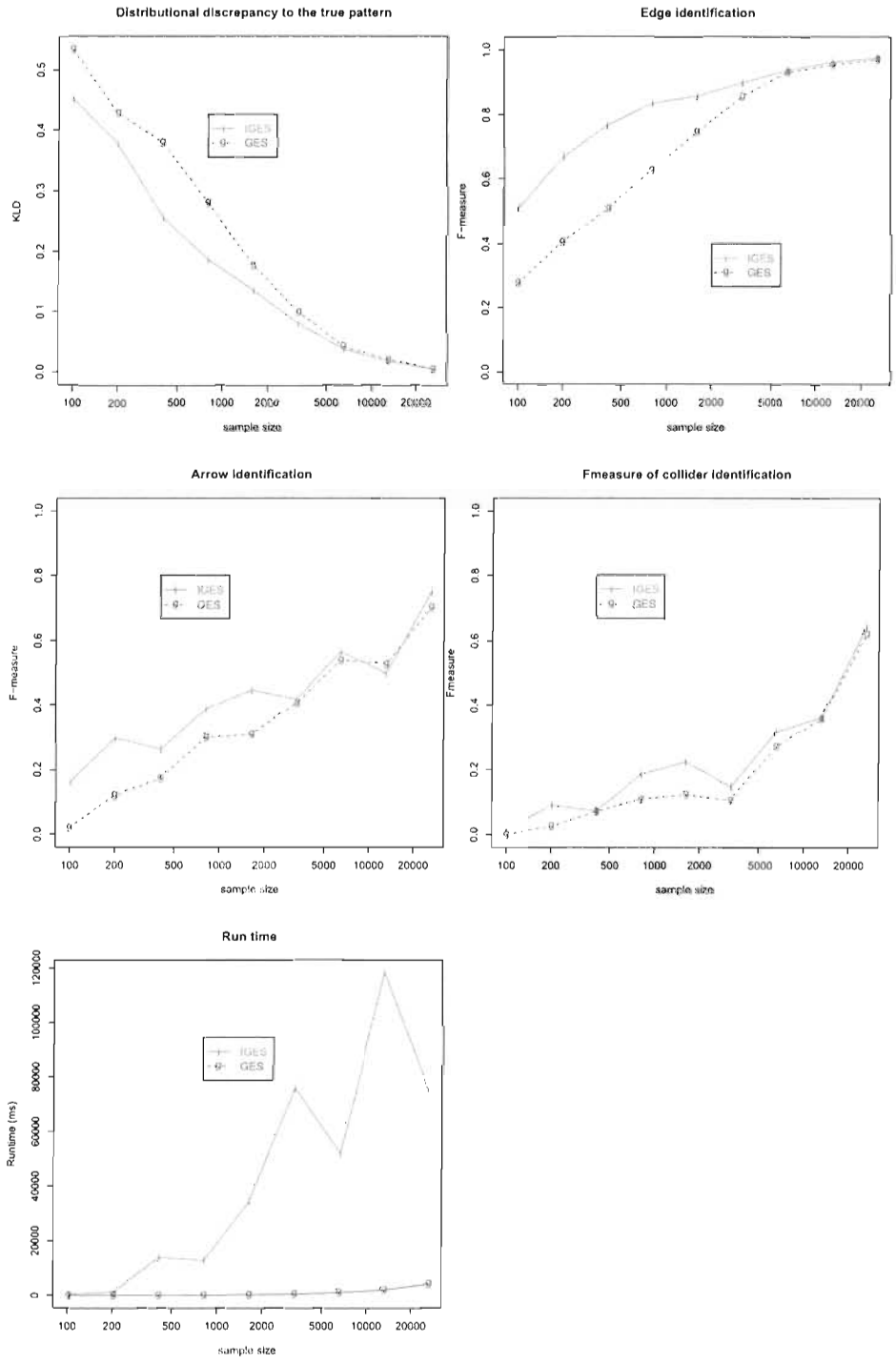


Figure 7.10: Evaluation measures of learned graphs from target graphs with exactly 8 nodes and up to 24 edges. The structure prior is set to 1, the sample prior to 1. Average is taken over every 10 samples.

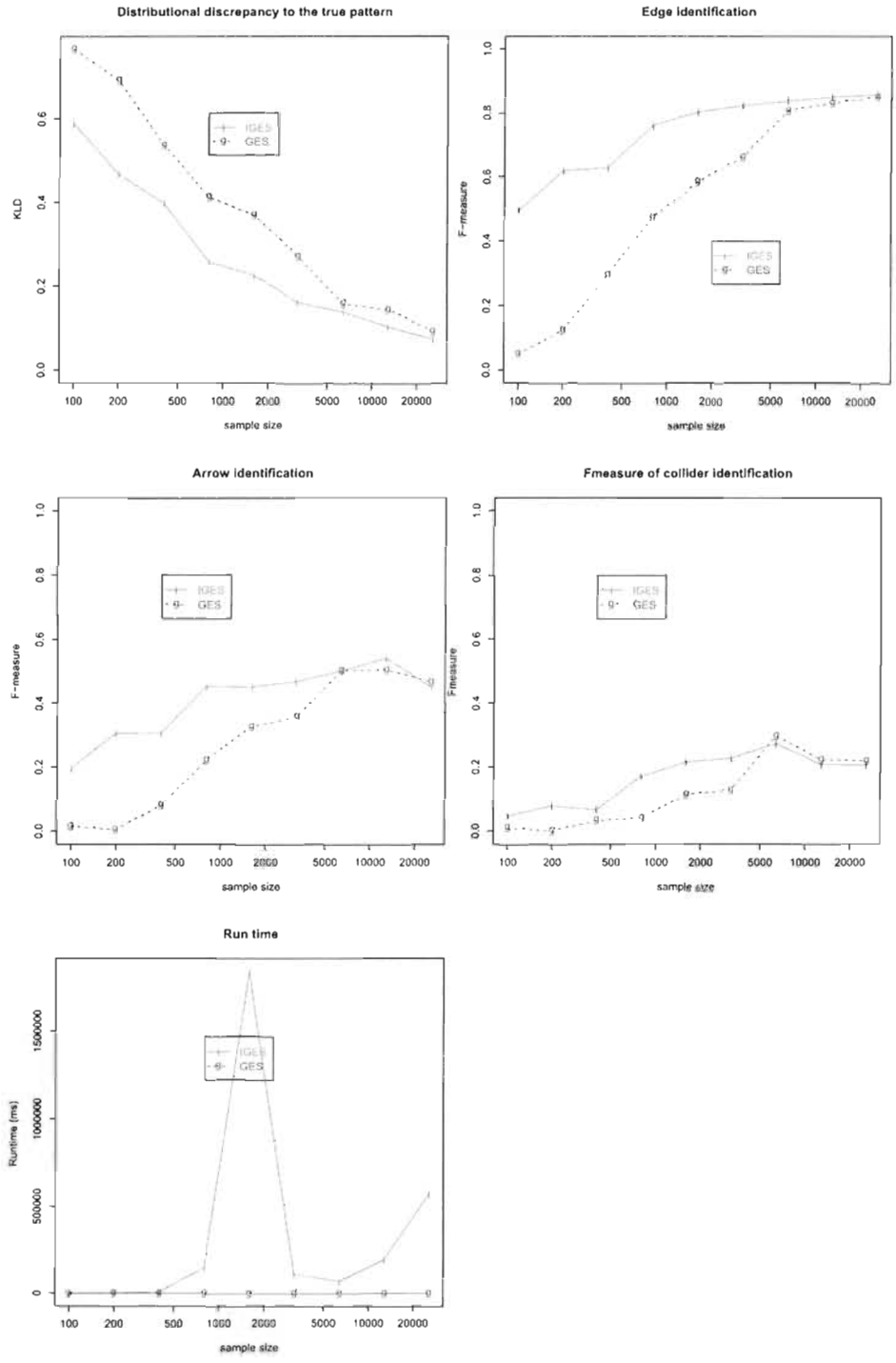


Figure 7.11: Evaluation measures of learned graphs from target graphs with exactly 10 nodes and up to 30 edges. The structure prior is set to 0.001, the sample prior to 10. Average is taken over every 10 samples.

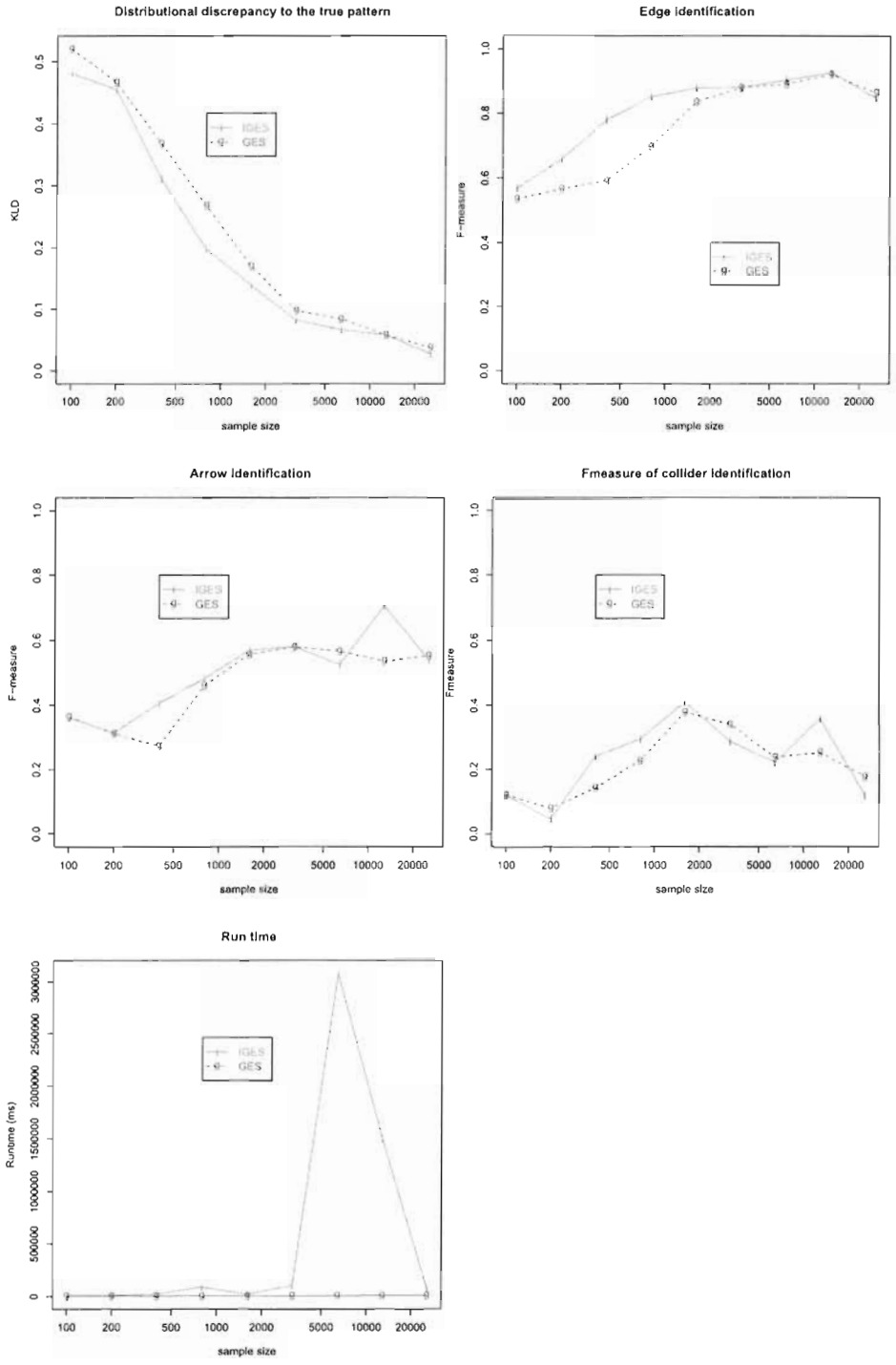


Figure 7.12: Evaluation measures of learned graphs from target graphs with exactly 10 nodes and up to 30 edges. The structure prior is set to 1, the sample prior to 10. Average is taken over every 10 samples.

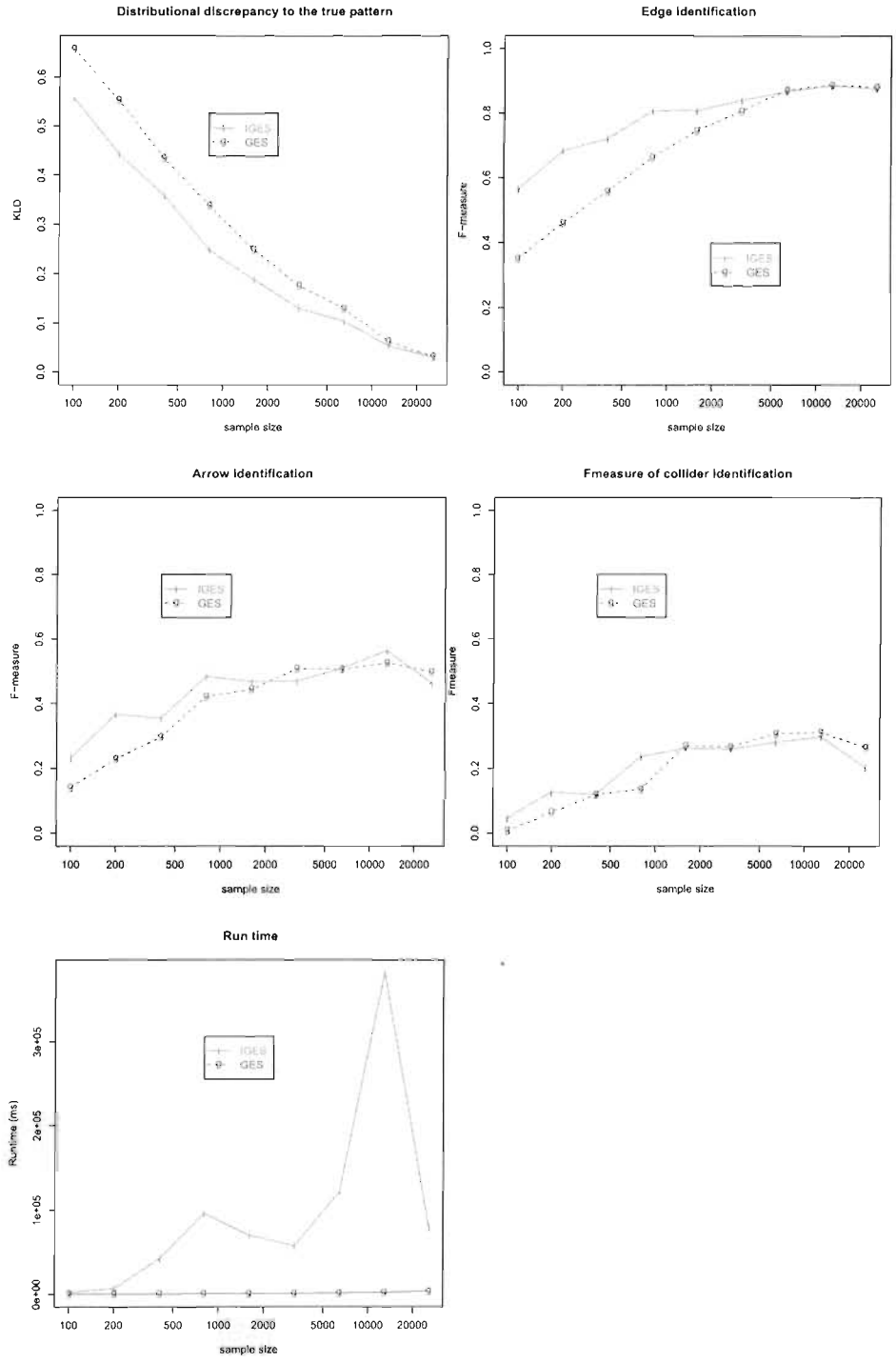


Figure 7.13: Evaluation measures of learned graphs from target graphs with exactly 10 nodes and up to 30 edges. The structure prior is set to 1, the sample prior to 1. Average is taken over every 10 samples.

Chapter 8

Conclusion

People learn inductively.

Joel Peterson

This thesis establishes an inductive principle in terms of mind change complexity in the context of Gold's learning paradigm: *When a learner changes its mind to a hypothesis, the hypothesis should correspond to a language that uniquely has the highest accumulation order in the subproblem entailed by previous observations.* When applied to the problem of learning Bayes net structure, the principle leads to a hybrid criterion that combines score-based search with information from statistical tests. Empirical evaluation provides evidence that the application of the inductive principle effects a substantial improvement on the machine learning task.

8.1 Contributions of the Thesis

The contributions of the thesis to learning theory and machine learning are summarized as follows.

Mind Change Efficient Learning We investigated a refinement to the notion of *identification with bounded mind change* in Gold's language learning paradigm. Briefly, a learner is mind change optimal (or mind change efficient) if the learner achieves the best possible mind change bound not only for the entire problem, but also relative to any data sequences that the learner may observe. We provided necessary and sufficient conditions for a learner to be mind change efficient. These results show that mind change optimality strongly constrains

the conjectures of learners in a similar way as Occam's Razor does for model selection problems. Hence mind change optimality can be viewed as Occam's Razor for language learning problems.

Topological Characterization of Mind Change Complexity We studied the notion of mind change complexity in the framework of point-set topology. Previous work has shown the usefulness of topology for learning theory. We showed how to view a language collection as a topological space; this allowed us to apply Cantor's classic concept of *accumulation order* which assigned an ordinal to a language collection, if the collection had bounded accumulation order. We showed that a language collection $\mathcal{L}$ is identifiable with mind change bound α by a learner if and only if the accumulation order of the topological space entailed by $\mathcal{L}$ is α. This result establishes a purely information-theoretic, structural necessary condition for language identification with bounded mind changes.

A New Model for Constraint-based Learning of Bayes Net Structure We analyzed the problem of learning the structure of a Bayes net in the framework of Gold's language learning paradigm. We based the learning on observed conditional dependencies among variables of interest, which to our knowledge is a new practice in constraint-based learning of Bayes net. Applying learning criteria in this model leads to the following results.

1. The mind change complexity of identifying a Bayes net pattern over variables $\mathbf{V}$ from dependency data is $\binom{|\mathbf{V}|}{2}$, the maximum number of edges for graphs over $\mathbf{V}$.

2. There is a unique fastest mind-change optimal Bayes net learner, when convergence speed is evaluated using Gold's dominance notion of "uniformly faster convergence".

A New Framework for Combining Constraint-based and Score-based Learning of Bayes Net Structure We developed a hybrid criterion for learning Bayes net structure; the criterion combines search based on a scoring function $S(\cdot)$ with information from statistical tests: Given a sample $\mathbf{d}$, search for a structure G that maximizes the score $S(G, \mathbf{d})$, over the set of structures G that satisfy the dependencies detected in d. Again, we rely on the statistical test only to reveal conditional dependencies, not conditional independencies. We showed how to adapt local search algorithms to accommodate the observed dependencies. Simulation studies with the GES search and the BDeu scoring function provide evidence

that the additional dependency information leads to a substantially improved structure on small to medium samples.

8.2 Recommendations for Further Study

There are several avenues for future work, in both Gold's paradigm and Bayes net theory.

Mind Change Optimality and Time Efficiency An interesting open issue in the general theory of SMC-optimal learning is the relationship between mind change optimality and time efficiency. As the example of one-variable patterns shows, there can be a trade-off between time efficiency and producing consistent conjectures, on the one hand, and the procrastination that minimizing mind changes may require on the other (see Sect. 4.2). We would like to characterize the learning problems for which this tension arises, and how great the trade-off can be. For example, if a language collection $\mathcal{L}$ is closed under intersection, then conjecturing $\cap(\mathcal{L}|\sigma)$ for every data sequence σ yields an SMC-optimal learner that never procrastinates (the so-called "closure algorithm"[9]). The language collection LINEAR and the learner Ψ_{LIN} are an instance of an intersection-closed language class and the corresponding closure algorithm. Are there other general sufficient or necessary conditions for a procrastination-free SMC-optimal learner?

Other Potential Applications of the Theory As we have seen, mind change optimality imposes strong constraints on learners. This means that we can apply our theory to design optimal learning algorithms for problems of interest. Such an analysis can validate existing inference procedures, as in the case of learning conservation laws, or lead to the development of new ones, as with one-variable patterns. In Chapter 5, we have seen a mind change optimal algorithm that identify a correct graph in the limit from independence data. Other potential applications include the following. For pattern languages, the next challenge is to find an SMC-optimal algorithm for learning a general pattern with arbitrarily many variables. An important step towards that goal would be to determine the accumulation order of a pattern language $L(p)$ in the space of pattern languages [59]. Another application is the design of SMC-optimal learners for logic programs. For example, Jain and Sharma have examined classes of logic programs that can be learned with bounded mind changes using explorer trees [45]. Do explorer trees lead to mind change optimal learning algorithms?

Simulation Studies of Constraint Optimization with Other Score Functions and Search Methods We presented evidence from simulation studies (from both real-world structures and synthetic ones) with the well-established BDeu criterion that fitting dependencies even with relatively small conditioning sets (at most 3 variables) leads to better learning, as evaluated both by distributional and topological criteria. Most model selection criteria penalize networks with more parameters. For BN structures with discrete variables, the number of parameters is large compared to the number of variables, and thus the penalization term tends to lead to underfitting. We expect that our hybrid approach will be effective for score functions that penalize parameter counts—a task for future research is to verify this expectation for other scoring functions than BDeu. An important task for future work is to improve the efficiency of implementing IGES search, mainly in terms of the number of statistical tests performed and the time for checking if a graph satisfies the given set of dependencies. For example, if the starting point of the IGES search contains an edge A-B, then during the growth phase it is not necessary to test if $A \not\perp\!\!\!\perp B|\mathbf{S}$ for any variable set $\mathbf{S}$.

The idea of imposing dependency constraints on Bayes net learning is quite general and can be applied to modify CB methods as well as score-based methods. For example, the well-known PC algorithm [93, Ch.5] performs a number of independence tests, but does not always cover the dependencies returned by the test (it does cover the independencies). A natural application of our approach is to expand the structure learned by the PC search to satisfy these dependencies; our experiments so far indicate that this leads to improvements for PC search similar to those for the BDeu score.

Adaptively Select Statistical Independence Tests In constrained score optimization of Bayes net structure, an important and challenging topic of research is how to target statistical independence tests so that they are maximally helpful for a given stage of model construction. Constraint-based and hybrid methods follow a strategy for applying statistical tests/computing statistical measures to aid the model search as it proceeds; it should be possible to adapt some of these ideas for constrained optimization. This could motivate interesting problems in both extremal combinatorics [48] and active learning [7][26].

Bibliography

[1] T. V. Allen and R. Greiner. Model selection criteria for learning belief nets: An empirical comparison. In *ICML*, pages 1047–1054, 2000.

[2] A. Ambainis. The power of procrastination in inductive inference: How it depends on used ordinal notations. In P. Vitanyi, editor, *Proceedings of EuroCOLT 1995)*, pages 99–111. Springer, Berlin, 1995.

[3] A. Ambainis, S. Jain, and A. Sharma. Ordinal mind change complexity of language identification. *Theor. Comput. Sci.*, 220(2):323–343, 1999.

[4] D. Angluin. Finding patterns common to a set of strings. *J. Comput. Syst. Sci.*, 21(1):46–62, 1980.

[5] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45(2):117–135, 1980.

[6] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[7] D. Angluin. Queries revisited. *Theoretical Computer Science*, 313:175–194, 2004.

[8] K. Apsitis. Derived sets and inductive inference. In S. Arikawa and K. P. Jantke, editors, *Proceedings of ALT 1994*, pages 26–39. Springer, Berlin, Heidelberg, 1994.

[9] P. Auer and R. Ortner. A new pac bound for intersection-closed concept classes. In *COLT*, pages 408–414, 2004.

[10] G. Baliga, J. Case, and S. Jain. The synthesis of language learners. *Information and Computation*, 152:16–43, 1999.

[11] J. D. Baum. *Elements of Point Set Topology.* rep. Dover, 1991, 1964.

[12] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth. Occam's razor. *Information Processing Letters*, pages 337–380, 1987.

[13] R. R. Bouckaert. *Bayesian belief networks: from construction to inference.* PhD thesis, Universiteit Utrecht, 1995.

[14] R. R. Bouckaert. *Bayesian Network Classifier in Weka*, 2004.

[15] G. Cantor. Grundlagen einer allgemeinen Mannigfaltigkeitslehre. In W. Ewald, editor, *From Kant to Hilbert*, volume 2, pages 878–920. Oxford Science Publications, 1996.

[16] G. Casella and R. L. Berger. *Statistical Inference*. Duxbury Press, 2001.

[17] J. Cheng and R. Greiner. Learning bayesian belief network classifiers: Algorithms and system. In *AI 2001*, number 2056 in LNAI, pages 141–151, 2001.

[18] J. Cheng, R. Greiner, J. Kelly, D. Bell, and W. Liu. Learning bayesian networks from data: An information-theory based approach. *Artificial Intelligence*, 137:43–90, 2002.

[19] D. Chickering. Optimal structure identification with greedy search. *J. Mach. Learn. Res.*, 3:507–554, 2003.

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

[21] D. M. Chickering, D. Heckerman, and C. Meek. Large-sample learning of bayesian networks is NP-hard. *Journal of Machine Learning Research*, 5:1287–1330, 2004.

[22] D. M. Chickering and C. Meek. Finding optimal bayesian networks. In *UAI*, pages 94–102, 2002.

[23] colt. Colt website. URL: www.learningtheory.org.

[24] G. Cooper. An overview of the representation and discovery of causal relationships using bayesian networks. In Glymour and Cooper [36], pages 4–62.

[25] R. G. Cowell, S. L. Lauritzen, and D. J. Spiegelhater. *Probabilistic Networks and Expert Systems*. Springer, 2005.

[26] S. Dasgupta. Analysis of a greedy active learning strategy. In *NIPS*, 2004.

[27] L. M. de Campos. A scoring function for learning bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, pages 2149–2187, 2006.

[28] M. H. Degroot. *Probability and Statistics*. Addison Wesley, 2 edition, 1986.

[29] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, 2 edition, 2000.

[30] R. A. Fisher. *Statistical Methods for Research Workers*. Macmillan Pub Co, 1925. Available at http://psychclassics.yorku.ca/Fisher/Methods.

[31] R. Freivalds, E. Kinber, and C. H. Smith. On the intrinsic complexity of learning. *Inf. Comput.*, 123(1):64–71, 1995.

[32] N. Friedman, D. Pe'er, and I. Nachman. Learning bayesian network structure from massive datasets: The "sparsecandidate" algorithm. In *Proc. Fifteenth Conf. on Uncertainty in Artificial Intelligence (UAI)*, pages 206–215, 1999.

[33] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman Co., 1979.

[34] R. N. Giere. The significance test controversy. *The British Journal for the Philosophy of Science*, 23(2):170–181, 1972.

[35] C. Glymour. The hierarchies of knowledge and the mathematics of discovery. *Minds and Machines*, pages 75–95, 1991.

[36] C. Glymour and G. Cooper, editors. *Computation, Causation, and Discovery.* AAAI Press/The MIT Press, 1999.

[37] E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967.

[38] P. Grünwald. A tutorial introduction to the minimum description length principle. In *Advances in Minimum Description Length: Theory and Applications.* MIT Press, 2005.

[39] D. Heckerman. A tutorial on learning with bayesian networks. In *Proceedings of the NATO Advanced Study Institute on Learning in graphical models*, pages 301–354, Norwell, MA, USA, 1998. Kluwer Academic Publishers.

[40] D. Heckerman, D. Geiger, and D. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. In R. L. de Mantaras and D. Poole, editors, *Uncertainty in Artificial Intelligence*, volume 10, pages 293–301, San Mateo, CA, 1994. Morgan Kaufmann.

[41] F. L. Hogben. Statistical prudence and statistical inference. In D. Morrison and R. Henkel, editors, *The Significance Test Controversy.* Aldine Publishing, 1970.

[42] S. Jain, E. B. Kinber, and R. Wiehagen. Language learning from texts: Degrees of intrinsic complexity and their characterizations. *J. Comput. Syst. Sci.*, 63(3):305–354, 2001.

[43] S. Jain, D. Osherson, J. S. Royer, and A. Sharma. *Systems That Learn.* M.I.T. Press, 2 edition, 1999.

[44] S. Jain and A. Sharma. The intrinsic complexity of language identification. *J. Comput. Syst. Sci.*, 52(3):393–402, 1996.

[45] S. Jain and A. Sharma. Mind change complexity of learning logic programs. *TCS*, 284(1):143–160, 2002.

[46] A. J. Jayanthan. Derived length for arbitrary topological spaces. *International Journal of Mathematics and Mathematical Sciences*, 15(2):273–277, 1992.

[47] F. V. Jensen. *Bayesian Networks and Decision Graphs*. Springer, 2002.

[48] S. Jukna. *Extremal Combinatorics: With Applications in Computer Science*. Springer-Verlag, 2001.

[49] K. Kelly. *The Logic of Reliable Inquiry*. Oxford University Press, 1996.

[50] K. Kelly. Efficient convergence implies Ockham's Razor. In *Proceedings of the 2002 International Workshop on Computation Models of Scientific Reasoning and Applications*, pages 24–27, 2002.

[51] K. Kelly. A close shave with realism: Ockham's razor derived from efficient convergence. Completed Manuscript, available at http://www.andrew.cmu.edu/user/kk3n/kelly/ockham.pdf, 2003.

[52] K. Kelly. Justification as truth-finding efficiency: How ockham's razor works. *Minds and Machines*, 14(4):485–505, 2004.

[53] S. Kocabas. Conflict resolution as discovery in particle physics. *Machine Learning*, 6:277–309, 1991.

[54] S. Kullback and R. A. Leibler. On information and sufficiency. *Ann. Math. Statist.*, 22:79–86, 1951.

[55] K. Kuratowski. *Topology*, volume 1. Academic Press, 1966. Translated by J. Jaworowski.

[56] S. Lange and T. Zeugmann. Language learning with a bounded number of mind changes. In *Symposium on Theoretical Aspects of Computer Science*, pages 682–691, 1993.

[57] S. Lange and T. Zeugmann. Learning recursive languages with a bounded number of mind changes. *International Journal of Foundations of Computer Science*, 4(2):157–178, June 1993.

[58] H. Linhart and W. Zucchini. *Model Selection*. John Wiley and Sons, 1986.

[59] W. Luo. Compute inclusion depth of a pattern. In *COLT 2005*, pages 689–690, 2005.

[60] W. Luo and O. Schulte. Mind change efficient learning. *Information and Computation*, 204:989–1011, 2006.

[61] D. Margaritis and S. Thrun. Bayesian network induction via local neighborhoods. In *Advances in Neural Information Processing Systems*, volume 12, pages 505–511. MIT Press, 2000.

[62] E. Martin and D. N. Osherson. *Elements of Scientific Inquiry.* The MIT Press, Cambridge, Massachusetts, 1998.

[63] E. Martin and A. Sharma. On a syntactic characterization of classification with a mind change bound. In *COLT 2005*, number 3559 in Lecture Notes in Computer Science, pages 413–428. Springer, 2005.

[64] E. Martin, A. Sharma, and F. Stephan. Learning, logic, and topology in a common framework. In *Proceedings of the 13th International Conference on Algorithmic Learning Theory*, pages 248–262. Springer-Verlag, 2002.

[65] C. Meek. *Graphical Models: Selecting causal and statistical models.* PhD thesis, Carnegie Mellon University, 1997.

[66] T. Motoki, T. Shinohara, and K. Wright. The correct definition of finite elasticity: corrigendum to identification of unions. In *Proceedings of COLT 1991*, page 375. Morgan Kaufmann Publishers Inc., 1991.

[67] Y. Mukouchi. Characterization of pattern languages. In *Proc. 2nd workshop on algorithmic learning theory (ALT'91)*, pages 93–104, 1991.

[68] Y. Mukouchi. Inductive inference with bounded mind changes. In S. Doshita, K. Furukawa, K. P. Jantke, and T. Nishida, editors, *Proceedings of ALT 1992*, pages 125–134. Springer, Berlin, Heidelberg, 1993.

[69] K. P. Murphy. The bayes net toolbox for matlab. In *Computing Science and Statistics*, volume 33, pages 1–20, 2001.

[70] R. E. Neapolitan. *Learning Bayesian Networks.* Pearson Education, 2004.

[71] P. Odifreddi. *Classical Recursion Theory.* North-Holland, October 1999.

[72] D. Osherson and S. Weinstein. A note on formal learning theory. *Cognition*, 11:77–88, January 1982.

[73] D. N. Osherson, M. Stob, and S. Weinstein. *Systems that learn: an introduction to learning theory for cognitive and computer scientists.* MIT Press, 1986.

[74] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994.

[75] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kauffmann, San Mateo, CA, 1988.

[76] J. Pearl. *Causality: Models, Reasoning, and Inference.* Cambridge university press, 2000.

[77] C. Philosophy. Tetrad project. url: http://www.phil.cmu.edu/projects/tetrad/.

[78] S. Pinker. Formal models of language learning. *Cognition*, 7:217–283, 1979.

[79] G. Plotkin. A note on inductive generalization. In *Machine Intelligence*, volume 5, pages 153–163. Edinburgh University Press, 1970.

[80] G. Plotkin. A further note on inductive generalization. In *Machine Intelligence*, volume 6, pages 101–124. Edinburgh University Press, 1971.

[81] H. Putnam. Trial and error predicates and the solution to a problem of mostowski. *The Journal of Symbolic Logic*, 30(1):49–57, March 1965.

[82] J. Ramsey, P. Spirtes, and J. Zhang. Adjacency-faithfulness and conservative causal inference. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 401–408. AUAI Press, 2006.

[83] M. T. Review. 10 emerging technologies that will change your world, Feburary 2004.

[84] J. Rissanen. Modeling by shortest data description. *Automatica*, pages 465–471, 1978.

[85] R. Scheines, P. Spirtes, C. Glymour, C. Meek, and T. Richardson. *TETRAD 3:Tools for Causal Modeling – User's Manual.* CMU Philosophy, 1996.

[86] O. Schulte. Formal learning theory. Stanford Encyclopedia of Philosophy. available at http://plato.stanford.edu/entries/learning-formal.

[87] O. Schulte. Means-ends epistemology. *The British Journal for the Philosophy of Science*, 50:1–31, 1999.

[88] O. Schulte. Inferring conservation laws in particle physics: A case study in the problem of induction. *The British Journal for the Philosophy of Science*, 51(4):771–806, December 2000.

[89] O. Schulte. Automated discovery of conservation principles and new particles in particle physics. Manuscript submitted to *Machine Learning*, 2005.

[90] O. Schulte, W. Luo, and R. Greiner. Mind change optimal learning of bayes net structure. In *COLt 2007*, 2007.

[91] A. Sharma, F. Stephan, and Y. Ventsov. Generalized notions of mind change complexity. In *Computational Learing Theory*, pages 96–108, 1997.

[92] T. Shinohara. Inductive inference of monotonic formal systems from positive data. *New Gen. Comput.*, 8(4):371–384, 1991.

[93] P. Spirtes, C. Glymour, and R. Scheines. *Causation, prediction, and search.* MIT Press, 2000.

[94] P. Spirtes and C. Meek. Learning Bayesian networks with discrete variables from data. In *Proceedings of First International Conference on Knowledge Discovery and Data Mining,* Montreal, QU, pages 294–299. Morgan Kaufmann, August 1995.

[95] S. Thrun et al. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

[96] J. W. Tukey. *Exploratory Data Analysis.* Addison-Wesley, 1977.

[97] R. Valdés-Pérez. Algebraic reasoning about reactions: Discovery of conserved properties in particle physics. *Machine Learning*, 17:47–67, 1994.

[98] R. Valdés-Pérez. On the justification of multiple selection rules of conservation in particle physics phenomenology. *Computer Physics Communications*, 94:25–30, 1996.

[99] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[100] M. Velauthapillai. Inductive inference with bounded number of mind changes. In *Proceedings of COLT 1989*, pages 200–213. Morgan Kaufmann Publishers Inc., 1989.

[101] T. S. Verma and J. Pearl. Equivalence and synthesis of causal models. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 220–227, Mountain View, CA, 1990.

[102] K. Wexler and P. Cullicover. *Formal Principles of Language Acquisition.* MIT press, Cambridge, Massachusetts, 1980.

[103] Wikipedia. Computational learning theory. Wikipedia entry.

[104] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques.* Morgan Kaufmann, 2 edition, 2005.

[105] K. Wright. Identification of unions of languages drawn from an identifiable class. In *Proceedings of the second annual workshop on Computational learning theory*, pages 328–333. Morgan Kaufmann Publishers Inc., 1989.

[106] Y. Xiang, S. K. Wong, and N. Cercone. Critical remarks on single link search in learning belief networks. In *Proceedings of the 12th Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pages 564–57, 1996.

[107] W. Zucchini. An introduction to model selection. *Journal of Mathematical Psychology*, 44:41–61, 2000.