H24/3597

MONASH UNIVERSITY
THESIS ACCEPTED IN SATISFACTION OF THE
REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
ON..... 22 July 2003 .....
.....[redacted].....
Sec. Research Graduate School Committee

Towards Robust Discovery Systems

By Murlikrishna Viswanathan

B.Sc. Honours (Deakin Univ.)

This thesis is submitted in fulfillment of the requirements

for the Doctor of Philosophy

School of Computer Science and Software Engineering

Monash University, Melbourne, Australia

July 2003

Contents

List of Figures

List of Tables

Abstract

Inductive learning — learning about the world we are in — is fundamental and pervasive. And amongst the plethora of learning activities, scientific induction stands out as the crown jewel of intellectual enquiry, and so the natural aim of machine learning would be to produce it. However, it has long been held by prominent philosophers of science that scientific reasoning is actually *impossible* to automate. Hans Reichenbach's view [81] was that, whereas there may be a logic of justification for deciding which theories, once proposed, are right and which wrong, there could be no logic of discovery. Popper [78], and the other logical positivists, concurred, holding discovery to be within the realm of psychology, if not mysticism (despite the misleading title of his book, Logic of Scientific Discovery, as translated into English). That view (and positivism generally) is on the retreat. On the one side, it has been undermined by Bayesian philosophies of science, which hold that the origin of a theory has much to do with its plausibility (prior probability) and so with its ultimate justification. On the other side, AI researchers have been actively developing algorithms for generating scientific theories, wherein claims of success have often been grossly exaggerated. Nevertheless, we find that the prospects for automating scientific discovery in the long run are encouraging, and in particular, we present algorithms that provide robust model selection in the domains of curve-fitting and analysis of time-series data.

The history of science reveals at least five stages in scientific discovery: *concept/taxonomy formation*, *discovering qualitative laws*, *discovering quantitative laws*, *discovering structural models* and *discovering temporal models*. Recent research in computational discovery has addressed all the five stages of the discovery process. Clustering systems like Cluster/2, SNOB, AUTO-

CLASS and others deal with the task of taxonomy formation, whereas systems like NGLAUBER search for qualitative relations. Starting with BACON, researchers have developed a great variety of systems that discover numeric laws. Systems like DALTON, STAHLP, and GELL-MANN formulate structural models; some others, such as TETRAD II, generate specifically causal models. Researchers have also tried to integrate the different stages of discovery as found in systems like Falkenhainer and Michalski's ABACUS, Nordhausen and Langley's IDS, and Kulkarni and Simon's KEKADA. [59]

Despite a dearth of real applications to which any of these systems have been put, we believe that the general project of automating scientific discovery is promising for the long term. One serious flaw which all of the above systems share is an overblown respect for Reichenbach's discovery-justification distinction. Whereas they all take seriously the attempt to furnish heuristics for discovering scientific theories, none of them makes a serious attempt to integrate the contexts of discovery and justification. There are two specific failings in this research tradition: the failure to apply any principled metric for directing the discovery process; the failure to apply any principled metric for evaluating the theories produced by the discovery process.

The Minimum Message Length principle (MML) [122, 127] defines an information theoretic Bayesian approach to assessing the probabilities of theories relative to some data. By preferring theories which minimize the joint encoding of the theory itself and the data relative to that theory it plays a trade-off between theoretical simplicity and fit to the data which demonstrably favors the most probable hypothesis a posteriori. The MML derived metric is useful not just for a final evaluative step, but also for directing discovery towards the most promising regions of the space of theories. MML is especially amenable to incorporation into computer programs and has al-

ready been applied to a variety of important classes of inductive inference problems, including concept and taxonomy formation, polynomial curve fitting and causal model discovery. The novel contributions of this dissertation include the design of MML-based systems including empirical studies that demonstrate the plausible performance of the MML principle in comparison to several well-known metrics. These MML-based systems developed for the domains of polynomial curve fitting [115] and time series analysis [116] with noisy data, which form the core contributions of this dissertation, have shown performance clearly superior to its main statistical competitors.

Declaration

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university or other institution.

To the best of my knowledge, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Where the work in this thesis is based upon joint research, this thesis discloses the relative contributions of the respective authors.

Murlikrishna Viswanathan

July 2, 2003

Acknowledgments

I am grateful to my supervisors, namely Dr. Kevin Korb and Dr. David Dowe for providing all the motivation and support over the last three years. They have provided excellent supervision and have been instrumental in my acquiring good research skills. I am also obliged to Professor Chris Wallace who has been informally finding problem areas and guiding my efforts to find solutions.

I would also like to express my gratitude to Dr. Guoqi Qian, Dr. Rohan Baxter, Dean McKenzie, for extensive comments on papers submitted earlier.

And finally I would like to dedicate this thesis to my wife Ramya and our eight-month old Brinda, for the love, patience and support in this endeavour.

Chapter 1

Introduction

Keep on the lookout for novel ideas that others have used successfully. Your idea has to be original only in its adaptation to the problem you're working on. Results! Why, man, I have gotten a lot of results. I know several thousand things that won't work.
- Thomas Edison (1847-1931)

1.1 Introductory comments

Inspired by the encouraging words of Thomas Alva Edison I present here my dissertation. Herein I have attempted to present an account of my journey through the different research fields of automated discovery and inductive inference, and the "tiny" contributions that I have so far been able to make. The novel contributions of this dissertation are encapsulated in chapters 5 and 6 while the rest of the chapters describe some relevant literature and a schematic framework.

1.1.1 Gyaanaarjanam (Knowledge Acquisition)

The word "Gyaanaarjanam" is a synonym for the acquisition of knowledge in one of the oldest languages of the world, namely, Sanskrit. Knowledge acquisition or learning has hundreds of definitions reflecting the fact that the human intellect has yet to formalize this innate phenomenon clearly. Learning can be described as a process that reduces our uncertainty about the nature of the object/subject we are trying to understand. But whether we define it or not, learning is fundamental to our existence.

Learning can be understood via the processes of Induction, Abduction and Deduction. The American philosopher Charles Sanders Peirce offers us a classification of these reasoning forms in terms of the function they perform in science. Three stages are discerned: (1) formulating a hypothesis, (2) drawing predictions from the hypothesis, (3) evaluating these predictions. The first stage, coming up with an explanatory hypothesis, is what Peirce calls abduction. Predictions are drawn by deduction; and assessing a hypothesis by testing these predictions in the real world is what Peirce calls induction.

The goal of this dissertation is the design and development of theory and software for the inductive inference of models. As science progresses there is a growing need for the development of inductive systems to understand large and complex, information-rich data from virtually all fields of business, science and engineering. Examples include medical diagnosis, analysis of astronomical data, weather forecasting, corporate data analysis, DNA sequence analysis, etc. The ability to extract useful knowledge hidden in these data and to act on that knowledge is becoming increasingly important.

1.1.2 Automated Inductive Inference

Inductive inference thus primarily refers to acquiring general knowledge from a set of specific examples. Machine Learning is the area of Artificial Intelligence that focuses on developing principles and techniques for automating inductive inference. The study and computer modelling of knowledge acquisition in its multiple manifestations is encompassed by machine learning. Machine learning is often distinguished on the basis of the underlying learning strategies where each strategy corresponds to a specific technique that can be applied to a particular domain using a distinct knowledge representation. Some machine learning methods can dramatically reduce the cost of developing knowledge-based software by extracting knowledge directly from existing databases. Other machine learning methods enable software systems to improve their performance over time with minimal human intervention. A generic model for a typical learning system to this effect is depicted in the following figure.

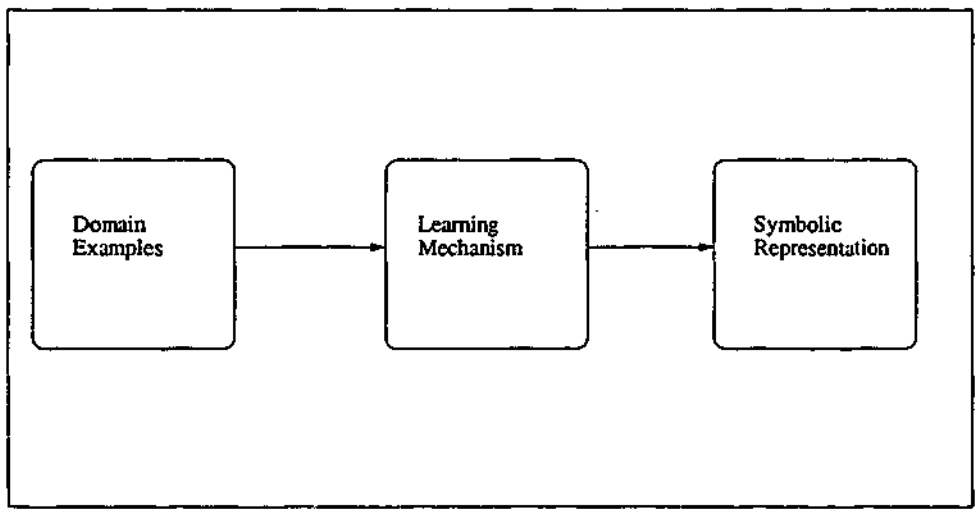


Figure 1.1: A Generic Learning System

Based on this fundamental framework, the field of machine learning has spawned a number of approaches. Some of these include Decision tree learn-

ing, Rule induction, Bayesian methods, Neural computing, Lazy learning, Rough and Fuzzy sets and Support Vector learning. In the long-term it is expected that these approaches will lead to the development of effective software for autonomous systems that must operate in poorly understood environments. This dissertation will focus on Bayesian methods, although comparisons will be made with a wide spectrum of approaches.

Research in machine learning has thus focussed on the design and development of learning algorithms that perform increasingly well on a variety of real-world domains. Several systems have been developed with a good degree of predictive accuracy in real-world domains, examples being Quinlan's C5 [80], Breiman, Friedman, Olshen and Stone's CART [9], Wallace's Snob [126], Michalski's AQ family [64] and more recently Webb's COVER [133].

1.2 Minimum Message Length Inference

The Minimum Message Length (MML) [122, 127] principle for inductive inference was pioneered by Wallace and Boulton in the 1960's. Founded on fundamental ideas from information theory, coding theory, and Bayesian learning, the MML principle suggests that the best theory or explanation (in some formal language) for a body of data is the one that best compresses a two-part encoding of the theory and data (assuming the theory is true) under an optimal encoding scheme. Information compression has been a significant theme in research on inductive inference including inference of grammars under the banner of Minimum Length Encoding (MLE) pioneered by Solomonoff [98], Wallace and Boulton, Rissanen [87] and others. MLE is an umbrella term covering the closely-related concepts of Minimum Message Length encoding (MML) and Rissanen's Minimum Description Length encoding (MDL) [84].

The key idea in MLE is that in inductive inference, one should seek to minimise $(s(T) + s(D))$, where $s(T)$ is the size of the pattern, theory, or grammar (in bits or comparable measure) and $s(D)$ is the size of the raw data from which the grammar is derived, after the raw data has been encoded in terms of the grammar. This idea produces a happy compromise between grammars that are, at one extreme, very small but are inefficient for encoding data and, at the other extreme, grammars that are very efficient for encoding data but are excessively large.

An analogy of this idea can be seen in the behaviour of children who make grammatical generalisations and eventually correct their erroneous 'over' generalisations (e.g. "He hitted me", "Look at those mouses") without apparently needing explicit correction from other people. Both kinds of generalisation, by definition, have zero frequency in the child's experience but MLE provides a means of distinguishing 'good' ones from 'bad' ones without external supervision of the learning process.

1.3 Scientific Discovery

Scientific discovery has always enjoyed the highest reputation among creative intellectual pursuits. However, creative intelligence has yet to be defined and as a result it has seemed to be an unlikely candidate for automation. But as Langley suggests [59], researchers in AI have questioned this attitude by developing artifacts that replicate the act of discovery (at least in a restricted sense). This section provides brief summaries of the scientific discovery process, its philosophy and a historical survey of computational scientific discovery systems.

The history of science has revealed the following definitive stages in sci-

entific discovery. It is important to note that these stages of discovery may not be purely sequential.

1. Taxonomy formation. Prior to establishing theories it is necessary to categorize or classify concepts from the domain under study.

2. Discovering Qualitative laws. The discovery of qualitative laws helps to characterize the behaviour or determine the correlations between a set of entities.

3. Discovering Quantitative laws involves stating mathematical relations between numeric variables representing a set of entities.

4. Discovering Structural Models. Scientists often use structural models which incorporate unobserved entities to explain phenomena that are inexplicable by empirical summaries.

5. Discovery Process Models. Process models help to explain phenomena in terms of models that involve temporal changes.

1.4 Machine Discovery

The computer-aided discovery of scientific knowledge has spawned a field for itself, known as machine discovery. According to Darden [27], the study of computational scientific discovery emerged from the view that science is a problem solving activity, that heuristics for problem solving can be applied to the study of scientific discovery in either historical or contemporary cases, and that methods in artificial intelligence provide techniques for building computational systems. Early research in scientific discovery focused on replicating discoveries from the history of disciplines as diverse as mathematics [62], physics [57], chemistry [143] and biology [55].

Recent research in computational discovery has addressed all five stages of the discovery process. Clustering systems like Cluster/2 [65], AUTOCLASS [17], Snob [126] and others deal with the task of taxonomy formation, whereas systems like NGLAUBER [50] search for qualitative relations. Starting with BACON [57], researchers have developed a great variety of systems that discovery numeric laws. Systems like DALTON [75], STAHLP [89], and GELL-MANN [139] formulate structural models, whereas a smaller group, like MECHEM [107] and ASTRA, instead construct process models.

Researchers have also tried to integrate the different stages of discovery as found in systems like Lenat's AM, Nordhausen and Langley's IDS [72], and Kulkarni and Simon's KEKADA [55].

1.5 Autonomous Discovery and the BODHI project

Discovery systems such as the Bacon family and its successors like Abacus, Coper, Kepler and others were designed to find functional relationships of scientific significance in numerical data. Bacon and its sister programs have been widely critisized for the following reasons:

- results published from the application of these systems were based on the replication of previous scientific discoveries,
- a systematic and principled approach to handling noisy data was lacking, and,
- there was a lack of a well-founded induction principle to guide the search and evaluation of models,

The research towards this dissertation originally commenced with the aim of integrating the several processes involved in the discovery process through the Bayesian-Oriented Discoverer of Heterogeneous Information-structure (BODHI) system. The idea with BODHI is to search a variety of model spaces guided by domain heuristics and the Minimum Message Length (MML) methodology to find a model that best explains the data at hand. The end-product would be a modular environment for automated discovery. While preliminary investigations towards the larger aim of developing a system for the systematic discovery of quantitative and qualitative knowledge from real world data were conduted, the immediate aim was to test the plausibility of the MML methodology for model selection. So the research headed in the direction of developing systems for robust learning in the domains of polynomial regression and binary sequence segmentation by employing the MML principle to guide the search and selection process.

1.6 Objectives and Contributions

This research work originally commenced with the aim of achieving three major goals.

- To study the process of automating scientific discovery;
- To demonstrate the plausibleness of the Minimum Message Length principle in model selection and
- Investigate the modular development of a Bayesian-Oriented Discoverer of Heterogeneous information structure (BODHI). The BODHI project will culminate in the development of an integrated system for scientific discovery.

The PhD work aimed at the above has resulted in the following contributions.

- A polynomial function discovery module based on the MML methodology which can be integrated within a larger discovery system. This function discoverer has been compared and benchmark tested against standard approaches. The system is robust in the presence of noisy data and significantly better in its performance in comparison to standard tecnhiques.

- A system for the segmentation of binary sequence data. The module also based on the MML principle and designed with a larger framework in mind learns Boolean models from binary data. The system has been shown to offer robust inference in the presence of erroneous data.

- A brief review of systems for automating scientific discovery.

1.7 Thesis Overview

Chapter 2 presents the general literature review of the machine learning area with relevance to this work. The chapter also offers a formal survey of bias in machine learning.

Chapter 3 provides some background literature in machine discovery and a review of automated discovery systems.

Chapter 4 introduces model selection methods based on the principles of Minimal Length Encoding. The origins of the MML principle are traced along with the practical techniques used in applying the methodology.

Chapter 5 describes a system for learning univariate polynomial models from noisy observations. The chapter presents one of the two novel contributions of this dissertation. One highlight of this chapter is the empirical analysis of a classical bound on prediction error. The results from this chapter demonstrate the weakness of this bound and the plausible performance of the MML metric. The chapter includes a detailed empirical evaluation of the MML approach with standard benchmark methods.

Chapter 6 presents a system for learning segmentation models from binary sequence data. This chapter highlights another significant contribution of this thesis. A comparative evaluation of the system with standard approaches is included. The chapter offers an important lesson in applying minimum length encoding methods for model selection by demonstrating the problems with sub-optimal coding schemes. Results from empirical evaluation are analysed.

Finally, chapter 7 summarizes the findings of this study and provides directions for future work.

Chapter 2

Machine Learning - A Survey of Relevant Issues

Learning is a many-faceted phenomenon. The ability to learn from observations and experience is crucial for any intelligent being. Knowledge acquisition in this light is the process of acquiring knowledge from one or more sources and passing it in a suitable form to some entity or system. Machine learning within this context can be defined as the transfer and transformation of problem-solving expertise from some knowledge source to a program.

Research in machine learning has been primarily concerned with building computer programs able to construct new knowledge or to improve already existing knowledge from evidence. Besides the fact that learning is a part of any kind of problem solving or process, a diversity of research tasks and testing grounds are available to researchers. Machine learning has become central to the development of artificial intelligence owing to its importance in several areas, including expert systems, problem solving, computer vision, speech understanding, autonomous robotics, conceptual analysis of data bases, and intelligent agents [66].

The purpose of this section is to provide a general and personal overview of the machine learning area. In the following sections I aim to provide summarized descriptions of the different components of a learning system, focusing on the areas which are relevant to this research work.

2.1 Learning - In the Generic Sense

According to Michalski, the learning strategy or technique refers to the type of inference employed by a system during learning [30]. Based on this principle several strategies have emerged:

- Rote learning, which does not require any inference on the part of the learner and the learning system directly accepts information from the supervisor. Memorizing the multiplication table is an example.

- Learning from instruction (or learning by being told), where the learner performs some inference but is guided with background knowledge from the supervisor and the learning system in this case just reformulates the background knowledge. This is akin to a student who memorizes a formula or law and then goes on to understand its true meaning when she/he realises how it fits her/his prior knowledge.

- Learning by analogy involves the acquisition of new and useful facts or skills by transformation of existing knowledge which is similar to the desired concept and is a combination of induction and deduction requiring strong inference on the learner's part. Kekule's discovery of the structure of the benzene molecule after dreaming about a snake biting its tail is a good example of this aspect of learning.

- Inductive learning, which can be distinguished into *learning by examples*, the most investigated area of machine learning and *learning by observation and discovery.* In learning by observation and discovery the learner performs inference without the help of a supervisor (no examples to guide the inference process). Inductive learning from examples or supervised learning is the focus of this dissertation and will be extensively discussed in the following chapters.

- Discovery, both inductive and deductive learning in which an agent learns without help from a teacher. It is deductive if it proves theorems and discovers concepts about those theorems; it is inductive when it raises conjectures.

2.2 Learning from Examples

Inductive learning from examples can be technically defined as the process in which the learner produces a representation of a target mapping (e.g., a set of rules) using a learning strategy (e.g., heuristic search) from training information (e.g., positive instances) derived from some environment (domain) [66]. A more relevant definition for this thesis is that induction can be defined as a process by which a system develops an understanding of principles or theories that are useful in dealing with a domain by generating accurate models from the specific examples or instances presented to it. This includes the process of experimentation and discovery; that is, searching for hypotheses and then employing a principled metric evidence to confirm or deny their validity.

More formally, an example is a pair $(x, f(x))$, where x is the input and $f(x)$ is the output of the function applied to x. The task of pure inductive inference (or induction) is, given a set of examples of $(x, f(x))$, to find a

hypothesis h that approximates f.

Most inductive learning algorithms are *supervised.* That is, pre-classified examples are supplied by an external source to the learning algorithm. These examples typically consist of initial conditions and resulting actions or decisions (attribute-value vectors), thus similar to having a teacher who provides a correct set of answers. On the other hand, one can remove the teacher entirely and use the external world as a source of data. In *unsupervised* learning the learner is presented with unclassified data and is expected to form explanatory theories, such as classification schemes. Computational methods for learning from examples have been mainly distinguished into two families: *classification* when the number of classes or concepts is discrete and *regression* when the number of classes is continuous.

A Generic Model

Four questions are sufficient to design or distinguish a system that learns from examples.

1. What do you want to learn or, what is the nature of the learning task?
2. What is the nature of the existing background knowledge?
3. How do you want to represent the learned model?
4. What is your preference criterion for model selection?

By specifying the nature of the learning task we are explicitly defining the hypothesis (model) space. Our existing domain knowledge could consist of real/discrete valued attributes or both. The data could be erroneous. Every representational language defines a distinct partitioning of the instance space. It is thus important to employ a representational scheme that captures the

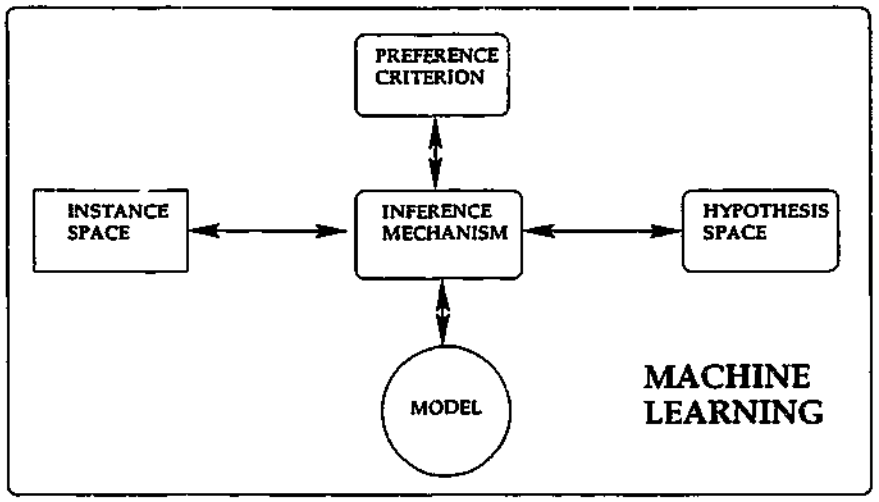


Figure 2.1: A Generic Learning System

intended model. Of equal importance is the question of model selection. Figure 2.1 presents a simple view of machine learning based on the above.

Based on this generic model, machine learning can be distinguished along three main branches, the underlying learning techniques implemented by the learning algorithms, the representation language used to present the knowledge acquired by the system and finally the application domain from which the training data has been extracted.

2.3 Knowledge Representation

Every learning system needs a language for representing knowledge acquired during the application of learning techniques. Thus machine learning can be classified on the basis of the representation language used. Several such knowledge representation formalisms exist such as:

- Linear and non-linear functions in various manifestations stand out as some of the earliest representation schemes. This dissertation presents

experiments with learning polynomial and Boolean functions from data.

- Production rules [64], which is one of the most commonly used formalisms and consists of condition-action pairs. For example, IF X THEN Z.

- Splines [28] are systems of piecewise polynomial functions with the pieces smoothly connected together.

- Decision trees [9] are directed graphs whose nodes are tests, branches are outcomes and the leaves represent the classes. An object or example is classified by tracing out a path from the root of the decision tree to one of its leaves after being subject to the conditions associated with the intermediate nodes if there are any.

- Frames [67], which are data structures for the representation of stereotypical (normal) situations.

- Formal logic-based representation [79], which uses formal logic expressions (e.g., predicate calculus) to represent acquired knowledge.

- Bayesian networks [76] are directed acyclic graphs representing probability distributions. The nodes in a Bayesian network represent random variables, and the arcs represent conditional probabilistic relations between the variables.

This research work primarily investigates the learning systems which use polynomial functions and Boolean functions as their knowledge representation schemes. Therefore discussion in the further sections will be restricted to these two specific formalisms.

2.3.1 Effect of Representation Language on the Instance Space

This section looks at the effect of different representational languages on the partitioning of the instance space. An instance or example described by n attributes corresponds to a vector in an n-dimensional Euclidean space. The learning task in this context is to find the function that best describes the training instances and partitions the instance space.

As mentioned earlier models partition the instance space based on their representation languages. Polynomial functions which are of primary importance to this research work produce curved partitions of the instance space. Decision trees and rules partition the instance space with hyper-planes. Other classifiers based on discriminant functions and K-nearest neighbour representation schemes produce polygonal partitions of the instance space. Figure 2.2 illustrates the partitioning of an instance space defined by the attributes X and Y, with '+' and '-' signs representing positive and negative instances; the dotted lines denote the partitioning of the instance space by the respective representation language.

2.4 Data Domains and Noise

In its early years machine learning was classified on the basis of the application domain in which it was applied. Several systems were developed that were specific to particular domains, like decision making, problem solving, image processing, task execution and, planning. But in recent years most machine learning systems developed have been independent of any domain. Although the systems based on different approaches are applicable to many domains, empirical comparison has shown that each performs best in some,

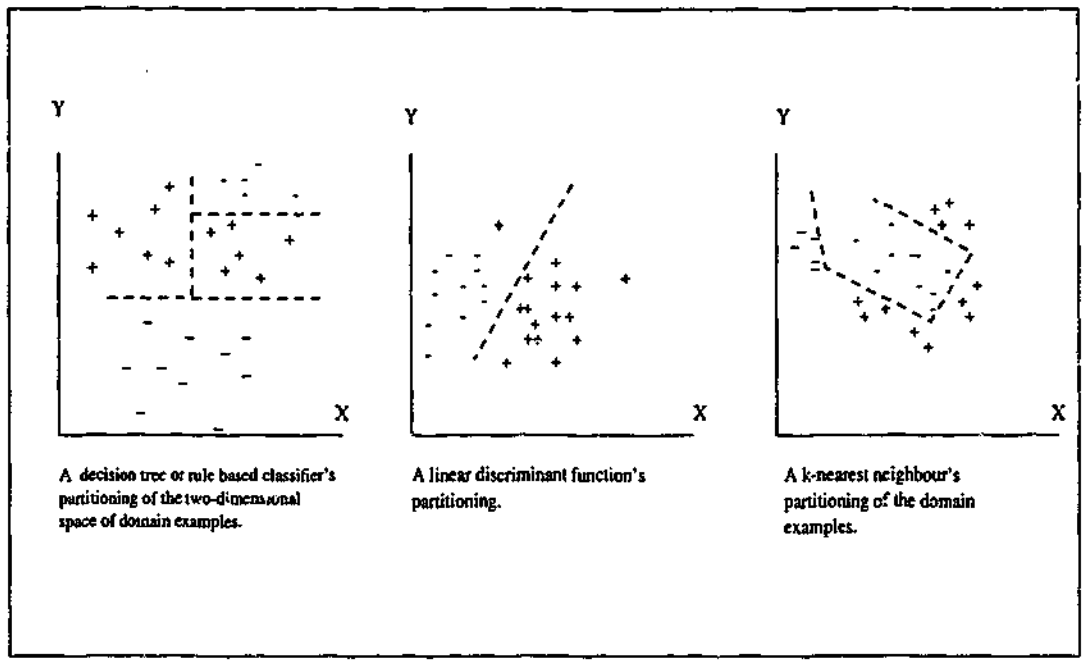


Figure 2.2: Effect of Different Representation Languages

but not all domains. This performance variation among different systems has been termed as the "selective superiority" problem [10]. As mentioned in the introduction, this raises the question of how to determine the suitability of a data domain to a particular inductive learning system, known as the meta-learning problem.

Supervised learning systems have also been classified on the basis of the type of examples supplied to the learner. This is because certain learners may use only *positive* examples (i.e., examples describing the target concept) while others may employ both *positive and negative* examples (i.e., examples that do not belong to the target concept).

Typically, the data to be analysed is a flat file containing a number of examples. In learning from examples or supervised learning, each example or instance is a vector of attribute values and a class representing the decision. Attribute values may be discrete or continuous. Unfortunately, real world data domains suffer from two major complications, *noise* and *missing*

attribute values. Noise is essentially caused by examples being described by incorrect attribute values and due to examples being misclassified (i.e. assigned incorrect classes). This often results in the inadequacy of attributes or generation of unnecessarily complex classifiers. Missing attribute values within data domains often occur due to discrepancies in data entry or data collection. Most systems therefore have to be modified in order to tolerate missing values.

2.5 Techniques in Computational Learning

This section presents a cross-section of approaches to inductive learning from examples.

2.5.1 Parametric and Nonparametric Regression

The most common approach to learning in statistical data analysis has been to fit a *global* parametric function $y = f(x, \theta)$, with finite length parameter vector θ to the data (x, y). The fitting procedure minimizes some error function over all the data pairs (x, y) in the training set so as to yield the optimal set of parameters θ. The criterion used to minimize is often a least squares criterion. If f is linear in the parameters θ, least squares regression corresponds to linear regression and the parameters can be calculated in closed form. In more interesting applications, however, f is nonlinear in the parameters and the error function must be minimized iteratively. Parametric methods are very successful if the assumed structure of the function $f(x, \theta)$ is sufficiently close to the function which generated the data to be modeled [92].

Nonparametric regression has no formal definition. The name nonparametric indicates that the function to be modeled consists of very large families of distributions which cannot be indexed by a finite-dimensional parameter vector in a natural way [47]. Hence, the nonparametric estimation of the input-output correlation is often called "distribution-free", implying that no assumptions about the statistical structure are made.

There are many methods in the statistical literature that can be used for flexible parametric and nonparametric modeling. These methods include:

- Polynomial regression [36]
- Fourier series regression [36]
- Wavelet smoothing [32]
- K-nearest neighbor regression and discriminant analysis [48, 83]
- Kernel regression and discriminant analysis [36, 83]
- Local polynomial smoothing [130, 36]
- Locally-weighted regression, or loess [21], is a procedure for estimating a regression surface by a multivariate smoothing procedure: fitting a linear or quadratic function of the independent variables in a moving fashion that is analogous to how a moving average is computed for a time series
- Smoothing splines (such as thin-plate splines) [36, 23]
- B-splines [36]
- Tree-based models (CART, AID, etc.) [9]
- Multivariate adaptive regression splines (MARS) [42]

- Projection pursuit [43, 83]
- Group Method of Data Handling GMDH [38]

2.5.2 Support Vector Learning

Support Vector Machines (SVM) are learning systems that use a hypothesis space of linear functions in a high dimensional feature space, trained with a learning algorithm from optimisation theory that implements a learning bias derived from statistical learning theory [24]. In very simple terms an SVM corresponds to a linear method in a very high dimensional feature space that is nonlinearly related to the input space. Even though we think of it as a linear algorithm in a high dimensional feature space, in practice, it does not involve any computations in that high dimensional space. By the use of kernels, all necessary computations are performed directly in input space.

SVMs deliver good performance in real-world applications such as text categorisation, hand-written character recognition, image classification, biosequences analysis, etc. Their first introduction in the early 1990s lead to a recent explosion of applications and deepening theoretical analysis, that has now established Support Vector Machines along with neural networks as one of the standard tools for machine learning and data mining.

2.5.3 Other Methods

Decision Tree Learning

Being robust to noisy data, decision tree learning, a method for approximating discrete-valued target functions, is one of the most widely used and practical methods for inductive inference. Benchmark systems that employ decision trees as the description language include CART [9] and C4.5 [80].

Instance-based Learning

Instance-based learning, a recent approach is based on the application of the similarity assumption. Instance-based learning algorithms assume that similar instances have similar classifications. As opposed to most other supervised learning methods instance-based learners don't construct explicit abstractions such as decision trees or rules [1]. In typical instance-based learning systems the entire training data or a subset is retained.

A new example is classified by finding the nearest stored example from the training data based on some similarity function, and assigning its class to the new example. Basically the performance of an instance-based learner depends critically on the metric used to compute the similarity between the instances.

Rule-based Learning

The covering technique for the induction of classifiers from examples has been a popular generic approach among rule-based induction systems. The significance of the covering strategy is that it forms a class description by constructing a rule that covers many positive examples, and few or no negative examples; it then separates the covered examples and starts again on the remainder.

Rule-based covering algorithms [64, 133] typically represent classification knowledge as a set of disjunctive logical expressions, one defining each class. Some popular systems under this class include systems like AQ [64], CN2 [20], and Foil [79].

2.6 Computational Learning Theory

Computational learning theory is a branch of theoretical computer science that formally studies how to design computer programs that are capable of learning, and identifies the computational limits of learning of machines. It provides a framework in which to precisely formulate and address questions regarding the predictive power and computational efficiency of learning algorithms. In this section I will summarize two well-known approaches and expose their shortcomings.

2.6.1 Learning in the Limit

The first theoretical study of machine learning commenced with the formal definition of learning given by Gold [44]. In Gold's model, the learner receives a sequence of examples and is required to make a sequence of guesses as to the underlying concept such that this sequence converges at some finite point to a single guess that correctly names the unknown rule. Gold's model avoids the question of computational feasibility.

In Gold's framework a learning algorithm is viewed as having successfully solved a learning problem (in an infinite instance space) if, after seeing a certain sequence of training pairs, it settles on the correct hypothesis and never changes its mind thereafter. The learning algorithm in this case is said to have identified the target concept in the limit.

2.6.2 PAC Learning

More recently, Valiant [109] introduced the PAC (probably approximately correct) learning model. In this model a learning mechanism is said to have

learned a concept if the hypothesis it has settled on returns an output which is probably approximately correct for any given input.

A class of concepts C is efficiently PAC-learnable if and only if any concept $c \in C$ can be learned from m training examples within error bound $0 \leq \epsilon \leq 1$ with reliability $0 \leq (1-\delta) \leq 1$, where m is a polynomial function of ϵ and δ.

According to Chris Wallace [114], the theories of computational learning outlined above have provided little insight into human or machine learning. Some apparent limitations of their approach are:

- In both frameworks learning is possible only if the necessary model is known to be in a restricted set.
- Neither of the theories provide inherent model-selection.
- There is only a limited consideration of noisy or erroneous data.
- The only well-proven inductive theory, statistical inference, is neglected.

2.6.3 Bias in Machine Learning Systems

Bias is an essential component of every supervised concept learning system. The term bias embodies a multitude of definitions. Mitchell [68] refers to bias as the choices made in designing, implementing and setting up an inductive learning system that lead the system to learn one generalization instead of another. A more general definition of bias is defined by Utgoff as the set of all factors that collectively influence hypothesis (model) selection, given a set of training instances.

A 'model', 'concept description', 'hypothesis' and 'classifier' are synonymous with a function that partitions a domain of instances into those that belong to the class (concept) represented by the function and those that do

not. In revision, the objective of every concept learning system is to infer a model from a set of known examples and counterexamples of the target concept, that describes the target concept. Learning from examples can be described as a function of the training instances and the induction principle for hypothesis preference [105].

Need for a Principled Preference Criterion in Model Construction

For most domains, examining the entire space of possible hypotheses is either infinite or computationally intractable. Furthermore, any learning system that learns concepts from examples has to make a satisfactory selection from among the available hypotheses [105]. Finally, the presence of noise in the data further complicates the accurate selection of models. Therefore to find an accurate hypothesis in a reasonable amount of time, learning algorithms employ a restricted hypothesis space bias and a preference ordering bias for hypotheses in that space.

Characteristics of Bias

As mentioned earlier, bias is a set of non-empirical factors that collectively influence the selection of hypothesis. Although the training data introduces its own implicit bias the techniques needed in determining the extent to which it affects the learning task are yet unknown. Two major sources of bias are the *concept description language* and the *concept learning algorithm.*

Specifically, factors constituting bias include, the representation language in which the induced hypotheses are described, the search space of possible hypotheses that the algorithm needs to consider, and the preference criteria that defines when a search procedure should stop searching for better hypotheses [106].

Utgoff [105] characterises bias along the dimensions of strength and correctness:

- A *strong* bias is one that enables the learning system to focus on a relatively small number of hypothesis. Conversely, a *weak* bias is one that allows the learning system to consider a relatively large number of hypotheses.

- A *correct* bias is one that allows the concept learner to select the target concept while an *incorrect* bias is one that prohibits the concept learner from selecting the target concept.

Both these characteristics imply that a strong and correct bias simplifies the concept learner's task by allowing it to easily induce the target concept while a weak and incorrect bias make the learning task extremely difficult by not providing a proper guidance regarding hypothesis selection.

Kinds of Bias

Other researchers have categorized bias in different ways. In addition to Utgoff's *strength* and *correctness*, Rendell [82] differentiates between *exclusive* and *preferential* bias. An algorithm having an exclusive bias against a certain class does not consider any of the concepts in that class while a preferential bias influences an algorithm to prefer one class of concepts over another. Many researchers distinguish bias as *representational* and *procedural* and most of the bias implementations fall under these categories. Both representational and procedural biases are examined in detail within this section.

Representational and Procedural Bias

The states in a search space of hypotheses are defined by a *representational* bias [45]. In other words, a representational bias constrains the rep-

resentational language from considering certain concepts (since they cannot be expressed in the language), and in this sense it is a form of an *exclusive* bias. A representational bias defines the set of states in the search space by specifying the language (e.g., Polynomial functions, Boolean functions), its implementation (e.g., the classes of functions searched), and a set of primitives (the admissible features, their types and their range of values).

In view of Utgoff's framework, if a representational bias is *strong* (i.e. the language for expressing the classifiers is powerful) the space of hypotheses becomes small while a *weak* representational bias implies a large hypothesis space. A *correct* representational bias defines a hypothesis space that includes the target concept while an incorrect bias fails to do so.

The order of traversal of the states in the hypothesis space defined by a representational bias is determined by the *Procedural Bias* or *algorithmic Bias* [45]. Procedural bias is a form of preferential bias. Common examples of procedural bias include the beam width in a beam search and preference for simpler hypotheses (e.g., Minimum Message Length principle(MML) [122, 127]) and Occam's Razor [7].

A procedural bias thus helps to restrict the space of hypotheses through which the learning algorithm conducts its search.

Static and Dynamic Biases

Biases have also been classified on the basis of their implementations within learning programs. Bias in a learning program is *static* if it remains fixed throughout the learning process. A *dynamic* bias can be altered during the learning process. Most of the machine learning systems primarily use a static bias. Examples of programs using a static bias include Quinlan's C4.5, Michalski's STAR, Clark and Niblett's CN2. Utgoff's STABB (Shift to a Better Bias) system and Brodley's Model Class Selection (MCS) system are

examples of systems that embody a dynamic bias.

Limitations of Bias In Model Selection

Searching the entire space of possible hypotheses for any domain has been a daunting task. As consequence learning algorithms employ a restricted hypothesis space (representational) bias and a preference ordering (procedural) bias in order to find an accurate hypothesis in a reasonable amount of time. Empirical comparisons among algorithms illustrate that no single bias exists that performs well for all learning tasks. Each algorithm is found to be better than the other for certain kinds of tasks.

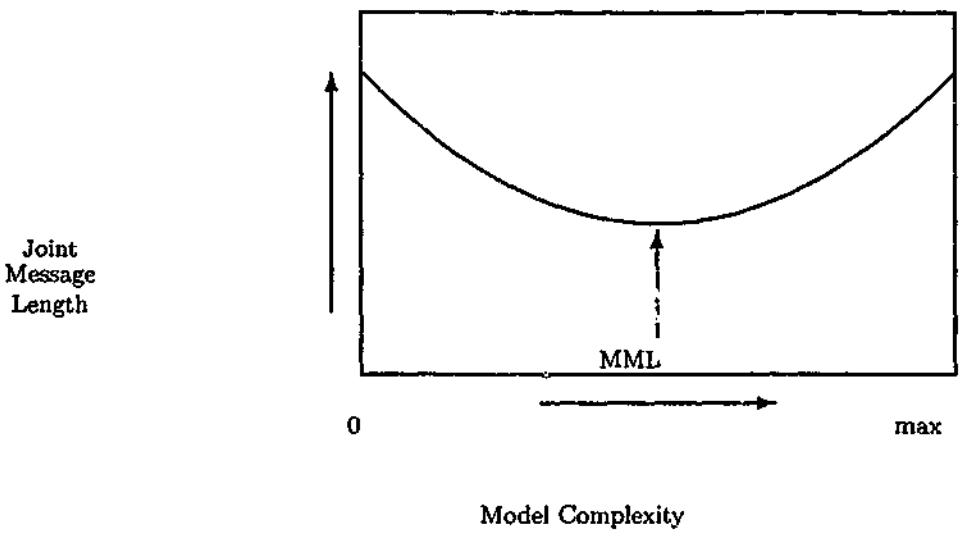
Furthermore, selecting an appropriate bias is complicated since different learning tasks or subtasks are suited to different representation languages. For example, if the learning task represents an instance space that is obliquely partitioned, a classifier based on an axis-orthogonal partitioning would perform poorly. Another unexamined area is that of bias interaction. A representational and procedural bias may interact with each other in synergy or conflict.

2.7 Minimum Message Length Inference for Model Estimation and Selection

The Minimum Message Length principle (MML) [122, 127, 125] is a Bayesian information-theoretic approach for the estimation and selection of models. Given some data and optional prior knowledge, the most probable theory or explanation for the data according to MML is the theory that has the shortest encoding of the theory and data combined. Consider the example of the Identi-Kit system which contains pictures of facial parts that are used to

construct facial images. By providing a small number of "standard" chins, mouths, and eyes enormous data compression is achieved as opposed to storing actual photographs. The standard components are models abstracted from previous examples (the collection of facial features). If a new example (such as a cat's face) is sufficiently different from all-existing examples, then there may be no compact encoding in the Identi-Kit system. It may be simpler to describe the new example directly. In such cases, MML indicates that the new example is very surprising relative to abstractions over the previous examples [16].

Let us consider a fundamental issue in inductive learning. Given that one has a number of different theories (models) for a set of data (observations), how do we select the "best" theory. It is well known that one can find a sufficiently complex theory to fit exactly any finite set of observations. For example, a set of N points can always be fitted by a polynomial of order N. However, if there is measurement error in the collection of these points then we are in fact fitting the noise in the data. The solution to this problem of *overfitting* is to find a suitable trade-off between the complexity of a model and its fit to the data. An overly complicated model may fit the data well but offers very little predictive value which a model with little structure does not explain the given data sufficiently. MML, and the similar Minimum Description Length (MDL) [84] principle, aims to achieve this trade-off.



The picture above shows how MML avoids overfitting . A highly complex model increases the encoding length of the model while an overly simple model increases the cost of encoding the data.

How to apply MML

- Provide a two-part message describing a given set of data, $M(\theta) + M(D|\theta)$.
- These optimally encode model parameters θ and, data D assuming model to be true. (We only need to calculate the message lengths, rather than actually produce the messages.)
- Specify real-valued parameters to precision δ where $\delta = \frac{1}{\sqrt{k_d F(\theta)}}$. $F(\theta)$ is expected "Fisher information" and k_d is a dimensionality constant.
- Compute the message length using $-\log g(\theta) - \log f(x|\theta) + 0.5 \log F(\theta) - 0.5 \log 12 + 0.5$ where $g(\theta)$ is the prior density on θ, $f(x|\theta)$ is the likelihood of data x given θ and $F(\theta)$ is the "Expected Fisher Information".

- According to the MML principle, the best model is one that enables the shortest two-part encoding of the model and the data, given the model. That is, Best model ⇒ Maximum two-part data compression.

The MML principle has been employed to a variety of domains. Recent MML work includes single factor analysis [128], multiple factor analysis [119], von Mises circular distributions [124, 126], causal modelling [11], spherical Fisher distributions [34], mixture modelling [126, 121, 35], segmentation of a binary sequence [116] and general surveys [127, 125].

2.8 Recapitulation

This chapter provides a review of the literature relevant to this research. We have reviewed the machine learning field, classifying it across different dimensions. Issues of special importance, for example, the effect of the representation language on the instance space were included in order to provide an understanding of the problem area.

Chapter 3

Machine Discovery

From 1600 to 1619 Johannes Kepler, working on astronomical data accumulated by astronomer and nobleman, Tycho Brahe, published in sequence the famous laws of planetary motion. Kepler's laws allowed the positions of planets to be predicted with accuracies ten times better than Ptolemaic or Copernican models. They also set the stage for Newton's discovery of universal gravitation. The tortuous way in which Kepler approached his insights is well documented. Inspite of making these discoveries, Kepler was still committed to his older erroneous spheres theory which made more sense to him at that time. Kepler's scientific achievement was immense, as he had uncovered the fundamental regularities of planetary motion and thus set the stage for Newtonian physics. However, Kepler's exploits seem to undermine the idea that the process of learning – individual or communal – might be carried out in any automatic fashion. If the story of Kepler tells us anything, it is that unpredictability and audacity are the key [104].

The processes of discovering new scientific laws and concepts have been a topic of AI research spanning several decades. Research in this area has been directed at two goals: namely, understanding human scientific reason-

ing, and secondly developing systems which autonomously, or collaboratively with the expert, contribute to scientific knowledge. These independent yet complementary goals have been intermingled in a way such that ideas derived from research on human scientific discovery have been applied in developing systems while the design of discovery systems has provided insights into the cognitive processes employed by humans [97].

As stated by Herbert Simon, scientific discovery entails a variety of processes. Research problems must be identified and representations that describe them and their potential solutions must be formulated. Data must be gathered by making observations and carrying out experiments. Instruments must be developed for carrying out observations and experiments. Regularities and patterns, that is to say, generalizations (laws) and concepts, must be sought to describe the data parsimoniously; once the initial laws have been formed, the observations are often then explained at a more detailed level where each of the substages are sometimes given a causal explanation. Laws and concepts must be generalized for use in creating more comprehensive theories. Inferences must be drawn from theories and empirical predictions made that can be tested by new observations and experiments [97, 141].

The machine learning methods outlined in the previous chapter provide for generic analysis of data. These methods often form the core of automated discovery systems. However, creative discovery entails far more than just a data-driven analysis. This chapter thus explores the process of discovery from psychological and computational perspectives. A study of computational discovery systems is offered.

3.1 Philosophical Aspects

The philosophical aspects of scientific discovery are depicted in the following historical survey based upon [70]. In the first half of the century, most philosophers concentrated on the logical analysis of science and set aside the problem of discovery. Logical positivists like Reichenbach distinguished the context of discovery from the context of justification and excluded the former from their logical analysis. Although Popper criticised their deductivism and suggested systematic falsification as an alternative, he also left the elucidation of discovery to psychological inquiries. The problem of scientific discovery was finally addressed by the reaction to logical positivism in the 1960's. N.R. Hanson introduced the method of "retroduction" on the basis of Peirce's logical theory. Thomas Kuhn's paradigm theory located (fundamental) discoveries in the context of scientific revolutions and emphasized the function of metaphor in scientific discoveries. Metaphorical use of existing terms and concepts to overcome theoretical difficulties often plays an important role in developing new ideas.

3.2 Creative thinking: Psychological Aspects

Many people have studied the phenomenon of creative thinking. Some psychologists devote a substantial part of their careers trying to understand what distinguishes the creative person from the individual who rarely, if ever, comes up with a new idea. There are no definitive answers, but there are at least some indications of factors that appear to affect our creativity. And as Margaret Boden puts it [8], creativity seems to materialize from composers to chemists, cartoonists to choreographers. At the same time, creativity seems puzzling and mysterious, and often inventors, scientists, and artists fail to ex-

press how their original ideas arise. While intuition is often credited, nobody can explain how it works. Thus, the apparent unpredictability of creativity seems (to many people) to rule out any systematic explanation, whether scientific or historical.

3.3 Models of the Creative Process

Psychological literature presents a variety of models describing the creative process. These are not rigid models; the steps may come out of order, be skipped entirely, or be revisited.

3.3.1 Wallas model - the psycological perspective

The early twentieth-century reformer Graham Wallas, got somewhat nearer the source of the creative process, which he outlines in his book, *The Art of Thought* [129]. Summarising his own and other people's work in this area, Wallas described four stages of creation. This is a four-step model originally proposed in 1962.

1. **Preparation**: In the preparation stage, we define the problem, need, or desire, and gather any information the solution or response needs to account for, and set up criteria for verifying the solution's acceptability. The person expecting to gain new insights must know his field of study and be well prepared. And yet, although a certain threshold level of knowledge seems necessary for creativity, creative breakthroughs are not always the product of the most expert thinkers in a discipline.

2. **Incubation**: In the incubation stage, we step back from the problem and let our unconscious minds contemplate and work it through. This

is the formative stage which takes place while the mind is relaxed and the individual is engaged in an unrelated activity which frees up the conscious and perhaps unconscious mind to be in a receptive mode.

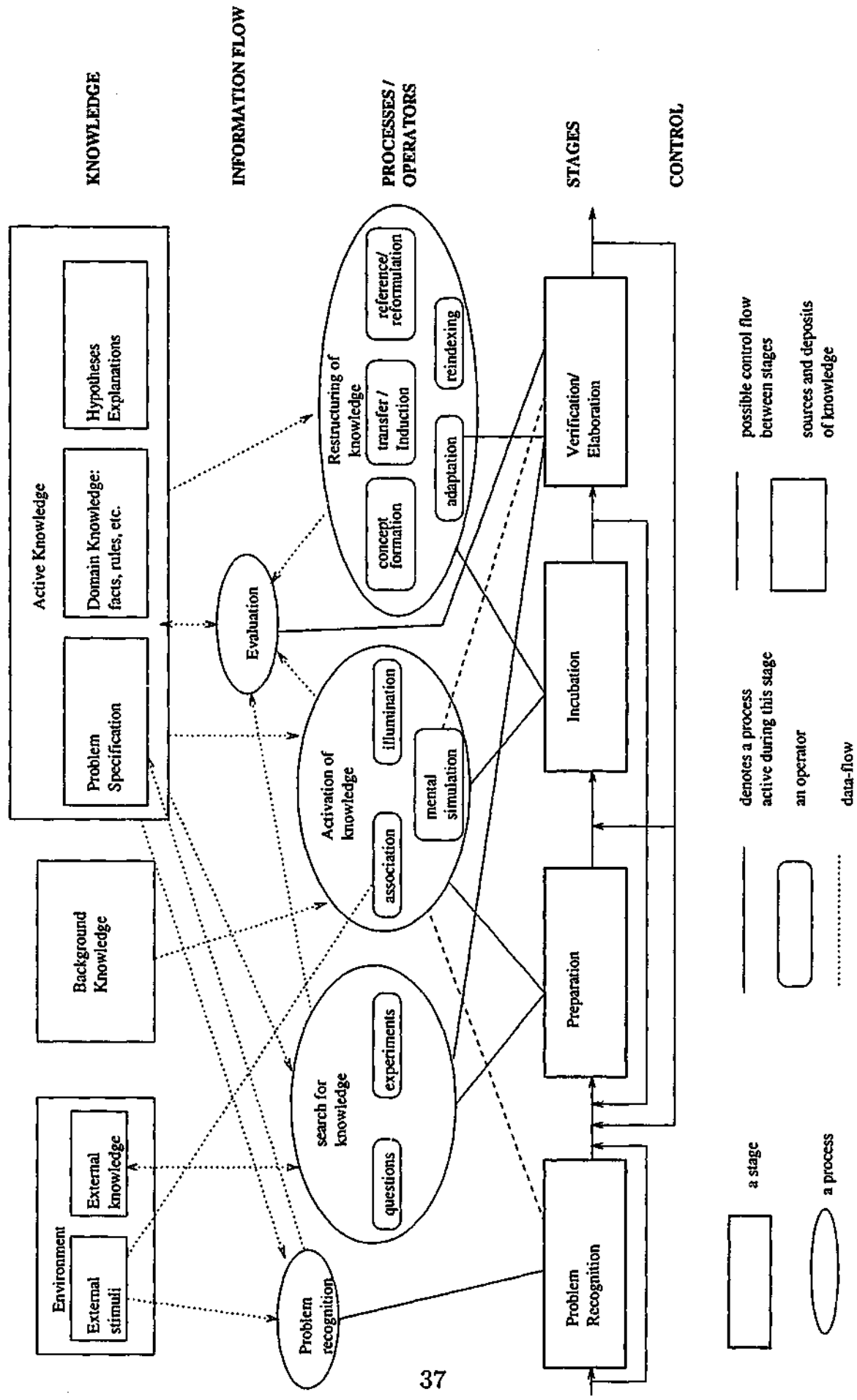
3. **Illumination**: In the illumination stage, ideas arise from the mind to provide the basis of a creative response. This is the conscious recognition of the new idea, the so-called "Eureka" phenomenon. These ideas can be pieces of the whole or the whole itself, i.e. seeing the entire concept or entity all at once. Unlike the other stages, illumination is often very brief, involving a tremendous rush of insights within a few minutes or hours. This was certainly the experience of Archimedes when he got his idea of displacement in the public bath.

4. **Verification**: In verification, the final stage, one carries out activities to demonstrate whether or not what emerged in illumination satisfies the need and the criteria defined in the preparation stage. In this final step, efforts are made to see if the "happy idea" actually solves the problem. Since "great" ideas don't always work out in actual practive, this final step is vitally important to the success of any project.

3.3.2 An Integrated Model of Creativity

Klaus Schmid [95] presents an integrated model of the creative process with reference to the wallas model. An adaptation of the model is presented in Figure 3.1.

Figure 3.1: Creativity with reference to the Wallas model



3.4 Computational Discovery

As early as 1960, Hoveland and Hunt presented a system that tried to simulate human concept learning by inducing simple concepts from examples. Three decades of research has spawned a huge variety of automated discovery systems. While some aim to replicate historical discovery others offer novel contributions to specific domains. According to Valdes Perez [108], the goal of discovery is to find knowledge that is novel, interesting, plausible, and understandable. Thus, a discovery program that too often leads to familiar, dull, wrong, or obscure knowledge won't be used.

Automated discovery systems can be distinguished on the basis of whether the discovery process is data-driven or theory-driven. In parallel, they can also be distinguished by the method of reasoning involved, namely, induction, abduction or deduction. The primary distinction between scientific discovery and other forms of learning is that while learning defines a mapping between a language and the environment, scientific reasoning extends learning by defining newer languages.

3.4.1 Generic Steps in the Discovery Process

Scientific Discovery involves some or all of the following:

1. Discovery of Empirical Laws
2. Qualitative Modeling
3. Discovery of Taxonomies / Concept Hierarchies
4. Discovery of Structural Models
5. Discovery of Process (Temporal) Models

3.4.2 The Learning Framework

When making a comparative study of discovery systems several characteristics may be observed.

Search/solution Method

Since discovery-oriented workers (discoverers, for short) undergo a long apprenticeship to become experts in their field, one might conclude that a discoverer's reasoning process resembles expert reasoning that is based on recognition, for example, the chess grandmaster who chooses a strong move after a glance at the board , or the physician who quickly selects a likely diagnosis based on the first few symptoms. If discovery is like such expert reasoning, then pattern-recognition approaches such as neural nets or discrimination trees - that match a current problem against previous experiences - would be the methods of choice for computerizing most discovery tasks. However, by definition discovery happens at the frontiers of knowledge where nobody is an expert, so the better analogy is to chess beginners or to physicians in-training, whose reasoning is based partly on trial-and-error, or heuristic search, as it is known in artificial intelligence. The basic idea of heuristic search is that the solving of many reasoning tasks can be well viewed as a search within a large combinatorial space. We say the space is combinatorial because at each choice point there are many choices, and this often results in the creation of notoriously large spaces. In many practical tasks, people cannot search the entire space in their heads, so we must focus on some subspaces in preference to others. The knowledge that permits such focusing is called heuristics. Heuristics can be absolutely reliable or they can be rules-of-thumb; the key point is that they direct one's scarce resources toward more promising avenues. In chess, for example, a beginner's heuristic is "consider moves that

check the opposing king first".

Data-driven versus theory-driven discovery

A program or approach is theory-driven if it uses relatively general knowledge (including knowledge of how to search combinatorial spaces) as a main source of power. A data-driven program instead relies on specific measurements, statistics, or examples. Most approaches are some combination of both. A same task might be approached in either way: consider that a chess program could base its next move either on matching against a large database of examples of previous positions and good moves, or on general heuristics about strategy and tactics combined with a search over possible successive moves. The input to a data-driven program can be megabytes of data, whereas the input to a theory-driven program could be a mere few lines. Depending on the specifics of the problem, either might take longer to run.

Assessing machine discoveries

Often, the generation of models is the easy part of automated creativity in science. Programs can produce a plethora of models which satisfy the criteria for a solution to the problem, and some pruning of the models is necessary to save the user from having to deal with a deluge of information. As the implementation of a discovery program progresses, the pruning measures can often evolve into rules to guide heuristic search, and can sometimes specify the search space itself. Hence, how programs assess their results internally is an important part of machine creativity in science.

3.4.3 Discovery Systems

Numerical Law Discovery

Numerical law discovery systems try to tackle the problem of model discovery when the hypothesis space is not well defined. Systems under this family can be categorised into three types based on the nature of search involved. These include the heuristic, exhaustive and contextual approaches. After a brief definition of all approaches I present a table of discovery systems belonging to the different categories.

The *heuristic* approach attempts to employ heuristics able to guide this search efficiently, and tries to simulate human experts solving similar problems. These simple heuristics help to build the model iteratively and significantly reduce the size of the search space. The main principle in this category of systems is to discover the law step by step.

The second, and more recent, *exhaustive* approach does not attempt to discover the law by decomposing it, but explores exhaustively a solution space bounded by some parameters. This approach relies on brute computational force and yields no cognitive explanation. It is based on statistical regression and a law is simply an instantiation of a regression model.

The final approach, which I call *contextual*, solves the problem of model selection by constraining the search space by specifying a list of prototypical functions. The prototypes can also be considered as background knowledge. Thus the goal is to discover laws by efficiently search a constrained space.

A Review of Various Numerical Discovery Systems

In general any discovery system can be distinguished on the basis of the problem/hypothesis space, the search technique and the evaluation/model-selection criterion. Current function finding systems using regression can be

loosely categorised into three families. In the first group are BACON-like systems such as ABACUS and ARC.2 which use heuristics in an iterative fashion to search the problem space. The second group of systems such as Equation Finder and LAGRANGE, perform systematic regression over a large space of models using some statistical criteria to guide the search/model selection. Finally, systems like E* and KEPLER only consider a pre-defined set of models. Here I provide a brief summary of some of these popular discovery systems.

BACON

The BACON system by Langley et al. [58, 75] iteratively builds models using heuristics to guide/constrain the search. Given two variables, BACON looks for a constant or linear relationship between them. When unsuccessful, it tries to define higher order terms and continues the regression. In its test for constancy it uses pre-defined error values in order to create intervals for measuring constancy. It also allows users to specify some arbitrary functional forms and then tweaks the parameters of the functions until the correlation between the predicted and actual values of the dependent variable is strong.

BACON can also find complex relations between more than two variables. Given some data with four variables, BACON starts by varying one variable at a time and keeping the other variables constant. It finds simple relations between different combinations of the four variables which the user can then put together. BACON can also postulate intrinsic properties.

BACON has very limited noise-handling capabilities and uses user-defined tolerance parameters that allow it to deal with inexact laboratory data. According to Cullen Schaffer [94, 93], relevant information external to the data was communicated to the program (for example, the noise tolerance thresh-

olds) and this information, rather than the data, principally accounts for its discoveries.

ABACUS

The ABACUS [37] system of Falkanheiner is a direct descendant of BACON, but incorporates some clustering. Given a set of data measuring some variables ABACUS builds different clusters. In a manner similar to BACON, it studies the monotonic dependencies between variables and produces arithmetical combinations of the initial variables (for example x/y and x*y). The system studies the dependency only on groups of examples such that all the variables but the two under study remain constant. The set of these groups of examples is called the projection on the variables. The new combinations of the initial variables are then sorted according to an evaluation function and are separated into 2 sets: active nodes and suspended nodes. The process is repeated iteratively until a combination has a constancy higher than the required threshold constancy. The constancy is defined as the percentage of examples which have the same value for this variable or which can be clustered according to their value.

ABACUS deals with noise through a user-set parameter which represents the maximum relative error of any numerical value, and by default equals 2

ARC.2

The ARC.2 system [69] of Majourie Moulet is the similar to ABACUS. BACON has a major limitation of requiring the user to order the variables and to carry out a potentially large set of experiments. ABACUS drops these constraints, but remains limited to a hypothesis space which is a subset of polynomials. In particular, ABACUS cannot find polynomials with real pa-

rameters since it replaces regression by + and - operators. ARC.2 employs a method that merges BACON and ABACUS. Thus, in ARC.2 the model is built iteratively by generalising an n-dimensional relation to an n + 1 dimensional one, using a less severely constrained dataset like ABACUS does. Thus, ARC.2 extends BACON and ABACUS by incorporating polynomial regression and using the clustering advantage.

Equation Finder

The EF system [138, 135] of Zytkow automatically explores a space of models until it finds a satisfactory one. EF only applies to two numerical variables. Its input is a list of triples (x,y,g) where g is the uncertainty of the dependent variable. First, EF generates the list of all polynomials of the form y = P(x) up to the given degree limit d. If none of the models satisfy the given statistical criterion, x and y are transformed in order to create new terms using operators such as log, exponentiation, square root and inverse and multiplication and division. This is iteratively repeated until the best model based on the statistical criterion is found. The statistical criterion is subjected to a chi-square test and each best-fit model is assigned a probability that its chi-square p value would have been reached by chance. And finally a model is acceptable if its Q value is higher than some pre-defined Q value.

E*

The E* system [94, 93] of Cullen Schaffer only considers two variables and eight possible functional forms, the linear relationship $y = kx_i + k_2$, six power proportionalities, $y = kx^n for\ n \in \{-2, -1, -0.5, 0.5, 1, 2\}$ and the "no relationship" case. For each of the six proportionalities E* computes a measure of fit, which is basically a correlation function. It selects the proportionality

with the best MF. The system also note the ration of the model with the highest MF with the next best. The system then looks at $y = kx_i + k_2$ for the best model and computes k2. It then calculates the t-statistic for k2. Using the D and t values the systems decides on further searching. The following table offers a chronological listing of numerical law discovery systems.

Author	System	Year	Type
D. Gerwin	MODEL	1974	Heuristic
Langley et al.	BACON 1-6	1986	Heuristic
Falkenhainer et al.	ABACUS	1986-1990	Heuristic
M. Moulet	ARC.2	1993-1994	Heuristic
Zembowicz et al.	EF	1991-1992	Exhaustive
J. Zytkow	FAHRENHEIT	1990	Exhaustive
Dzeroski et al.	LAGRANGE	1993	Exhaustive
Y. Wu	DISCOVER.2	1988	Contextual
C. Schaffer	E*	1990	Contextual
Y. Wu & S. Wang	KEPLER	1991	Contextual

Mathematical Conjecturing

Mathematics comprises various activities. The end products are familiar: things like calculus books used to train engineers in various abilities useful in their work. The fundamental results in any mathematical text include its theorems, mathematical propositions that have been proved. The existence of a theorem rarely begins with the last line of a proof. Rather a mathematician conjectures that some mathematical proposition is true or it is not. He, or other mathematicians, may try and prove or disprove the proposition. In

working on a proof, a mathematician may introduce a new concept in the belief that it will assist his work. Conjecture-making, theorem-proving, and concept formation are three of the central activities in mathematics and the ones researchers have attempted to automate. Much less work exists in the other two areas. In automated conjecture-making, perhaps two dozen papers have been published. Researchers in this field have written the following programs. (The researcher's name is followed by the name of the program and then by the year of its first published description.)

Authors	System	Year
D. Lenat	AM	1976
D. Lenat	EURISKO	1983
E. Morales	DC	1985
K. Haase	CYRANO	1986
S. Fajtlowicz	GRAFITTI	1986
W. Shen	ARE	1987
S. Epstein	GT	1987
M. Sims	IL	1990
R. Bagai et al	GTP	1993
S. Colton et al.	HR	1999

Taxonomy and Concept Formation

Concept discovery systems can be categorized into three groups, based on their degree of autonomy. Concept learning systems are "supervised" by a teacher and restricted by the assumption that all examples are covered by one concept. Conceptual clustering systems are "unsupervised" and autonomously partition a set of examples using multiple concepts. Exploratory approaches are not necessarily guided by examples, and must explore the

incoming data or hypothesis space autonomously.

UNIMEM The UNIMEM system of Lebowitz [61], like COBWEB, addresses the problem of incremental conceptual clustering in an attributevalue hypothesis space. UNIMEM has an interesting evaluation scheme for concepts, based on a predictability analysis [60]. Similar to COBWEB, this is based on how well inferences about missing features can be made using a discovered concept. For each feature (unary literal) in a concept, a confidence score says how many times that feature had been confirmed or disconfirmed by examples in the past. Wasserman's MERGE system [132] extends the UNIMEM system to deal with multilevel structured objects and binary relations.

CLUSTER/2 As discussed earlier, the CLUSTER/2 system of Michalski and Stepp [65] transforms the abstract conceptual clustering task into a series of supervised learning problems. Unlike all of the systems surveyed above, CLUSTER/2 uses background knowledge. This is expressed as a network of determination rules between attributes. Examples are not completely saturated with the background knowledge, rather, the network directs the partial saturation process by inspecting the set of attributes present in an example. The intuition behind this restricted saturation process is that not all background knowledge will be relevant to a given clustering task [26] . CLUSTER/2 uses a complex evaluation function to drive its search over clusterings: this is roughly based on the simplicity of cluster descriptions and the fit of the clustering to the examples.

Qualitative and Structural Models

Discovery of hidden structure has been the subject of various case studies in the field of chemistry that led to the development of a several discovery

systems.

Much of chemistry is concerned with the following questions:

- What substances exist in nature (elements, atoms,...), and in what structures are they to be found (molecules, polymers, hydrates, ...) ?
- What are the properties of these substances, and structures ?
- What are the mechanisms for combining/breaking apart such substances, structures and sub-structures ?

Previous work in computational scientific discovery has addressed how each of these types of chemical knowledge can be used to elucidate the others. For example, the STAHL and DALTON systems [144, 143, 75] use the concept of chemical reaction to determine the components of a substance (STAHL), and its molecular composition (DALTON).

Related systems are Rose and Langley's REVOLVER [89, 90], BR3 by Kocabas [52] and Fischer and Zytkow's GELL-MANN [140, 142]. These systems all use a different mechanism, the use of collisions to produce sub-atomic particles, to establish quark models of the fundamental particles in physics. Unlike the previous systems, which used only very general heuristics, REVOLVER uses domain specific knowledge in evaluating the models that it generates and deals with revision of beliefs about hidden structure. GELL-MANN uses an additivity principle (in which properties of an object must be the sum of the contributions of the structures from which it is formed) and a combination and conservation principle (in which the same fundamental structures must appear on either side of a reaction) to generate and verify models.

BR3 is unique in proposing new quantum properties, together with a general conservation of properties principle, to account for observed particle

reactions. BR3 demonstrates how hidden properties can be postulated for observable objects. Each of the systems just described assumes that a single mechanism is operating for combining objects, structures or substructures or breaking them apart. In STAHL and DALTON, the mechanism was chemical reaction, in the other systems the mechanism involved the use of high energy collisions to produce sub-atomic particles.

Process Models

A recurring problem in catalytic chemistry has been to formulate the sequence of steps, known as the reaction pathway, for a given chemical reaction. In addition to the reactants and products of the reaction, this inference may also be constrained by information about intermediate products, concentrations over time, relative quantities, and many other factors. Even so, the great number of possible pathways makes it possible that scientists will overlook some viable alternatives, so there exists a need for computational assistance on this task. Valdes Perez [107] developed **Mechem** with this end in mind. The system accepts as input the reactants and products for a chemical reaction, along with other experimental evidence and considerable background knowledge about the domain of catalytic chemistry. Mechem lets the user specify interactively which of these constraints to incorporate when generating pathways, giving him control over its global behavior. The system carries out a search through the space of reaction pathways, generating the elementary steps from scratch using special graph algorithms. Search always proceeds from simpler pathways (fewer substances and steps) to more complex ones. Mechem uses its constraints to eliminate pathways that are not viable and also to identify any intermediate products it hypothesizes in the process. The final output is a comprehensive set of simplest pathways that

explain the evidence and that are consistent with the background knowledge. This approach has produced a number of novel reaction pathways that have appeared in the chemical literature.

3.5 Hybrid Models for Integrated Discovery

3.5.1 IDS

Nordhausen and Langley's IDS system [71, 72] takes as input an initial hierarchy of abstracted states and a sequential list of "histories" (qualitative states, see Hayes [49] or Forbus [41]). Using each history IDS modifies the affected nodes of the abstracted state tree to incorporate any new knowledge gained from that history. Its output is a fuller, richer hierarchy of nodes representing history abstractions. Qualitative and quantitative state laws are grouped under individual nodes, and, transition orderings, transition conditions and transition constraints are placed between nodes.

The initial hierarchy of abstracted states represents states of the system under study. Knowledge is in an is-a hierarchy where nodes may have more than one parent.

Knowledge of the system under study comes to IDS as histories, which are sequences of qualitative states that the system under study is in. Each state specifies values of variables of the system under study that are held for the duration of the state. The boundaries between states are given by sign changes of a derivative of a system variable.

Upon seeing the next state in the sequence, IDS walks from the tree's root to its leaves and makes the state a leaf of an abstracted state. The hierarchy may be modified during the traversal by adding a new abstracted state if

the new instance will not fit under existing ones. Similar states may also be merged.

All nodes in the hierarchy have one or more successor links that tell how one state is changed into another, what must be true for the transition to happen, and, constraints that hold between the transitions. States also hold constraints that are valid within that state which are used to define that state. Inter- and intra-state information is updated when new states are incorporated into the tree. The information may be in the form of equations or conjunctions of predicates.

IDS can predict a sequence of states and give qualitative and quantitative information about them given an initial state and a learned annotated state hierarchy.

3.5.2 ABACUS

The ABACUS system [37] aims to integrated quantitative and qualitative discovery. Specifically, given data consisting of numeric and possibly also symbolic characterizations of some phenomenon (an object, a process, a system), ABACUS will generate mathematical equations characterizing this phenomenon and qualitative conditions under which these equations apply. These equations can then be used for predicting the behavior of this system or process. For example, given data characterizing an electric circuit (voltage, current, resistance, and any other relevant or irrelevant properties, ABACUS will generate the Ohm's Law)

3.5.3 49er

Zytkow and Zembowicz's 49er.b [136, 137] was designed as a general knowledge discovery tool that can find regularities in scientific and other domains.

It has a two stage search process for regularities. The first stage, done by module 1, looks for contingency tables between two attributes to be compared. The second stage, done by module 2, uses the contingency tables to look for other regularities like equations. Users may intervene and refine the searching in both modules if they notice interesting patterns.

For relational data with attributes X, Y, and perhaps others, a contingency table of X and Y is a table that has all of the values occurring under the X attribute along one axis and all of Y along the other. If either attribute has many values, values may be grouped by intervals. Each table element (X_i, Y_j) holds the count of how many times the pair X_i, Y_j appeared in the data.

Module 1 uses domain knowledge to constraint which attributes it looks for contingency tables between. Statistical tests are conducted on contingency tables, including a test for statistical equivalence between two variables. Equivalence relations allow the further pruning of the search space because one of the variables drops out of the analysis.

Module 2 uses the results of module 1 and continues the search for higher-order patterns. It can combine results across dimensions. For example, it can use equivalence relations to form taxonomies by the joining shared values. Module 2 can also improve ranges and find equations and further refine them.

Both modules allow users to control the search. Both modules have built-in mechanisms for deciding search thresholds and parameters. Users may tell 49er.b to search more deeply in an area of the search space than it would naturally if an interesting pattern catches the user's eye.

The output of 49er.b is a set of assertions (e.g. contingency tables, equivalence relations, taxonomies and equations) from which predictions can be made, and, which give explanatory insight into the system under study.

3.6 Limitation of Bacon-like systems

Programs like Bacon, Abacus, Coper, Kepler and others are designed to find functional relationships of scientific significance in numerical data without relying on the deep domain knowledge scientists normally bring to bear. Each program accepts numerical data and attempts to perform domain-independent function finding.

While a great deal of effort has been extended in designing function finding systems, little has been done to evaluate them. Researchers have nearly always relied on anecdotal evidence, reported the successes of their programs on a few hand-selected cases, most of which have consisted of artificial data generated to conform almost exactly to a functional relationship. It is important to know the likelihood of a program's success on a new problem involving real scientific data.

Criticisms of Bacon-like systems come from following directions.

3.6.1 Re-discovery

A view criticizing Bacon for defining its task in a trivial way describes Bacon's task as follows: Given all the right data, and only the right completely noise-free data, find previously known relationships involving products, ratios, or simple arithmetic relations. Not surprisingly, this task can be done; the program clearly does not deal with problems of selecting which variables to examine, noisy data, and so on. A specific point in this case was that artificial data used as input for Bacon always conformed to the relationship the program was desired to find. The data were not merely artificial, but exact.

3.6.2 Noisy Data

While Bacon's authors acknowledge that real scientific data are noisy, they did not make any attempt to reflect this aspect of reality in constructing artificial test problems. This issue provides an important insight. Implicitly, these researchers suggest that they consider noise a *detail* of function finding rather than a central issue. This can be contrasted with that of statisticians, who predicate their very existence as a discipline on the ubiquity of noise and other random effects. The main point here is that demonstrating Bacon's performance on exact artificial data provides no evidence regarding its likely performance on real-world type noisy data.

3.6.3 Search Efficiency and Irrelevant variables

Another area of criticism relates to the presence of irrelevant variables. It is known that chemists made little progress until they decided to turn their attention to the weights and volumes of elements and compounds. One can easily imagine a Bacon-like system methodically considering and rejecting variables in a noise-free environment. However, the dual presence of irrelevant terms and noise makes the task much more difficult, since one can never be entirely sure that an independent variable is irrelevant. Search efficiency is another serious issue. Early Bacon runs took hours for a database of four or five variables and 70 observations. Runtimes would be orders of magnitude greater, were the program to deal with noisy-data, more variables, irrelevant variables, and the additional rules about combining and examining variables.

3.7 Model Selection in Discovery Systems

Several criteria have been proposed by philosophers of science for comparing competing hypotheses. Among them are accuracy (empirical support), simplicity, novelty and cost (utility). Most automated approaches consider accuracy and simplicity [77].

IDS by Nordhausen and Langley was perhaps the first general program for scientific discovery. IDS takes as input an initial hierarchy of abstracted states and a sequential list of "histories" (qualitative states). Using each history, IDS modifies the affected nodes of the abstracted state tree to incorporate any new knowledge gained from that history. Its output is a fuller, richer hierarchy of nodes representing history abstractions.

Thagard introduced Processes of Induction (or PI), as a computational scheme for scientific reasoning and discovery, but not as a working discovery tool. PI represents models as having theories, laws and data. It evaluates scientific models by multiplying a simplicity metric by a data coverage metric. The simplicity metric is a function of how many facts have been explained and of how many co-hypotheses were needed to help explain them. The evaluation scheme is fixed and has no notion of degree of inaccuracy.

Zytkow and Zembowicz developed 49er, a general knowledge discovery tool. It has a two stage process for finding regularities in databases. The first stage creates contingency tables (counts of how often values of one attribute co-occur with those of another) for pairings of database attributes. The second stage uses the contingency tables to constrain the search for other, higher order, regularities (e.g, taxonomies, equations, subset relations, etc.). Valdes-Perez has suggested searching the space of scientific models from the simplest to ones with increasingly more complexity, stopping at the first that fits the data. MECHEM uses this approach to find chemical reaction mechanisms.

Such orderings would be easy to encode as heuristic functions. The MML theory extends these approaches by providing for adjustable heuristic functions that not only result in robust model selection but also allow for the encapsulation of domain knowledge through priors.

3.7.1 A General Basis for Model Preference and Validation

Machine discovery is progressing towards general programs that will assist scientists in creating and improving scientific models. Realizing this goal requires progress in machine learning, knowledge discovery in databases, data visualization and search algorithms. Most importantly, it requires progress in scientific model selection. The scientific model preference problem is compounded by the fact that several scientists with very similar background knowledge may see the same data but may prefer different models.

The minimum message length (MML) criterion is a mathematically well-grounded approach for choosing the most probable theory from given data. Inspired by information theory and Bayesianism, the criterion states that the most probable model has the smallest encoding of both the theory and data. Ideally, the theory's encoding results from an estimation of its prior probability. The encoding of the data is also probabilistic: a function of the given theory.

Despite its generality and power for finding parameters in single classes of models (e.g., the class of polynomials), many have expressed skepticism about whether MML may meaningfully be applied to model selection in heterogeneous model spaces (e.g., general scientific discovery). This leads to the general issue of learning versus prediction. The goal of inductive inference as expressed by the MML principle is the need to provide a plausible explana-

tion for some observational data. A plausible and accurate explanation is a natural predictor.

Our immediate, limited goal is to devise a heuristic function that can help users in large and heterogeneous model spaces. Ideally, a search algorithm that is informed with our heuristic will return several regions in the model space that contain promising models, some known and some novel. Our approach is to use MML in an effective manner. The following chapters describe systems that employ the MML principle in model discovery. The results from those chapters offer a strong justification for the use of the MML method in model search and estimation.

3.7.2 Justifying the MML Heuristic

We do not claim to have an optimal heuristic function in terms of returning the truly "best" model. Rather, our goal is to use MML as the preferred heuristic that can be incorporated into the discovery framework. Good heuristics for real-world problems are often tricky to design. The MML approach can be justified through the following criteria.

1. Generality of model space: We seek a function that is applicable to both primarily conceptual models as well as primarily numeric. The MML metric can be applied for model preference over structurely different model spaces.

2. Robust to noise: Discovery systems need to be robust in the present of noisy data. Unlike maximum likelihood models, the MML function takes into account the noise factor.

3. Simplicity of form: There are several competing beliefs for how scientific

models should be evaluated. The function's design should be as transparent as possible so that its assumptions are readily comprehended.

4. Consistency: The function should provide for an increase in the accuracy of models selection as more data is accumulated. The MML approach has been proven to be statistically consistent.

Some minor issues that arise in efforts to apply MML to general scientific discovery need mention. Among them are the specification of the initial theory prior probabilities, the inherently iterative nature of MML, and the difficulty in searching this space for a true "highest probability" theory. Like other MML efforts, there is no good rule for specifying an initial set of prior probabilities. Although Cheeseman and others warn about using syntactic features, this may be the easiest approach to try in a new domain. MML is an inherently iterative process of redefining theory spaces and prior probabilities.

3.8 Summary

As Joseph Phillips explains in [77], we are at the dawn of a new scientist-computer partnership. Previously, computers have been used by scientists to calculate parameters of particular models. Examples include the multitude of numerical methods that find the vector that best fits given constraints, e.g. least-square fitting and Fourier transforms. This relationship began to change, however, with the introduction of systems like Nordhausen and Langley's IDS. IDS was among the first programs to build heterogeneous models (models containing class hierarchies, and qualitative and quantitative relations) from scientific data.

At least two ramifications follow from the very explicit nature of writing these programs. One is that developing them will tell us where the study

of scientific discovery needs more work. The second is that developing them will give us pragmatic suggestions for the development of the philosophy of science. For example, precise definitions to terms like "model" and "theory" currently have no agreed meaning. A third implication is that because these systems codify knowledge and technique, they can serve as controlled testbeds to compare the effectiveness of proposed models of scientific discovery.

Chapter 4

Minimal Length Encoding for Model Selection

Since the late 1960's minimum length encoding techniques for model estimation and preference have gained popularity and accuracy in their applications to inductive inference. The following "minimum-length" description by Chris Wallace [117] describes the essence of inductive learning and the place of minimal length encoding methods.

> There is ample evidence that a well-accepted scientific theory has great explanatory power. Much of what is observed is deducible from the theory, so the observed data may be reconstructed from a statement of the theory and a detailing of the undeducible residuum. Typically, the statement of theory and residuum is far shorter than a plain statement of the data.
>
> Most accounts of scientific method emphasise predictions and the experimental checking of these. But predictive ability (although often justifying a society's investment in science) is not the touchstone of scientific enquiry. Quite accurate prediction of

eclipses was achieved by several societies with little understanding of their cause, and as is shown by Solomonoff, the best possible prediction explicitly avoids commitment to any theory. In fact, however, human science seeks understanding, usually couched as a theory, and prediction is an economically-valuable by-product more often than the primary goal.

If we turn this descriptive account of science as finding concise, theory-based explanations of observed data into a prescription for how to handle data, we get various flavours of inductive inference techniques based on measures of information: Solomonoff prediction, MML and MDL theory selection, MML and BIC estimation. The differences reflect independent and near-simultaneous development, and are in practice far less important than the similarities. They use Shannon and Kolmogorov measures of "information". Formally, if not always philosophically, they are closer to Bayesian statistical inference than to classical methods. All rely on a fundamental trade-off between complexity of theory and fit to data encapsulated in the length of a message stating both theory and data.

4.1 Introduction

Between 1960 and 1964 Ray Solomonoff [99], a mathematical physicist at Cambridge in Massachusetts,U.S.A., was interested in the inductive problem of finding an algorithm for discovering the "best" grammar for a given set of acceptable sentences. One of his primary concerns was : Given a set of positive cases of acceptable sentences and several grammars, any of which is able to generate all of the sentences – what goodness of fit criterion should be

used? Solomonoff realized that the existence of the Universal Turing Machine implies the existence of universal methods of inductive inference, meaning methods which given enough data will find the true model. The universal methods are not Turing computable, but one can look for computable approximations. Solomonoff independently realised and formally expressed in his paper [99] that the models that provided the shortest explanation for a given finite string could be construed as more probable. Thus arrived the concept of *algorithmic probability*.

Andrei Nikolaevich Kolmogorov, born 25 April 1903 in Tambov, Russia, was perhaps the foremost recent Soviet mathematician and counts as one of the great mathematicians of the 20th century. His many creative and fundamental contributions to a vast variety of mathematical fields are so wide-ranging and profound that it is impossible to describe them even briefly. Following a four decades long controversy on von Mises's definition of an infinite random sequence, in a 1965 paper [53] Kolmogorov used the theory of algorithms to describe the complexity of a finite object as the length of its smallest description (program to reconstruct it). This makes the definition of complexity depend on the programming language used. However, it turns out that there are universal methods for which the complexities of the objects described are asymptotically optimal. Algorithmic information theory, or "Kolmogorov complexity theory", originated with the discovery of universal descriptions of finite objects, and a recursively invariant approach to the concepts of complexity of description, randomness and a priori probability.

In this chapter we reflect on the historical foundations of the Minimum Message Length principle for inductive inference. The roots of MML-based

inference in the definitions of algorithmic probability and Kolmogorov complexity are briefly investigated. The chapter serves as a prelude to the applications of the MML methodology described in the following chapters.

4.2 Foundations and Background

Historically, Kolmogorov's motivation for the development of the complexity measure was firmly rooted in von Mises' notion of random infinite sequences, proposed in 1919 as the foundation for the frequency interpretation of probability. Von Mises's formulation of the notion of randomness was based on the intuition of 'the impossibility of a gambling system'. In his own words, randomness in a sequence lies in the "impossibility of devising a method of selecting the elements so as to produce a fundamental change in the relative frequencies". Thus, according to the axiom of randomness the limiting value of the relative frequency must be the same for all possible infinite subsequences of trials chosen solely by a rule of place selection within the sequence (i.e., the outcomes must be randomly distributed among the trials). Here an admissible place selection is a procedure for selecting a subsequence of a given sequence ξ in such a way that the decision to select a bit $\xi[n]$ does not depend on the value of $\xi[n]$ or any later value [131]. Von Mises's, however, left open the question of what selection functions should be admitted, with the consequence that nothing could be counted as being random due to non-computable place selections. In 1939 Wald proved that random sequences exist relative to any countable class of selection functions. Based on Wald's work, in 1940 Church [19] suggested that randomness in the intuitive sense should be viewed as algorithmic randomness and he proposed to call a sequence random if it is random relative to the class of computable selection functions. Kolmogorov felt that the frequency concept, based on the notion

of limiting frequency as the number of trials increases to infinity, did not contribute anything to the application of the results of probability theory to real practical problems where we always have to deal with a finite number of trials. With the advent of electronic computers in the 1950's, a new emphasis on computer algorithms, and a maturing general recursive function theory, ideas tantamount to Kolmogorov complexity came to many people's minds. Thus, R. Solomonoff in Cambridge, Massachusetts, had formulated similar ideas in 1960 and published his truly innovative work [99] on the subject in 1964 in *Information and Control.*

4.2.1 Randomness

The five years following the 1964 publications of Solomonoff was a time of much activity in this field. Kolmogorov and a number of other researchers were interested in randomness and complexity of one string with respect to another, as well as the development of information theory and probability based on lengths of codes. However, they appeared to be more interested in various aspects of randomness as opposed to inductive inference.

The algorithmic complexity of a string was defined to be the length of the shortest code needed to describe it. A random string was one whose complexity was about as large as its length. A completely different approach to the definition of random sequences was proposed by Martin-Löf [63] in 1966. He developed a quantitative (measure-theoretic) approach to the notion of random sequences. This approach was free from the difficulties associated with the frequency approach of von Mises. Around the same period and again independently of Solomonoff and Kolmogorov, Chaitin published papers [14, 15] defining randomness in terms of program-length. He informally suggested that the shortness of a program that describes a sequence might

be an index as to how good a theory that program represents. Later it was shown by Levin, Gács and Chaitin that one can refine the notion of algorithmic complexity by defining it relative to a set of admissible descriptions. If admissible descriptions are restricted such that no description is a proper prefix of any other description, then an infinite sequence is Martin-Löf random if and only if each of its finite initial sequences has a complexity that equals (up to a fixed constant) its length.

4.2.2 Solomonoff and Algorithmic Probability

Borrowing the definition from Grünwald [46], the Kolmogorov complexity of a sequence can be defined as the length of the shortest program that prints the sequence and then halts. The lower the Kolmogorov complexity of a sequence, the more regular or, equivalently, the less random or, yet again equivalently, the simpler it is. Measuring regularity in this way confronts us with a problem, since it depends on the particular programming language (UTM) used. However, in 1964, Solomonoff proved the invariance theorem, which roughly states that it does not matter much exactly what programming language one uses, as long as it is universal. More precisely, for any two general-purpose programming languages A and B, there exists a constant c such that, for every data sequence D, the length of the shortest program for D written in language A and the length of the shortest program for D written in language B differ by no more than c. Here c may depend on the languages A or B, but it is constant in that it does not depend on the size of D: if the data set D is large enough, then the difference in length of the shortest programs according to A and B is very small as compared to the length of the data set.

Consider the following definition of algorithmic probability from Solomonoff

[101]:

$$P(x) = \sum 2^{-l_i} \tag{4.1}$$

$P(x)$ is the Algorithmic Probability of finite string x where l_i is the length of the i-th description of string x summed over all such descriptions. That the sum is noncomputable, is associated with the fact that it is often impossible to verify in finite time, whether a particular string is a description of x or not. Willis [134] tried to deal with this "halting problem" by defining computationally limited Turing machines. The machines were chosen such that they had no halting problems. From these machines, he was able to define probabilities in an exact manner. By raising the computational limits on his machines, they approached the behaviour of universal machines.

Solomonoff's theorem on the convergence of algorithmic probability [100] makes Algorithmic Probability look very attractive as a means of induction. In his own words,

> It is the only induction system we know of that is "complete". By this we mean that if there is any describable regularity in a body of data, Algorithmic Probability is guaranteed to discover it using a relatively small sample of the data. It is the only probability evaluation method known to be complete. As a necessary consequence of its completeness, this kind of probability must be incomputable. Conversely, any computable probability measure cannot be complete.
>
> The term "incomputable" is used here in a rather special way. Algorithmic Probability is as "computable" as the value of PI – but with one important difference; when we make successive approximations to the value of PI, we know how large the error in each approximation can be. In the case of Algorithmic Proba-

bility, we have a procedure for successive approximations that is guaranteed to converge to the right value. However, at no point in the calculation can we make a useful estimate of the error in the current approximation. This might be regarded as a serious criticism of the use of Algorithmic Probability or approximations to it, to solve practical problems, but it is not. It is a difficulty shared by all probability evaluation methods. If they are complete, then they are incomputable. If they are computable, (either as independent probability measures or as approximations to Algorithmic Probability) then they must be incomplete. This incompleteness implies that there have to be regularities that are invisible to them. When used with data having regularities of these kinds, computable methods will have errors of unknown size.

In spite of its incomputability, Algorithmic Probability can serve as a kind of "Gold Standard" for induction systems – that while it is never possible to tell how close a particular computable measure is to this standard, it is often possible to know how much closer one computable measure is to the standard than another computable measure is. Over the years there has been a general impression in the scientific community that this incomputability would make it impossible to use Algorithmic Probability as a tool for statistical prediction. From the beginning, however, this difficulty was recognised and methods for dealing with it were proposed.

4.2.3 Computable Approximations

As mentioned earlier the Kolmogorov complexity or algorithmic probability as such cannot be computed – there can be no computer program that, for every set of data D, when given D as input, returns the length of the shortest

program that prints D: assuming such a program exists leads to a contradiction. Another problem is that in many realistic settings, we are confronted with very small data sets for which the invariance theorem does not say much.

One approach to overcome this problem of incomputability, termed "Resource Bounded Algorithmic Probability" by Solomonoff, was formalised by Willis in 1970 [134]. The most efficient way to implement Resource Bounded Algorithmic Probability is to approximate equation (4.1) by the largest lower bound on $P(x)$ that can be demonstrated in time, T. This is usually done by finding many short codes for x that give terms summing to that bound. This kind of approximation to Algorithmic Probability is an example of a "time limited optimisation problem". By getting as large a value of the sum as we can, we get as close as possible to Algorithmic Probability in the allotted time.

There are theoretical reasons for believing that the error may be small if the shortest code one has found thus far were indeed the shortest code for the data. However, for induction problems in which one is uncertain as to the class of stochastic functions that generated the data (as in all empirical sciences), one cannot know if one has the shortest code and the amount of accuracy lost must remain unknown. There have been several effective approximations to Algorithmic Probability. One of the earliest is that of Van Heerden [110], who considered the prediction of binary sequences by Boolean functions. His criterion for the best function to use: Add the length of the description of each function to the number of errors it made in prediction. The function minimising total length is best. This technique according to Solomonoff can be regarded as approximating the sum in equation (4.1) by the single largest term we can find.

In 1968 Wallace and Boulton [122] proposed the Minimum Message Length

(MML) principle that suggested a Bayesian information-theoretic approximation to algorithmic probability. The idea behind the independently pioneered MML principle was to scale down Solomonoff's approach so that it does become applicable: instead of using a code based on a universal computer language, they suggested using description methods M which still allow us to compress many of the intuitively regular sequences but which are nevertheless such that for any data sequence d, we can always compute the length of the shortest description of d that is attainable using a method from M. The price we pay is that, using the practical MML principle, there will always be some regular sequences which we will not be able to compress. But this is not an objection, since it is a known that there can be no method for inductive inference which will always give us all the regularity there is – simply because there can be no automated method which for any sequence d finds the shortest computer program that prints d and then halts. Moreover, it will often be possible to guide a suitable choice of M by a priori knowledge we have about our problem domain. For example, it is possible to pick a description method M_p that is based on the class of all polynomials P, such that with the help of M_p we can compress all data sets which can meaningfully be seen as points on some polynomial. Wallace and Boulton [122, 123, 127] considered various functional forms to assign probabilities to a sequence of data symbols. The function selected was the one for which the length of description of the function minus the logarithm of the probability assigned to the data by the function, was minimal.

In the following sections we will aim to describe the theory behind the development of the Minimum Message Length Principle. The MML approximation as developed for the problem of inductive inference will be derived and discussed in greater detail.

4.3 Model Preference in Inductive Learning

When learning a function $h : X \rightarrow Y$ from random training examples $\langle x_1, y_1 \rangle, \ldots, \langle x_t, y_t \rangle$, there is a well-known tradeoff between the size of the data description and the complexity of the function class being considered: If the class is too complex for the sample size, there is a risk of "overfitting" the training data and guessing a function that performs poorly on future test examples. On the other hand, an overly restricted class can prevent us from considering any good candidate functions. The most common strategy for coping with this dilemma in practice is to use some form of automatic model selection, such as complexity-penalization or repeated holdout testing, to balance complexity and goodness of fit. Under the simplest formulation of model selection, the idea is to first stratify the hypothesis class H into a sequence of nested subclasses $H_0 \subset H_1 \subset \ldots$ ordered by complexity and then (somehow) choose a class which has the appropriate complexity for the given training data. To understand how we might make this choice, note that for a given training sample $S = \langle x_1, y_1 \rangle, \ldots, \langle x_t, y_t \rangle$ we obtain a corresponding sequence of empirically optimal functions $h_0^*, h_1^*, \ldots$, one from each subclass, that achieve minimum average error ϵ on the training set S. The essence of the model selection problem is to choose one of these functions based on their observed empirical errors $err(h_0^*, S), err(h_1^*, S), \ldots$ However, these errors are monotonically decreasing, and therefore choosing the function with minimum training error simply leads to choosing a function from the largest complexity class. Therefore, the trick is to invoke some other criterion beyond empirical error minimisation to make this choice. Currently, three basic model selection strategies predominate [25].

The most common strategy is complexity-penalization. Here one assigns increasing complexity values $c_0, c_1, \ldots$, to the successive function classes, and

then chooses the hypothesis from the class h_i^*, that minimises some combination of complexity and empirical error (e.g., the additive combination $c_i + err(h_0^*, S)$). There are many variants of this basic approach, including the Akaike Information Criteria [2], "Bayesian" Information Criteria [96], Structural Risk Minimization [112], and "generalized" cross validation [23]. These strategies differ in the specific complexity values they assign and the particular tradeoff function they optimise, but the basic idea is still the same.

The other most common strategy is hold-out testing. Here one asks: for the given set of training data, which hypothesis class H_i generalises best? We answer this by partitioning the training set, $1, \ldots, t$, into a pseudo-training set, $1, \ldots, k$, and a hold-out test set, $k+1, \ldots, t$, and then using the pseudo-training set to obtain a sequence of hypotheses $\hat{h}_0, \hat{h}_1, \ldots$, etc. We then use the hold-out test set to obtain an unbiased estimate of the true errors of these hypotheses. (Note that the training set errors tend to be gross underestimates in general.) From these unbiased estimates, we can simply choose the hypothesis class H_i that yields the hypothesis $\hat{h}_i$ with the smallest estimated error. Once H_i has been selected, we return the function $h_0^* \in H_i$ that obtains minimum empirical error on the entire training sequence. Again, there are many variants to this basic strategy—having to do with repeating the pseudo-train pseudo-test split many times and averaging the results to choose the final hypothesis class; e.g., 10-fold cross validation, leave-one-out testing, bootstrapping, etc.

Finally, the third class of model selection techniques have been inspired by the notion of Kolmogorov complexity and Algorithmic Probability resulting in the computable approximations well-known as the Minimum Message Length [125] and Minimum Description Length [87] principles. While the value of Algorithmic Probability is clearly defined in a mathematical sense,

it is incomputable. Approximations usable from an inductive learning point-of-view depend critically on three major factors:

1. The data.
2. The a priori information — which is uniquely characterised by our choice of universal reference machine.
3. The resources available for computation – time and memory.

Any failure to specify each of these arguments exactly can lead to gross ambiguities in the value of the probability. Thus, the defining feature in this third class of model inference/selection procedures is the interpretation of algorithmic probability as a part of a two-stage coding process. The code length of the model's specification is an approximation of its algorithmic probability. The essence of this approach is that all models are encoded using an efficient coding scheme along with the model's unexplainable residuum on the training data. Models that minimise such an encoding scheme are preferred.

4.4 The MML Approximation

The Minimum Message Length principle pioneered in 1968 by Wallace and Boulton is based on the correspondence between *compression* and *regularity*. The idea being that the more one is able to compress a set of data, the more regularity one can detect in the data. The relationship between this and the concepts of Kolmogorov complexity and algorithmic probability are obvious. Consider the following example from Grünwald [46] that illustrates the essence of this method. Suppose, for the moment, that our data are a finite sequence of bits (zeroes and ones), and consider the following three

sequences. We assume that each sequence is 10,000 bits long, and we just list the beginning and the end of each sequence.

1. 0001000100010001000100010001.....0001000100010001000100010001

2. 0111010011010000101010101.....1010111010111011000101100010

3. 1110011111101001101111111.....0111101111101111100111101111

The first of these three sequences is a 2500-fold repetition of 0001. Intuitively, the sequence looks regular; there seems to be a simple law underlying it and it might make sense to conjecture that future data will also be subject to this law. The second sequence has been generated by tosses of a fair coin; this means that there is definitely no law or regularity underlying it. The third sequence contains exactly four times as many 1s as it contains 0s. In contrast to sequence (2), there is more discernible regularity in this data, but of a statistical rather than of a deterministic kind. Again, it seems sensible to note that such a regularity is there and to predict that future data will behave according to it.

The fundamental insight which leads to the MML principle is that any regularity in the data can be used to compress the data, i.e., to describe it in a shorter manner, assuming the regularity is there. Such a description should always completely determine the data it describes. Hence, given a description or encoding $\hat{d}$ of a particular sequence of data D, we should always be able to fully reconstruct D on the basis of $\hat{d}$. A particularly versatile description method is a general-purpose computer language like C or Pascal. A description of D is then any computer program that prints D and then halts. Let us see whether our insight works for the three sequences above. Using a language similar to Pascal, we can write a program

```
for i=1 to 2500; print '0001'; next; halt
```

which prints sequence (1) but is clearly a lot shorter than it. The shortest program (program length measured by number of characters) printing sequence (1) is at least as short as the program above, which means that sequence (1) is indeed highly compressible. On the other hand, we will show in the next section that, if one generates a sequence like (2) by tosses of a fair coin, then, with extremely high probability, the shortest program that prints (2) and then halts will look something like this:

```
print
'01110100110100001010101O.....1010111010111011000101100010';
halt
```

This program has size equal to the length of the sequence plus a constant. Clearly, it is does nothing more than repeat the sequence. It is easy to show that there exists quite a short program generating sequence (3) too.

4.4.1 Notes from Coding Theory

The MML methodology provides a general expression for computing the message length of models and data. However, before we go on to derive this expression, it is important to note the correspondence between codes and probability distributions. The theory of optimal codes is well developed and based on Shannon's theory of information.

For our purposes, we need to know that:

- A string optimally encoding a datum of probability p has length $-\log_2 p$, neglecting the fractional parts;
- A string optimally encoding a set of data drawn from the probability distribution assumed in constructing the optimal code has the statistical properties of a random binary stream;

- Well-known techniques exist for the construction of an optimal code for any given probability distribution. These techniques produce code strings at most one digit longer than the theoretical optimum of $-\log_2 p$. Common techniques are Huffman codes and arithmetic codes. It is very important to note that we are interested in these techniques only for obtaining code-lengths for our models and data, as opposed to building actual messages.

4.4.2 The Minimum Message Length Criterion

Let us now consider the concept of an "explanation" (borrowed from Wallace) which yields a better understanding of the two-stage MML encoding. Also assume that the available data is encoded as a binary string. An explanation of the data is another "two-part" binary string which encodes the data but embodies a certain structure. The first part E_1 encodes a model or theory θ. The second part E_2 encodes the data using a code which would be optimal were θ true. The optimality is in the sense of minimising the expected length of E_2.

The MML principle suggests that one prefer the hypothesis that gives the shortest "explanation" of the data. We now consider the two parts of the coding scheme in greater detail. It is clear from above that the length of the second part E_2 of an explanation is

$$-\log[\textit{Probability of data assuming } \theta \textit{ is true}] = -\log P(D|\theta) \qquad (4.2)$$

where θ is the "model" asserted in the first part E_1. An important observation here is the relationship between the length of E_2 and θ. If θ fits the data well then the length of E_2 is small, otherwise it increases as the goodness-of-fit diminishes.

The encoding of part E_1 of the explanation requires more attention. The MML methodology emphasizes the importance of prior knowledge and its use in devising optimal encoding schemes. Let us then define $Prior(\theta)$ to be the probability that we would assign to a model θ before the data is seen. The length of E_1 using an optimal code is given by

$$length(E_1) = -\log Prior(\theta) \tag{4.3}$$

The total length of our "explanation" E is then

$$\begin{aligned} length(E) &= -\log Prior(\theta) - \log P(D|\theta) \\ &= -\log Probability(\theta \text{ and } D) \end{aligned} \tag{4.4}$$

The importance of prior probabilities and the difficulties in assigning them to competing models in practice is well-known. This problem is further exaggerated if the models under consideration include continuous parameters. For example, consider a polynomial family of models where each parameter is a coefficient, in such a case it is hardly obvious that a particular model with a set of coefficients can be assigned any prior probability at all.

However, as Wallace points out, instead of asking how to assign prior probabilities to models, we look for different ways of describing a model. Specifically, what kind of language would be reasonable for describing a model? Consider for example (taken from Wallace [117]) that we are trying to learn a Boolean function. We might reasonably decide that the model or hypothesised function, could be represented in a code more or less as we would represent it on paper: as a string of variable labels and Boolean operators:

```
A.B or [(not C or D). not A]
```

It is quite easy to see how, knowing the number of variables on which the function might depend, a fairly efficient binary encoding of such formulas

can be designed. We could take such a code as defining a prior probability distribution over functions. In effect, the code assigns a low probability to functions represented by long, complex formulae, and high probability to functions having simple formulae. That is, we can substitute a complexity order implied by a language (or Universal Turing Machine) for our missing prior.

When the hypothesised model has one or more real-valued parameters, prior expectations will not usually assign finite prior probability to any specific value. Rather, the available domain knowledge will define a prior probability density over the continuum of possible values. Often the lack of prior knowledge together with arguments about symmetry lead to the adoption of some form of *non-informative* prior density. Having defined a prior density, the coding of a parameter value in E_1 of the explanation involves a compromise. Typically, the length of E_2 will be minimised by some single value of the parameter, so we would like E_1 to state this value. If the value is stated to high precision resulting in many binary places, the length of E_2 is minimised, but the coded value in E_1 will be long. If we state it imprecisely, rounding off to a few binary places, A becomes short but in general E_2 will be longer than its minimum possible length.

We shall illustrate the treatment of real valued parameters by considering a simple model with one parameter θ which determines the probability for the observation x from D. The probabilities $p(x|\theta)$ and $p(\theta)$ are assumed to be specified. Then we ask how, using the given model, we can encode x into a message using the least amount of bits. Assume that we require x to be encoded with the precision δ_x. The purpose is not actually to encode x or send a message to someone, but to learn about θ by developing a coding scheme. Wallace and Boulton [122, 123] suggested the following two-part message:

first encode θ discretised with precision δ_θ using the prior probability $Prior(\theta)$ and then encode x using the likelihood $p(x|\theta)$ and the encoded θ. This will produce a code with $length(E)$ as defined earlier. If δ_θ is small, the number of bits used for the first part of the message is approximately

$$length(E_1) = L(\theta) \approx -\log[p(\theta)\delta_\theta] \tag{4.5}$$

The number of bits used for the second part of the message depends on the discretised value of θ. Assuming that the true value of θ was $\hat{\theta}$, the discretised value lies between $\hat{\theta} - \delta_\theta/2$ and $\hat{\theta} + \delta_\theta/2$ with roughly uniform probability. This means that the expected number of bits used for the second part is approximately

$$length(E_2) = L(x|\theta) = \int_{\hat{\theta}-\delta_\theta/2}^{\hat{\theta}+\delta_\theta/2} -\log[p(x|\theta)\epsilon_x] \tag{4.6}$$

If δ_θ is small, it is possible to approximate the length of the second part of the message by using a second-order Taylor series expansion of $-\log[p(x|\theta)$ about $\hat{\theta}$.

Given x, the total message length is a function of $\hat{\theta}$ and δ_θ. The result of learning is the optimal value for both, which minimises the message length. Looking at the equations for $L(\theta)$ and $L(x|\theta)$, it is evident that there is an optimal value for δ_θ, because increasing the size of the discretisation bins will decrease $L(\theta)$ due to the term $-\log\delta_\theta$, but it will increase the term $L(x|\theta)$ because the discretisation errors will increase and the expected deviations from the optimal $\hat{\theta}$ grow larger. The optimal value for δ_θ depends on how quickly $p(x|\theta)$ drops as θ is moved further away from the optimal value, and it turns out that the optimal δ_θ is linearly dependent on the width of the maximum peak of $p(x|\theta)$. Therefore the optimal $\hat{\theta}$ will give us the most plausible value for the parameter, and δ_θ tells us how uncertain the value is.

According to Wallace [127] the best compromise states a parameter value to a precision δ_θ or maximum roundoff error $\delta_\theta/2$, where

$$\delta_\theta = \frac{1}{\sqrt{\kappa_k^k F}} \tag{4.7}$$

where F is the Fisher information, or the expected value of the second derivative of $-\log Probability(D|\theta)$ with respect to the parameter, and κ_k depending on the dimensionality k of the parameter (i.e., on the number of free scalar parameters). This choice of precision as discussed earlier is essentially the precision to which we expect to be able to estimate the parameter.

In general terms, given data D, a family of possible theories parameterised by the k-dimensional parameter θ, and a prior density $p(\theta)$, the length of an explanation asserting model $\hat{\theta}$ is

$$TotalLength(E) = -\log \frac{p(\hat{\theta})}{\sqrt{\kappa_k^k |F(\hat{\theta})|}} - \log P(D|\hat{\theta}) + k/2 \tag{4.8}$$

where κ_k is a geometric constant, the last term arises from round-off error in stating $\hat{\theta}$, and logs are to base e. The message length is in units of $\log_2(e)$ binary digits.

4.5 The Bayesian Connection

Let us now note some important relationships between the MML method and Bayesian inference. Information theory can offer a simple, intuitive point of view to inductive learning. Message length and probability are tightly linked. According to Shannon's coding theory, the shortest expected description length for a proposition equals the negative logarithm of the probability of the proposition. In this context, the information-theoretic approach to learning is akin to using a different scale for measuring the beliefs, and any

learning method derived in the information-theoretic context can be readily translated into the Bayesian context by a simple transformation of scale. Concepts from the Bayesian framework often have intuitive interpretations in the coding context. The prior probabilities, for instance, translate into the specification of a coding scheme. Optimal encoding of an observation into a binary string produces a seemingly random string of ones and zeros and some prior knowledge is needed about the instructions for decoding. This corresponds to the Bayesian prior. Another example is that slight approximations to the exact Bayesian learning translate into slightly non-optimal coding schemes.

Under one popular interpretation the goal of Bayesian learning is to find the most probable hypothesis given the data at hand. This is also well-known as the Maximum A Posteriori (MAP) method. Let h_{map} be our MAP hypothesis. Consider the following standard definitions:

- $P(h)$ = prior probability of hypothesis h
- $P(D)$ = prior probability of training data D
- $P(h|D)$ = probability of h given D (posterior density)
- $P(D|h)$ = probability of D given h (likelihood of h given D)

From the above definitions we can derive h_{map} as follows:

$$\begin{aligned} h_{map} &= arg\max_{h \in H} P(h|D) \\ &= arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)} \\ &= arg\max_{h \in H} P(D|h)P(h) \end{aligned}$$

This definition of the MAP hypothesis can be interpreted in terms of the MML principle. Information theory tells us that the optimal (shortest expected

coding length) code for an object (event) with probability p is $-\log_2 p$ bits. According to the MML principle the best hypothesis is one that minimizes the joint encoding (*MessLength*) of theory and data. That is:

$$\begin{aligned} \text{Best}h &= \min_{h \in H} -\log_2 P(D|h) - \log_2 P(h) \\ &= \max_{h \in H} \log_2 P(D|h) + \log_2 P(h) \\ &= \max_{h \in H} P(D|h)P(h) \\ &= h_{map} \end{aligned}$$

Specifically, let us now consider the expression for message length described in equation 4.8. Choosing the theory $\hat{\theta}$ to minimise *TotalLength* is equivalent to maximising

$$\frac{p(\hat{\theta})}{\sqrt{F(\hat{\theta})}}.P(D|\hat{\theta})$$

Now $p(\hat{\theta})\delta_\theta$ is approximately the total prior probability mass assigned to all theories within $\pm\delta_\theta/2$ of the stated model $\hat{\theta}$. Since the data does not allow us to distinguish among these models, we may regard $p(\hat{\theta})\delta_\theta$ as being the finite prior probability of $\hat{\theta}$, and indeed the length of E_1, stating $\hat{\theta}$, is just $-\log(p(\hat{\theta})\delta_\theta)$.

Thus, minimising *TotalLength* is equivalent to choosing $\hat{\theta}$ to maximise

$$P(\hat{\theta}).P(D|\hat{\theta}) = P(D).P(\hat{\theta}|D) \tag{4.9}$$

In other words this is equivalent to choosing $\hat{\theta}$ to maximise the Bayesian posterior probability $P(\hat{\theta}|D)$. The minimum message length criterion is quite a close parallel to Bayesian statistical inference. Conventional Bayesian analysis as such has no standard method for prior selection. Different versions of Bayesianism have applied different priors, like maximum entropy, empirical Bayes or others. The MML methodology advocates using coding theory to assign prior probabilities to models.

4.6 A Better Ockham's Razor

Ockham's Razor is the principle proposed by William of Ockham in the fourteenth century that "Pluralitas non est ponenda sine neccesitate", which translates as "entities should not be multiplied unnecessarily". In more modern terms, if you have two theories which explain the observed facts equally well then you should use the simplest until more evidence comes along.

As physicist Anthony Garrett[1] explains, although William of Ockham often wrote of the idea, he was certainly not its inventor. Later writers attached his name to it, but the idea was common among scholastic theologians, and goes back much further. Here is a quote from Ptolemy, writing in 2nd century Alexandria about changes in the earths solstices and equinoxes, called Procession. "It is a good principle to explain the phenomena by the simplest hypotheses possible, insofar as there is nothing in the observations to provide a significant objection to such a procedure".

Ockham's Razor is often disputed by showing that a more complicated theory was true and the simpler hypothesis was falsified. This is a strawman argument. The Razor doesn't tell us anything about the truth or otherwise of a hypothesis, but rather it tells us which one to test first. The simpler the hypothesis, the easier it is to shoot down. The physicist Richard illustrating the phenomenon of "overfitting"once said that, given enough parameters he could fit an elephant to the curve. So intuitively, there is going to be a trade-off between how well you can fit the data and how complicated the theory is. This trade-off between goodness of fit to the observations, and how simple the theory you're using is, is precisely what the razor suggests. So Ockham's Razor is not just "choose the simplest theory that fits the facts", but "choose the simplest theory that fits the facts well", and there is a measurable trade-

[1]refer to http://www.abc.net.au/rn/science/ockham/stories/s118778.htm

off, between goodness of fit and simplicity of the theory; a trade-off between flexibility and economy.

In the Minimum Message Length principle we see one possible realization of this razor. The MML principle of Wallace is thus an attempt at perfecting the balancing act between the complexity of a model and its fit to the available observations. In other words, the MML principle sharpens the razor of the William of Ochkam.

4.7 Summary

The foundations of the MML principle for inductive inference independent from the work of Solomonoff and developed using coding theory, were described in this chapter. In the following chapters we go on to demonstrate the practical applicability of the MML methodology. In chapter 5, the MML methodology is used to derive a metric for model selection in the domain of univariate polynomial regression, and in chapter 6, the MML approach is applied to the domain of segmentation.

Chapter 5

Polynomial Regression

This chapter describes the design and development of a system for learning univariate polynomials based on the MML methodology. The system is empirically evaluated by including several well-known criteria for model selection. The system was originally conceived as a module which could be encapsulated within a general discovery framework.

5.1 Introduction

In this study we focus on learning among competing models from a family of polynomial regressors. A diverse number of well-known techniques are available for the selection of models in polynomial regression, namely, Finite Prediction Error (FPE) [2], Akaike's Information Criterion (AIC) [3], Schwartz's criterion (SCH) [96] and Generalized Cross Validation (GCV) [23]. Wallace's Minimum Message Length (MML) principle [122, 127, 125] and also Vapnik's Structural Risk Minimization (SRM) [112, 113] – based on the theory of VC-dimensionality – are plausible additions to this family of model-selection principles. SRM and MML are generic in the sense that they can be applied

to any family of models, and similar in their attempt to define a trade-off between the complexity of a given model and its *goodness of fit* to the data under observation – although they do use different trade-offs, with MMLs being Bayesian and SRMs being non-Bayesian being only one of the differences. Recent empirical evaluations [120, 115, 39] comparing the performance of several methods for polynomial degree selection provide strong evidence in support of the MML and SRM methods over the other techniques.

In this study we consider a simple domain where the x data are randomly selected from the interval $[-1, 1]$ and the y values are derived from a univariate function, $y = t(x)$, corrupted with Gaussian noise having a zero mean. Least-squares approximations of polynomials of degree up to 20 are derived from the data generated. These univariate polynomials of varying orders generated by our system is then offered to the five model selection methods and the performance of the preferred polynomials are evaluated by their predictive accuracy on test data, similarly generated. This work presents a system for univariate polynomial function learning and includes an extensive empirical evaluation of five polynomial selection methods – FPE [2], SCH [96], GCV [23], MML and SRM. (In unreported results, we have found that AIC [3] performs almost identically to FPE.) The chapter provides an analysis of the behaviour of these five methods with respect to variations in the number of training examples and the level of noise in the data.

5.2 Background

The target problem is the selection of a univariate polynomial regression function. In this section we aim to summarize the two major approaches based on

the minimum message length (MML) and structural risk minimization (SRM) principles. Let us assume that we have a finite number N of observations of a function $t(x)$ corrupted with additive noise ϵ,

$$y_i = t(x_i) + \epsilon_i \qquad \text{for i} = 1,...,N.$$

Our approximation of the target function is based on the training set of N observations, where the values, x_i, of the independent variable x are independently and uniformly distributed in the interval $[-1, 1]$ and the noise values $\epsilon_i = y_i - t(x_i)$, are independently and identically distributed by a Normal density with zero mean and unknown variance. The framework of the problem follows Cherkassky et al. [18], and has been repeated using MML in Wallace [120] and Viswanathan and Wallace [115]. The values, x_i, of the independent variable x are randomly selected from the uniform distribution on the interval $[-1, 1]$.

The task then is to find some polynomial function, $f(x) = \hat{t}(x)$, of degree d that may be used to predict the value of $t(x)$ in the interval, $-1 \leq x \leq 1$. In our evaluation we only consider polynomials $f(\cdot)$ of degrees up to 20 and for any given degree d, we select the polynomial $f(d, x)$ that minimizes the squared error SE on the training data,

$$SE(f(d,x)) = \sum_{i=1}^{N} (y_i - f(d, x_i))^2 \tag{5.1}$$

The performance or *prediction risk* is the expected performance of the chosen polynomial for new (future) samples. This is measured by its Squared Prediction Error SPE, which is estimated using a simple Monte Carlo method:

$$SPE\,(f(d,x)) = \frac{1}{m} \sum_{i=1}^{m} (f(d, x_i) - t(x_i))^2 \tag{5.2}$$

where t(x) is the target function and m is $\max(N, 50)$ and the test data (i = 1 to m) are randomly selected from a uniform distribution in [-1,1].

5.2.1 Prominent Methods for Model Selection

Among the standard methods compared in this work two general approaches can be observed. While Generalized Cross-Validation (GCV) [23] is based on data re-sampling, Finite Prediction Error (FPE) [2] and Schwartz's Criterion (SCH) [96] attempt to penalize model complexity. The use of FPE as opposed to the Akaike Information Criterion (AIC) [3] is justified since FPE is specially derived under the assumption that the distributions of the predictors used in learning and prediction, is identical. Furthermore, FPE and AIC give almost the same inference for this class of problem [91, section 8.4], as borne out by some unreported results of our own. As described in Wallace [120] (replicating the problem framework from Cherkassy et al. [18]), the selection process for these model-selection methods is to choose the polynomial which minimizes

$$g(p, N) * SE(f(d, x))$$

where $p = (d + 1)/N$, and $g(\cdot, \cdot)$ is a function characteristic of the given method of inference. The function $g(\cdot, \cdot)$ is known as a penalty function since it inflates the training error (average residual sum of squares). The following characteristic penalty functions are derived from these benchmark approaches used in this comparative study:

1. Finite Prediction Error (FPE), $g(p, N) = (1 + p)/(1 - p)$
2. Schwartz's Criterion (SCH), $g(p, N) = 1 + 0.5 \log(N) * p/(1 - p)$
3. Generalized Cross Validation (GCV), $g(p, N) = 1/(1 - p)^2$

5.2.2 VC Dimension and Structural Risk Minimization

As defined earlier in section 5.2 (from equation 5.2), the *prediction risk* is the expected performance of an estimator on new samples. The Vapnik-Chervonenkis theory [112] provides non-asymptotic "guaranteed" bounds for the prediction risk of a model based on the concept of the VC-dimension [113]. Generally speaking, the VC-dimension [112, 113] is a measure of model complexity. For a given set of functions the VC-dimension is the number of instances that can be "shattered" – i.e., all possible subsets of the instances(from some data domain) being partitioned from their complement subset by functions from this set. For example, in the binary classification case, the VC-dimension is the maximum number of instances m which can be separated into two classes in all possible 2^m ways by using functions from the hypothesis space. The VC-dimension for the set of polynomial functions of degree d can be shown [113] to be equal to $(d+1)$.

The Structural Risk Minimization (SRM) principle [113, 114] is based on the well-known assumption that, in order to infer models with high generalization ability, we need to define a trade-off between the model complexity and goodness of fit to the data. Employing the VC-dimension as the measure of model complexity the SRM principle attempts to achieve this trade-off and avoid *over-fitting*.

According to Vapnik [113], in order to choose the polynomial $f(d, x)$ of the best degree d, one can minimize the following function based on the Structural Risk Minimization principle:

$$R(\underline{a}, d) = \frac{\frac{1}{N}\sum (y_i - f(d, x_i))^2}{\left(1 - c\sqrt{\xi_N}\right)} \tag{5.3}$$

where

$$\xi_N = \frac{V\left(\log \frac{N}{V} + 1\right) - \log \eta}{N}$$

In the expression above, $R(\underline{a}, d)$ is the estimate of the prediction risk of a polynomial of degree d and coefficients $\underline{a}$. The numerator of the Right Hand Side of (5.3) is the average squared error, namely $(1/N)\sum_{i=1}^{N}(y_i - f(d, x_i))^2$, achieved by a polynomial of degree d and set of co-efficients $\underline{a} =< a_0, ..., a_d >$ on N training examples, and the denominator is $1 - c\sqrt{\xi_N}$. ξ_N is the error term where N is the number of examples, c is a constant that reflects the tails of the training error distribution and V is the VC dimension for $f(d, x)$. This approach also takes into account the confidence interval for the prediction. Specifically, it provides an upper bound for the estimate, $R(\underline{a}, d)$, of the prediction risk. The inequality in Equation (5.3) then holds with probability (1-η), where η represents a confidence interval for the prediction [113]. In our empirical evaluation we have used $V = (d + 1)$, since the VC dimension for a polynomial of degree d is (d + 1) and, as suggested in [114], we have elected to use c = 1. In this evaluation we employ the error term ξ_N as described in equation (5.3) with $\eta = 0.125$ (this value of η was derived from Vapnik's book [113]). In empirical comparisons done by Wallace [120] the confidence interval η was implemented as a function of the sample size ($\eta = \frac{1}{\sqrt{N}}$). The recent version [114] from equation (5.3) employs a fixed user-defined value. This application of a fixed confidence interval improves the predictive performance of the older approach [113, 120, 115] at least in specific cases.

5.2.3 Minimum Message Length Principle

Minimal Length Encoding techniques like the Minimum Message Length (MML) [122, 127, 125] and Minimum Description Length (MDL) [84, 86, 87] principles have been popular due to their successful application in model selection. MML is an invariant Bayesian principle [123, 127, 125] based on the information-theoretic assertion that the best model for a given set of data is one that allows the shortest joint encoding of the model and the data, given the model. MML seeks to minimize the length of a "message" which encodes the data (in this problem the training y-values) by first stating a probabilistic model for the data, and then encoding the data using a code which would be optimal were the stated model true. MML has been shown to provide robust and highly competitive model selection in comparison to various standard model selection criteria (including AIC) within the polynomial degree selection framework [5], although we improve this even further here by our use of an orthonormal basis. A comparison with the MDL [87] principle of Rissanen, with A. N. Kolmogorov's notion of complexity [53], and with Chaitin's notion of algorithmic information theory [13] is given in Wallace and Dowe [125].

In our case, since the model is assumed to be a polynomial function with Gaussian noise, the model description need only specify the degree of the polynomial, the co-efficients of the polynomial, and the estimated variance (or, equivalently, SD) of the noise. In the general case, the best MML polynomial for any degree is the one that has the shortest two-part message length, of which the first part describes the polynomial in terms of its degree d, co-efficients $\underline{a}$ and estimated variance v, while the second part describes the data (via the ϵ_i) using the given polynomial. An important point to note is that our encoding system uses the degree of the polynomial rather than the number of non-zero coefficients. One reason for this is that the num-

ber of non-zero coefficients will depend upon whether we choose the basis $1, x, x^2, x^3, ...$, the orthonormal basis for integration over the region $[-1, 1]$ or another basis. However, the degree of the polynomial will remain the same regardless of this choice of basis.

In the current experiment, the coefficients are estimated using the maximum likelihood technique, since the difference from MML estimation is small for this problem and the other methods all advocate using the maximum likelihood estimate. (For examples of problems where MML and maximum likelihood estimation are substantially different, see, e.g., Wallace and Freeman [128], and Wallace [119].) The following sections provide details of the actual MML encoding scheme.

Encoding the Model with Prior Beliefs

MML is a Bayesian principle and thus requires the formal specification of prior beliefs. We start by considering the degree of our polynomial models. All degrees from 0 to 20 are considered equally likely a priori, so each degree is coded with a code word of length log(21) *nits*, or natural bits. On our assumption, the coding of the model degree has no influence on the choice of model as all degrees have the same code length .

In coding estimates of the noise variance v, and the polynomial coefficients $\underline{a}$, some scale of magnitude may be assumed. Here, we use the second order sample moment of the given y-values to determine such a scale by defining

$$\vartheta = \frac{1}{N} \sum_{i=1}^{N} (y_i)^2 \tag{5.4}$$

In encoding a polynomial model of degree d, we suppose that the noise and each of the $(d+1)$ degrees of freedom of the polynomial may be expected

a priori to contribute equally (in very rough terms) to the observed variance ϑ of the y-values. Defining

$$u = \sqrt{\left(\frac{\vartheta}{d+2}\right)} = \sqrt{\frac{\sum_{i=1}^{N} y_i^2}{N(d+2)}} \tag{5.5}$$

we assume the Standard Deviation s, of the "noise" v, where $s = \sqrt{v}$, has a Negative Exponential prior density with mean u, and that each of the coefficients $(a_0, \ldots, a_j, \ldots, a_d)$ of the polynomial has independently a $N(0, u^2)$ prior density. If the coefficients were the usual coefficients of the successive powers of x, the latter assumption would be unreasonable, and highly informative. Instead, we represent a d-degree polynomial as

$$f(d, x) = \sum_{j=0}^{d} (a_j Q_j(x)) \tag{5.6}$$

where the set $Q_j(\cdot) : j = 0 \ldots d$ is a set of polynomials, $Q_j(\cdot)$ being of degree j, which are orthonormal under integration on the interval $[-1, 1]$. The orthonormal polynomials represent effectively independent modes of contributing to the variance of $f(d, \cdot)$, and therefore it seems reasonable to assume independent prior densities for the co-efficients $\{a_j : j = 0, \ldots, d\}$. With these assumptions, the overall prior density for the unknown parameters $\{s, \{a_j\}\}$ is

$$h(s, \underline{a}) = (1/u)^{(-s/u)} * \prod_{j=0}^{d} \left(\frac{1}{\sqrt{2\pi}u}\right) e^{(-a_j^2/(2u^2))} \tag{5.7}$$

The amount of information needed to encode the estimates s and $\underline{a}$ depends on the precision to which these are stated in the "message". Specifying the estimates with high precision leads to an increase in the model part of

the message length while lower precision leads to lengthening of the data part [127, 124, 125]. The optimum precision, as described in [127], is inversely proportional to the square root of the *Fisher information* $F(s, \underline{a})$, which in this case is given by [120, 115]

$$F(s, \underline{a}) = 2 \left(\frac{N}{s^2} \right)^{(d+2)} |M| \tag{5.8}$$

where M is the co-variance matrix of the orthonormal polynomials evaluated at the given x-values:

$$M[j,k] = (1/N) \sum_{i=1}^{N} Q_j(x_i)\, Q_k(x_i) \tag{5.9}$$

Encoding the Data

Once the model polynomial $f(d, \cdot)$ and the noise standard deviation s have been stated, the given y-values can be coded simply by their differences from $f(d, \cdot)$, these differences being coded as random values from the Normal density $N(0, v)$. The message length required for this is then given by [120, 127]

$$DataMessLen = \left(\frac{N}{2} \right) \log(2\pi v) + (1/2v) * SE(f(d, \cdot)) \tag{5.10}$$

The Total Message Length

The total message length is approximately given by [127, 125]

$$\begin{aligned} MessLen = &- \log h(s, \underline{a}) + 0.5 \log F(s, \underline{a}) + DataMessLen \\ &- ((d+2)/2) \log(2\pi) + 0.5 \log((d+2)\pi) \end{aligned} \tag{5.11}$$

where the last two terms arise from the geometry of an optimum quantizing lattice in $(d+2)-$space. The noise variance $v = s^2$ is estimated as

$$\hat{v}_{MML} = \left(\frac{1}{(N-d-1)}\right) \sum_{i=1}^{N} (f(d, x_i) - y_i)^2 \tag{5.12}$$

and the co-efficients of $\underline{a}$ are estimated by conventional least-squares fit. These estimates do not *exactly* minimize the message length *MessLen*, but, for this problem the difference is small. The MML method selects that degree d which minimizes *MessLen* as calculated above in equation (5.11). Recent theory [33] suggests that the MML estimator will closely approximate the estimator minimizing the expected Kullback-Leibler distance.

5.3 Experimental Evaluation

The following discussion outlines the experimental procedure employed in this empirical evaluation. For each experiment a target function is selected in the required interval [-1,1]. The noise is defined in terms of the "signal-to-noise ratio" (SNR), where SNR is defined as the second moment of the target function, $t(x)$, about zero, divided by the noise variance v. The number of training data points N and the number of evaluations (training and test runs) are specified.

An experiment consists of (averaging over) 10000 evaluations. In each evaluation or "case", N training examples and $m = \max(N, 50)$ test examples are generated. For each "case", all least-squares polynomial approximations of degrees up to 20 are found by standard techniques, and the training error, $SE(d)$, computed for each of the 21 polynomials. These training error values are then given to each of the model selection methods being compared and

each method selects its preferred choice among the polynomials.

The prediction risk for a method in the current case is then the average squared prediction error (SPE) achieved by the polynomial chosen by the method on the test data. Note again that all selection methods must choose from the same set of 21 polynomials, and their choices are evaluated on the same set of test data. Thus if two selection methods choose the same degree in some case, they are choosing the same polynomial and will incur the same SPE for this case.

After the 10000 cases of an experiment have been completed, we obtain for each selection method:

- its Squared Prediction Error (SPE), averaged over all cases (10000); and,
- the Standard Deviation of its SPE.

The five selection methods – namely, MML, SRM, FPE, SCH and GCV – were evaluated on six target functions $t(x)$ in the interval [-1,1]. The methods were tested under different scenarios. First, the noise level was kept constant with varying numbers of training points and, then the noise levels were varied with a fixed number of training points. For each method evaluated under a particular scenario, comparisons were made on the basis of squared prediction error (SPE) and standard deviation of the SPE. As an artificial yardstick, we also include a method called "BEST". The "BEST" polynomial is obtained by selecting from the 21 candidate least-square fit polynomials the polynomial which would give the smallest SPE as calculated on the test data. Thus, BEST shows the best performance which could possibly be obtained

in principle; but these results are of course unrealizable, as they are based on knowledge of the noise-free values of the target function at the test x_i's.

5.4 Analysis of Results

In the experimental evaluation we consider two sets of target functions. In the first the target functions belong to the family of polynomials of maximum degrees of 20, while the second set consists of non-polynomials. Figures 5.3-5.8 include plots of results from experiments with six target functions. Each figure presents two scenarios. In the first case the sample size is increased and the standard deviation of the noise is kept constant, and vice versa in the second case. Some of the prediction methods can be seen to have a squared prediction error typically of the order of 100 times that of MML. Due to this relatively poor performance of those estimation techniques, squared prediction error is plotted on a logarithmic scale. As mentioned in section 5.3, we also plot the standard deviations from some of these simulation runs with sample size fixed at $N = 40$ and noise varying. Of the six target functions considered (see section 5.4.1), when $N = 40$, we found the ratio of FPE squared error to MML squared error to be the smallest for the smooth SIN function, closely followed by the discontinuous function. We found the ratio to be largest for the low order polynomial. In Figure 5.9, we plot results comparing the standard deviation of SPE of all methods on the SIN function and the low-order polynomial.

In general, from the examples included in this study and other tests it can be observed that MML and SRM give lower errors than other methods in most cases. The results clearly show that none of the methods FPE, SCH, or GCV is competitive with SRM or MML, except under conditions of low noise and large N, when all methods give similar results. An interesting observation

is that the SRM method is based on theory which does not assume that the target function belongs to the model family from which the approximation is drawn (in this case, the polynomials). MML, however, is in part motivated by theory which does make this assumption. It is curious that the only target/test conditions under which SRM performs comparably with MML are ones where this assumption is largely valid. Furthermore, an examination of the percentiles of the error distribution suggests that the MML and SRM-based methods usually have similar median errors. The high SPE of the SRM method seems to be the result of occasional very large errors using the selected model. Evidence of this is given in Figure 5.10.

Denison et al. [29] present a Bayesian approach to estimating a variety of curves using piecewise polynomials. Some reviewers suggested a comparison with their method. Although their approach is far more flexible due to the generality of the model class and the resulting variety of functions that their system can fit, we offer a limited comparison[1] with our MML-based orthonormal polynomial approach. The comparisons are not equivalent since in our approach the squared prediction error is computed from test data while Denison et al. take a posterior sampling approach.

[1]In Figure 5.11, we compare MML with the piecewise polynomial fitting (PP) on the low order polynomial. The samplesize is $N = 200$, the SNR is 150 (recall Section 5.3) and the standard deviation of the noise is $s = 0.317$. The plot presents the SPE achieved by both methods for each of 100 runs of the experiment. Figure 5.12 presents a similar scenario for the high order polynomial. The samplesize is 200, the SNR is 100 and the standard deviation of the noise $s = 0.724$. The MML fit was clearly better that the PP fit for these two polynomial cases, especially for the high-order polynomial, where the squared prediction error was more than a hundredfold worse for PP than for MML.

5.4.1 Target Functions

In evaluating the five polynomial selection methods we consider the following two polynomial and four non-polynomial target functions. The following polynomial target functions are presented in Figure 5.1.

1. **A Higher-order Polynomial**: $y = 0.623x^{18} - 0.72x^{15} - 0.801x^{14} + 9.4x^{11} - 5.72x^9 + 1.873x^6 - 0.923x^4 + 1.826x - 21.45$;

2. **A Lower-order Polynomial**: $y = 9.72x^5 + 0.801x^3 + 9.4x^2 - 5.72x - 136.45$;

Non-polynomial Target Functions

It is interesting to observe the performance of the different methods when the target function are not polynomials. The task here is challenging as we seek the best polynomial approximation. The following target functions were utilized in the comparison, with $-1 \leq x \leq 1$, and are plotted in Figures 5.1 and 5.2.

1. **SIN**: $y = (\sin(\pi(x+1)))^2$;

2. **LOG**: $y = \log(x + 1.01)$;

3. **FABS**: $y = |x + 0.3| - 0.3$; and,

4. **DISC**: if $(x < 0.0)\, y = 0.1$; else $y = 2x - 1.0$.

5.4.2 The "Guaranteed" SRM Bound: Is it Loose ?

In this section we raise some issues about the application of the Structural Risk Minimization Principle (SRM) to the current problem framework. The

model selection methods based on the SRM principle define a "guaranteed" bound on the prediction risk of a model, where the prediction risk, as defined earlier, is the expected squared prediction error achieved by the selected model. The SRM bound states that the prediction risk for a chosen model will not be exceeded with probability of at least $1 - \eta$. In our case, equation (5.3) is the upper bound on the prediction risk and it is "guaranteed" [113] with a probability of $1 - \eta = 1 - 0.125 = 0.875$ (see section 5.2.2). Based on our empirical analysis we find that this bound is not exceeded, with probability greater than $1 - \eta$, for any fixed order of polynomial and any sample size, noise ratio or target function. In fact, the probability of bound violations is usually much less than η.

However, the polynomial model selected by the SRM method from the 21 families of polynomial often exceeds the bound with a probability greater than η. For example, the bound is violated in 96.5% of the cases of an experiment on the smooth target function with a sample size and SNR ratio (see section 5.3) of 10. This is a typical result for model selection with small sample sizes. Thus the SRM bound seems to be plausible only when applied to selecting models from the same family. Once this assumption is relaxed, the SRM measure cannot be guaranteed to provide an upper-bound on the prediction risk. In this context another important observation is that the VC theory does not provide any information on the magnitude of the prediction error when this upper-bound is exceeded.

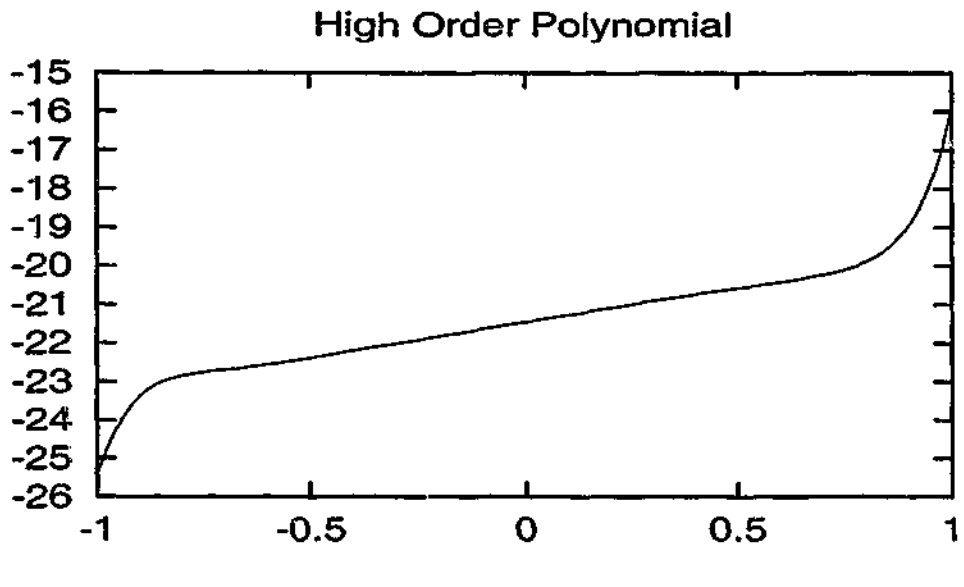
Finally, we find that, in its application to the current model selection framework, the SRM principle is not paying any attention to the variance of the selected model. On an examination of the cases where the performance of the SRM-based models is worse than usual we find that the estimated variance

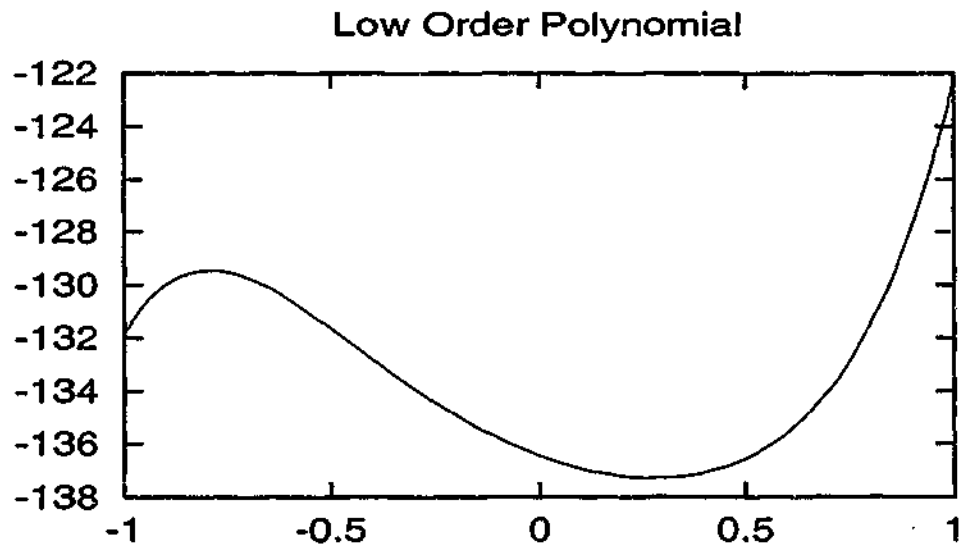
of the approximating polynomial is orders of magnitude larger than the second moment of the training samples. Thus the SRM principle is accepting some clearly unreasonable models. As an example, the plots in Figure 5.10 present a comparison of the variance of the worst model (in terms of SPE) selected by the SRM and MML methods over the 10,000 runs with their average training errors. The model selection methods were evaluated on the sine function from Section 5.4.1 with a fixed SNR ratio and variable sample size.

5.5 Summary

This chapter presents a system for learning polynomial functions from noisy datasets. The empirical studies demonstrate the plausible performance of the Minimum Message Length principle in comparison to other methods. Based on our empirical evaluation we find that the MML approach in general provides more robust and accurate model selection than the other methods in the current problem framework, especially FPE (and AIC), SCH and GCV. Finally, we have provided some analysis of the different model selection methods under varying scenarios. It is essential to note that the comparison with the polynomial splines (Denison et al.) is not exhaustive in any sense.

Figure 5.1: Plots of Target Functions





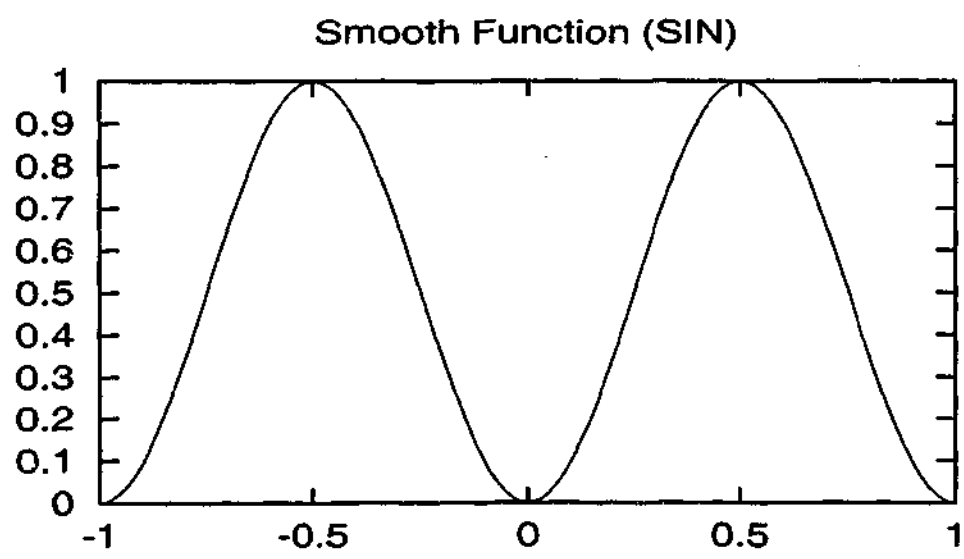
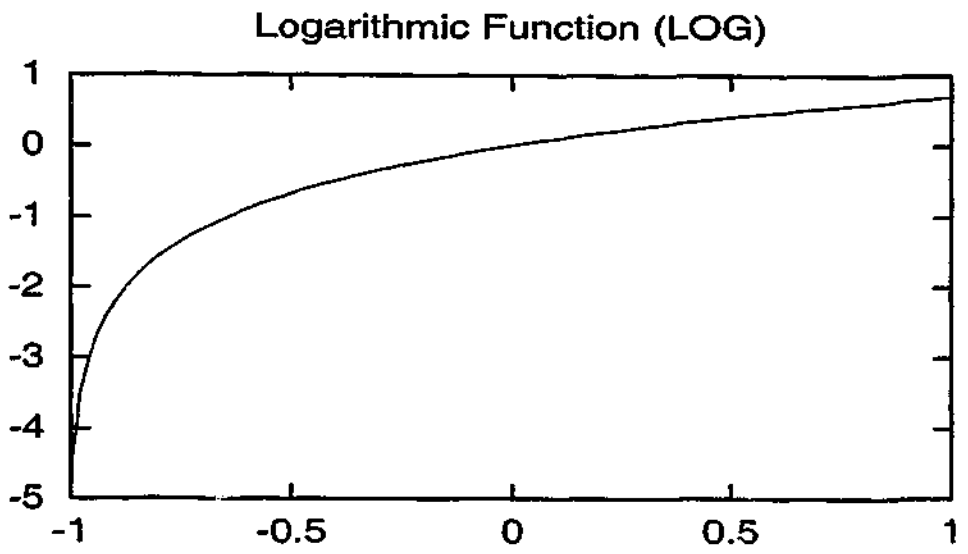
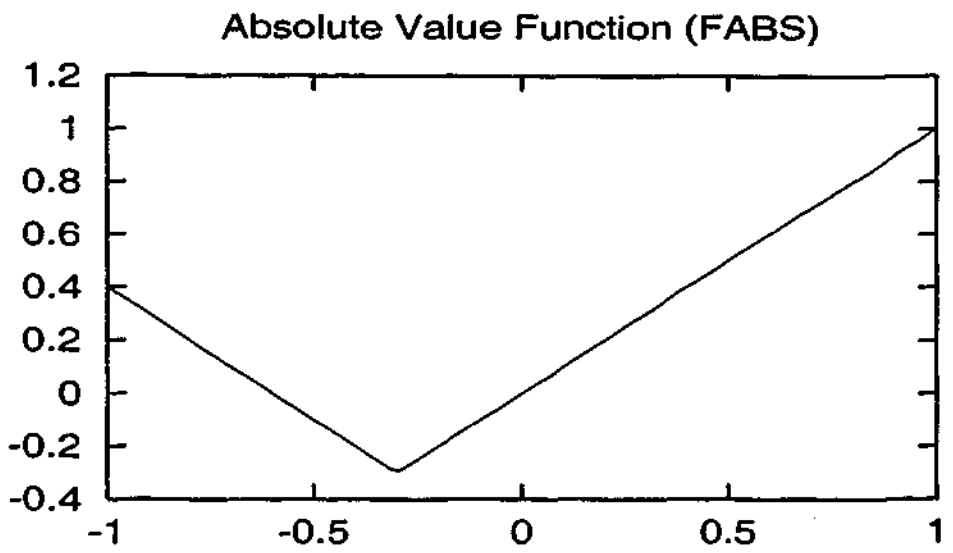


Figure 5.2: Plots of Target Functions





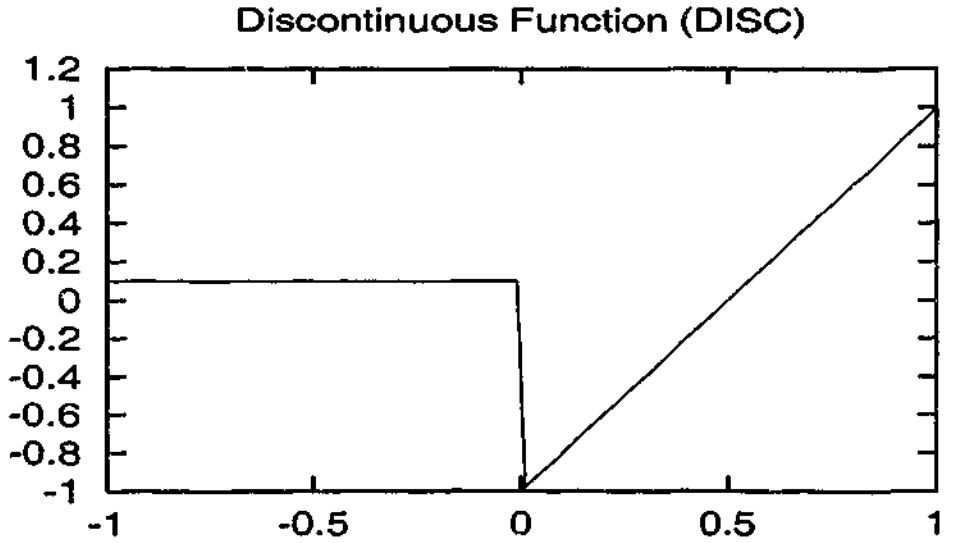


Figure 5.3: Comparing methods on Squared Prediction Error (SPE)

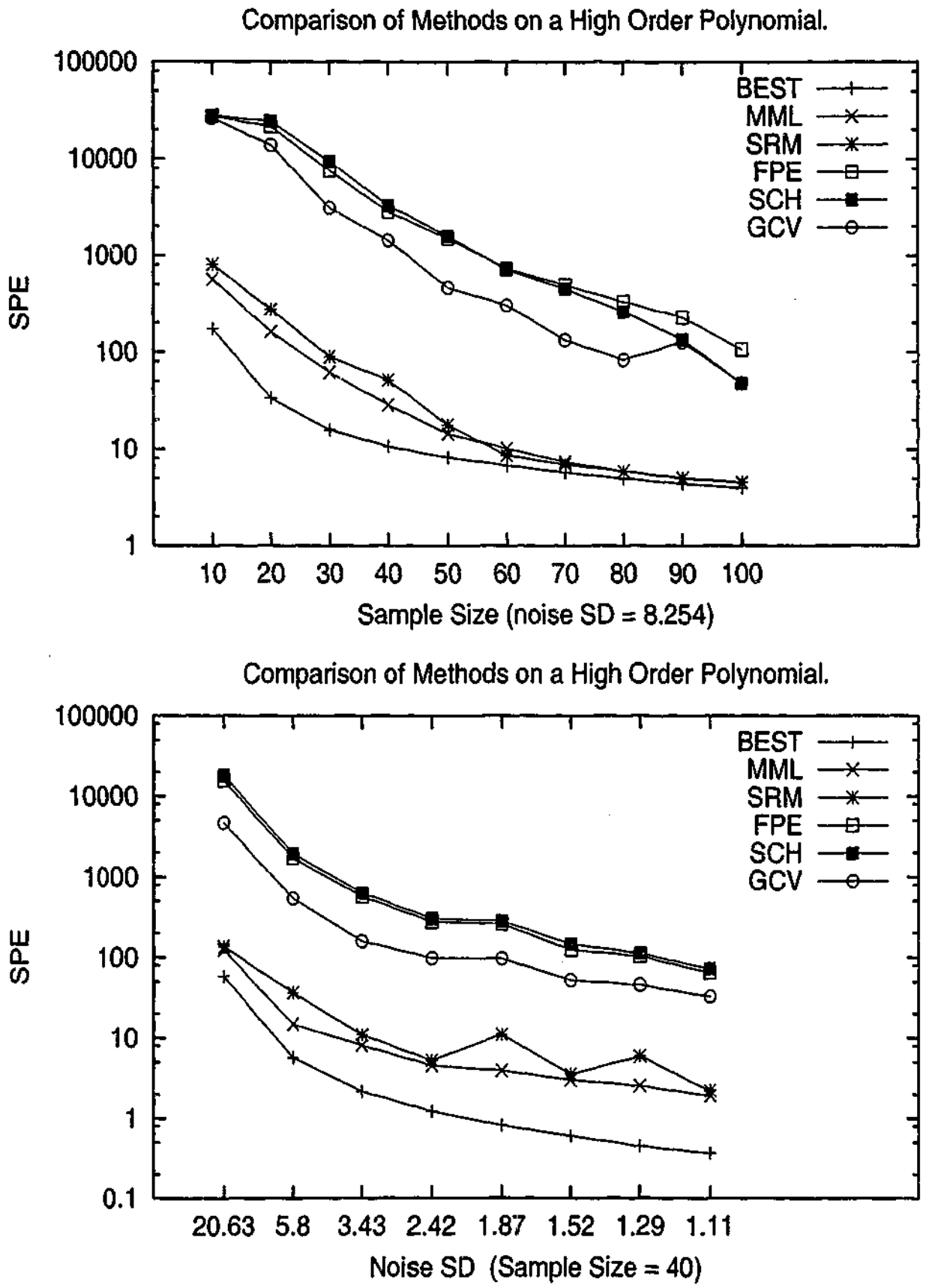


Figure 5.4: Comparing Methods on Squared Prediction Error

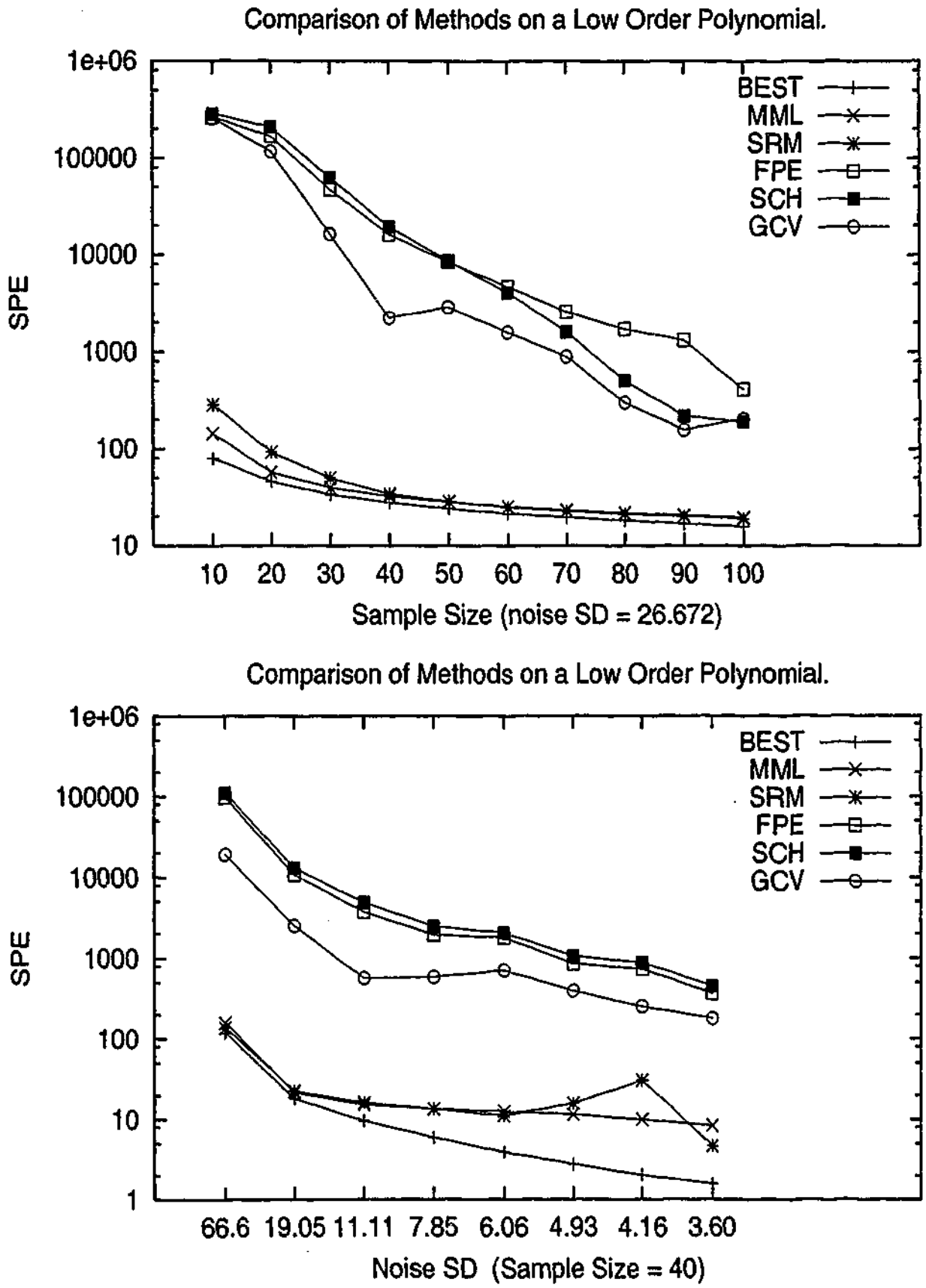


Figure 5.5: Comparing Methods on Squared Prediction Error

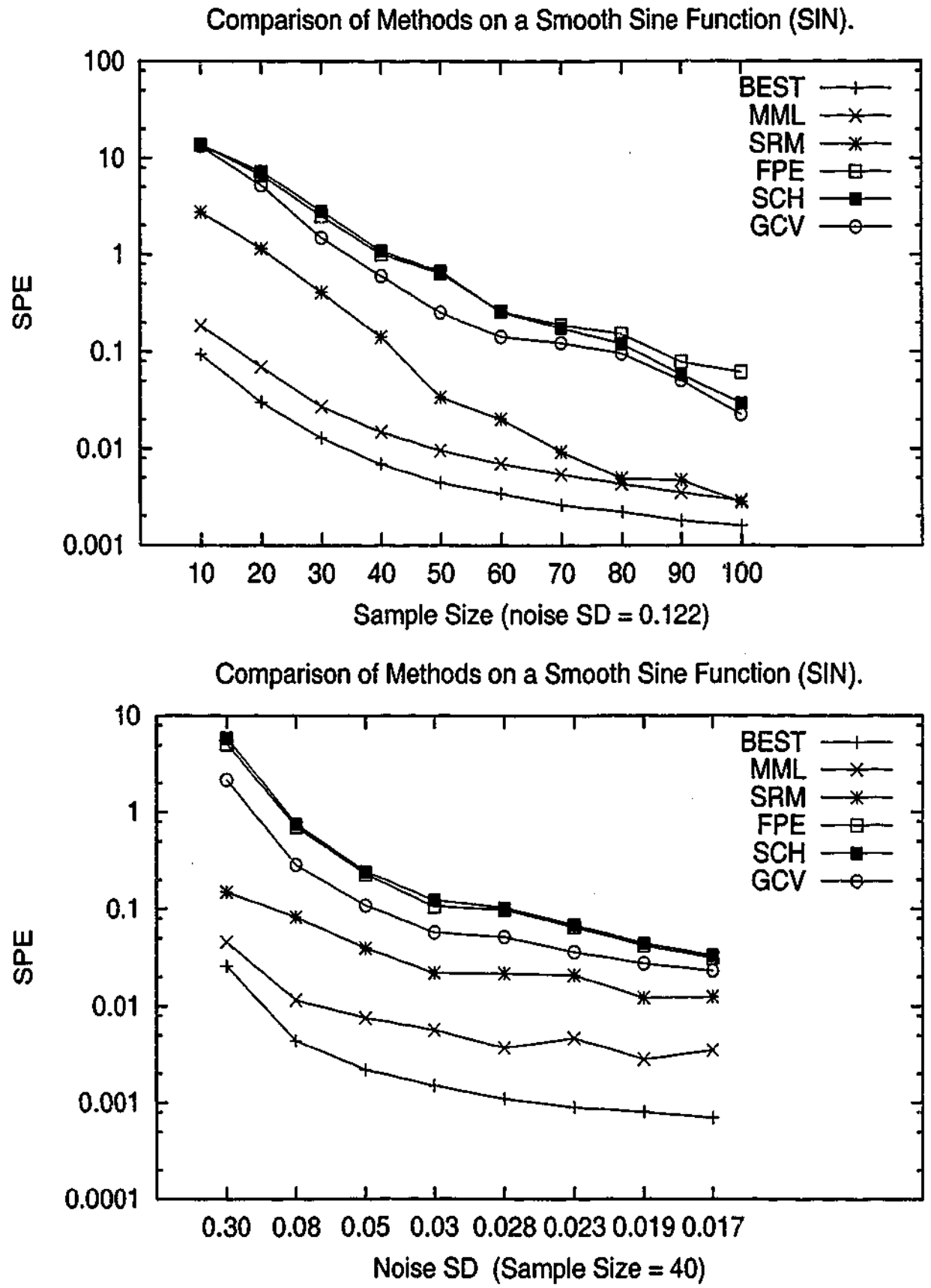


Figure 5.6: Comparing Methods on Squared Prediction Error

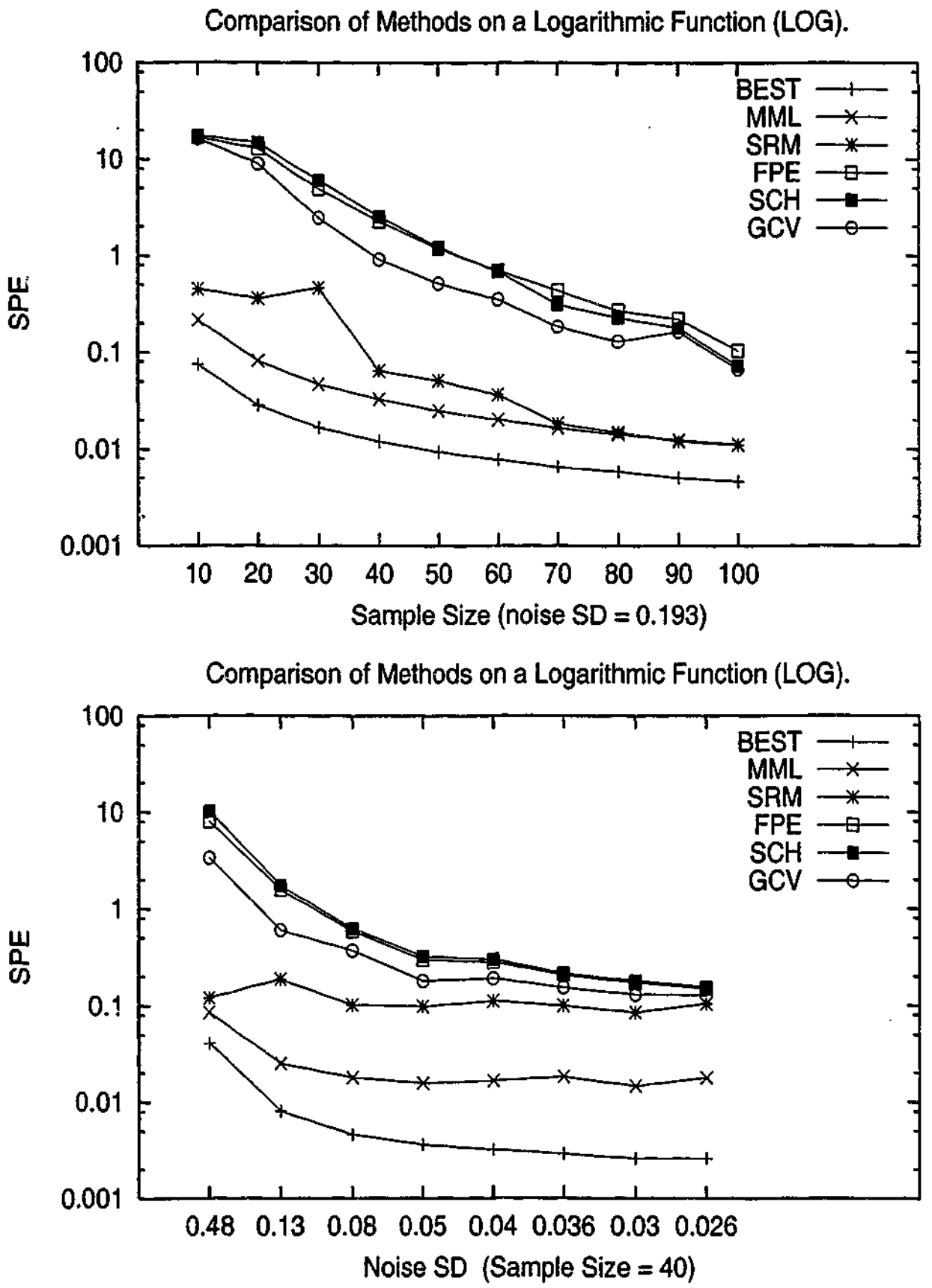


Figure 5.7: Comparing Methods on Squared Prediction Error

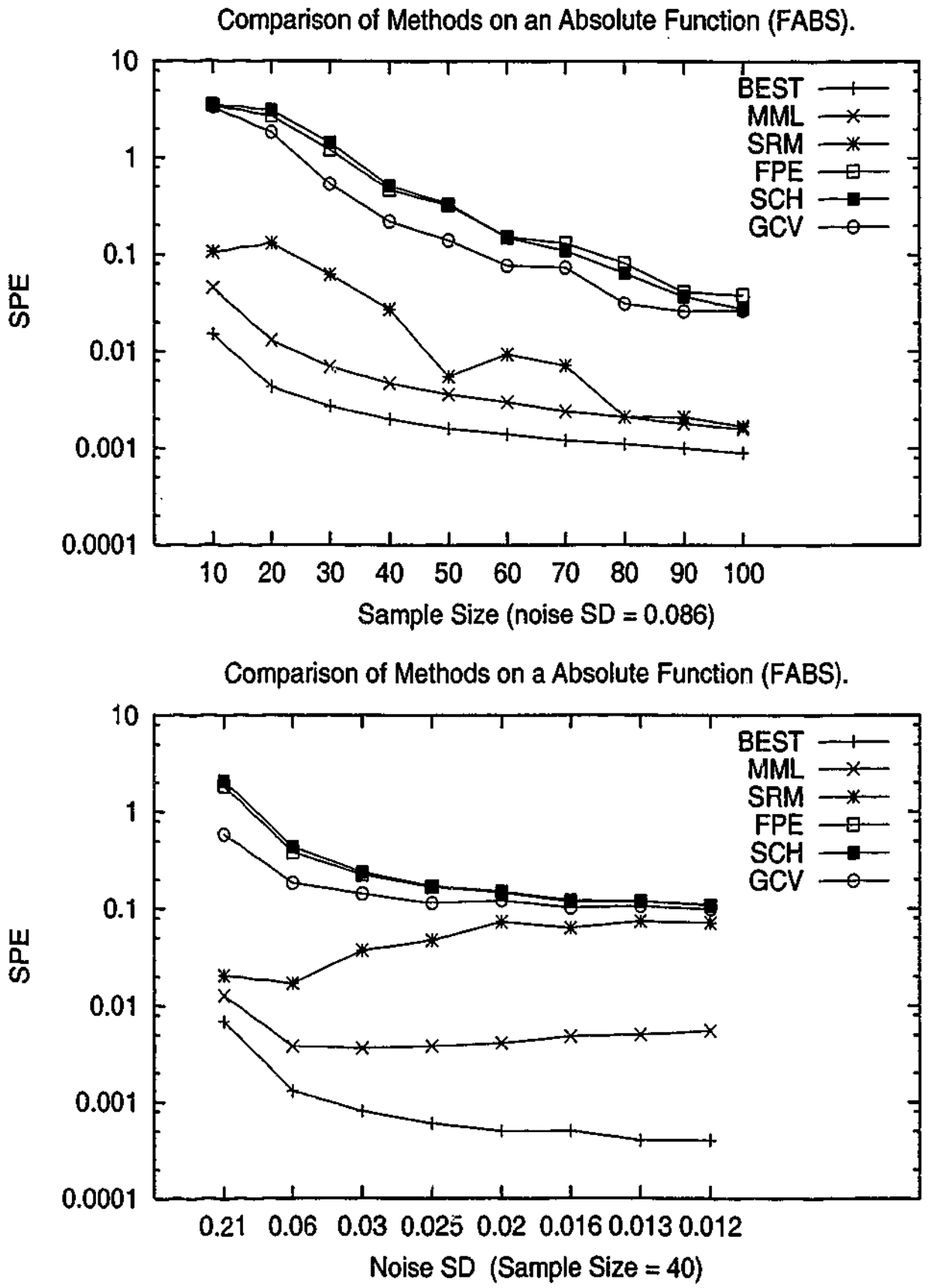


Figure 5.8: Comparing Methods on Squared Prediction Error

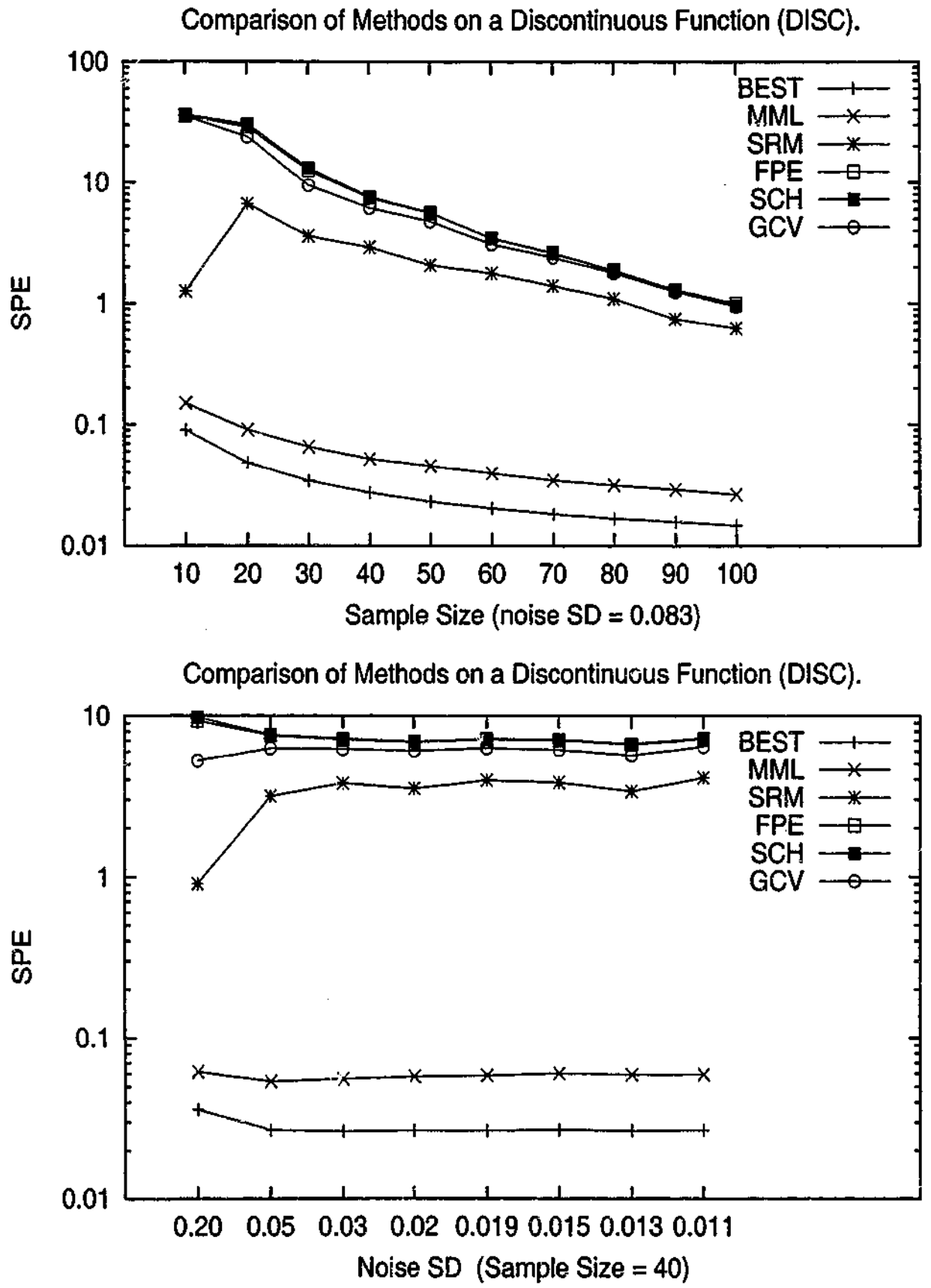


Figure 5.9: Standard Deviation of Squared Prediction Error

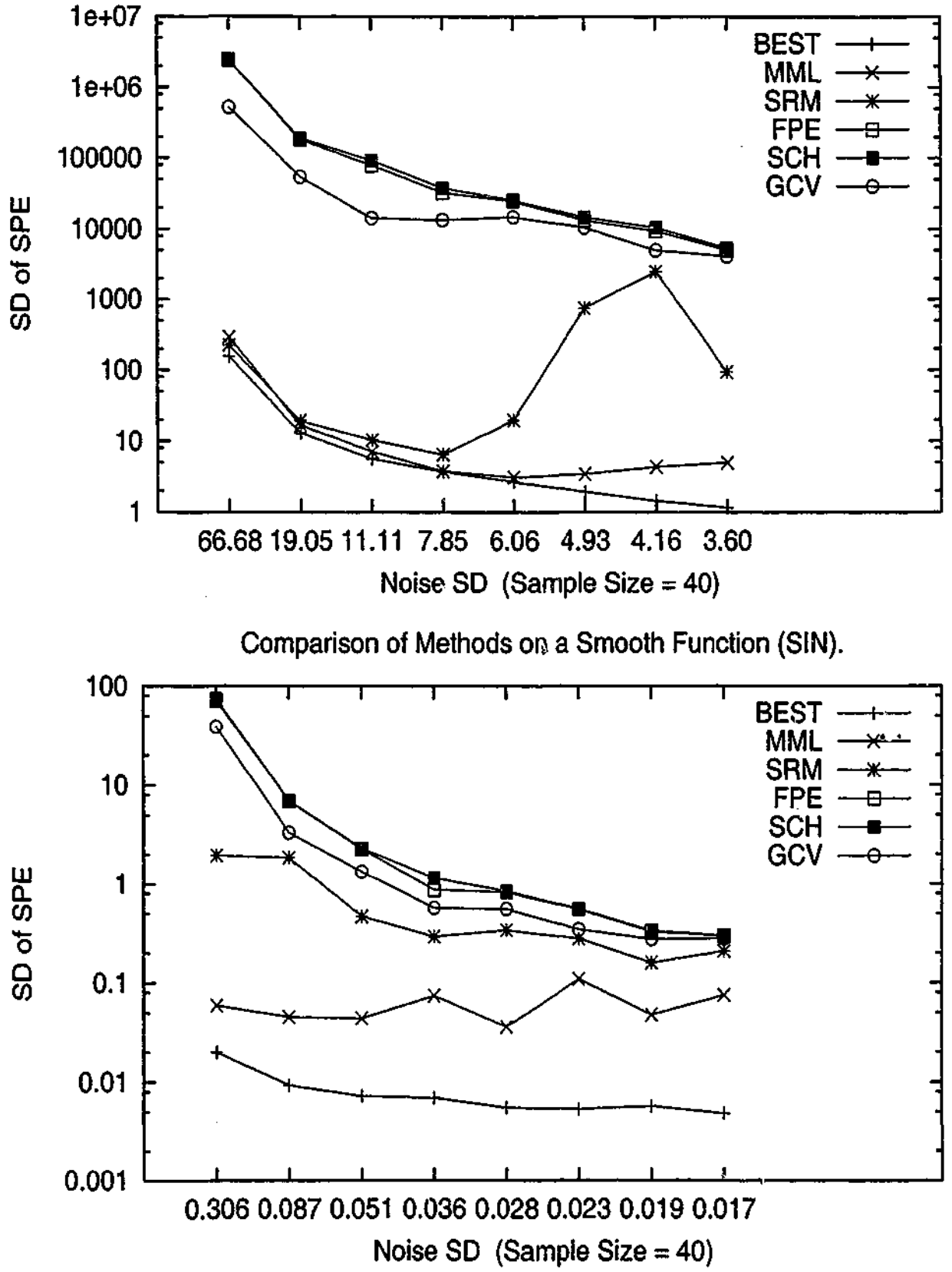
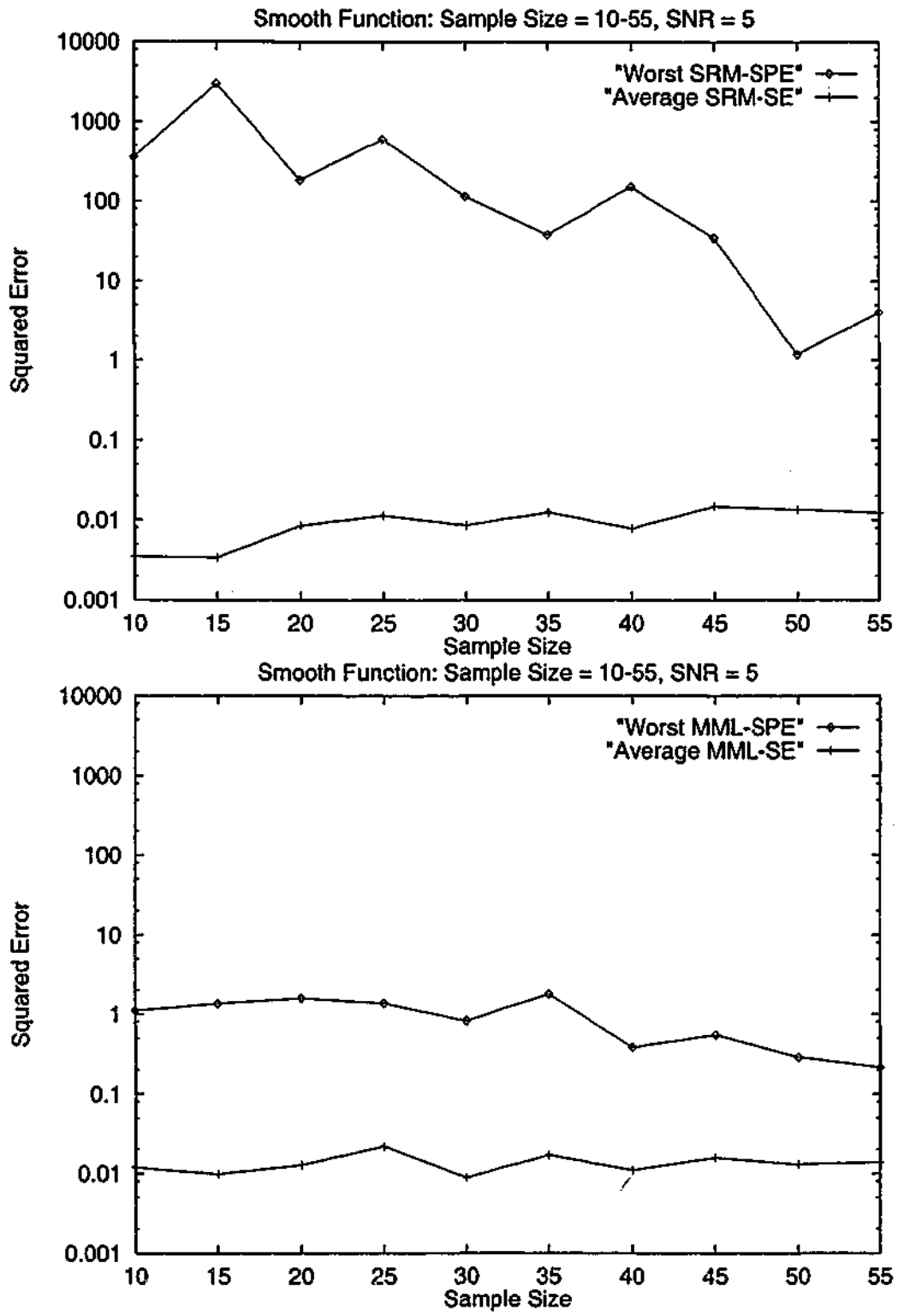


Figure 5.10: A Comparison of the SPE (test data) and SE (training data) of the Worst Models Selected by SRM and MML



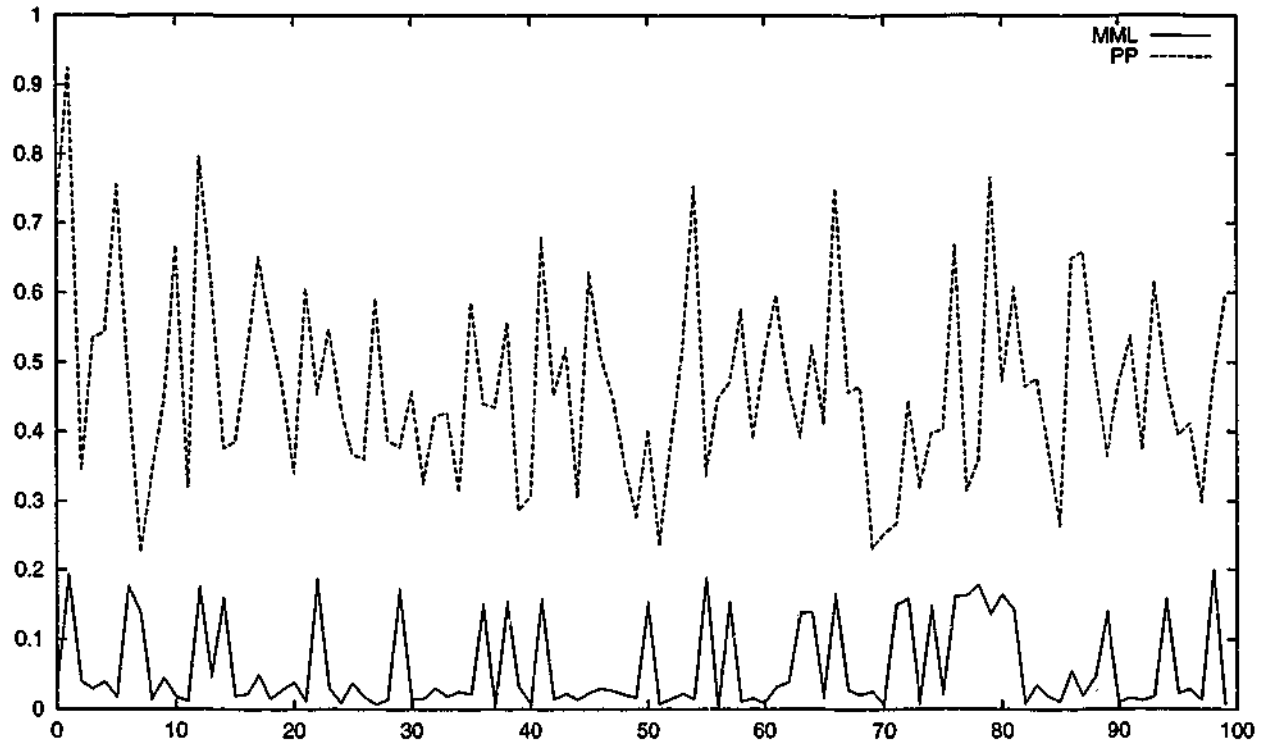


Figure 5.11: Comparing MML and Piecewise Polynomial Fitting on a Low Order Polynomial. X axis represents the run count (up to 100) while on the Y axis we have the squared prediction error (SPE).

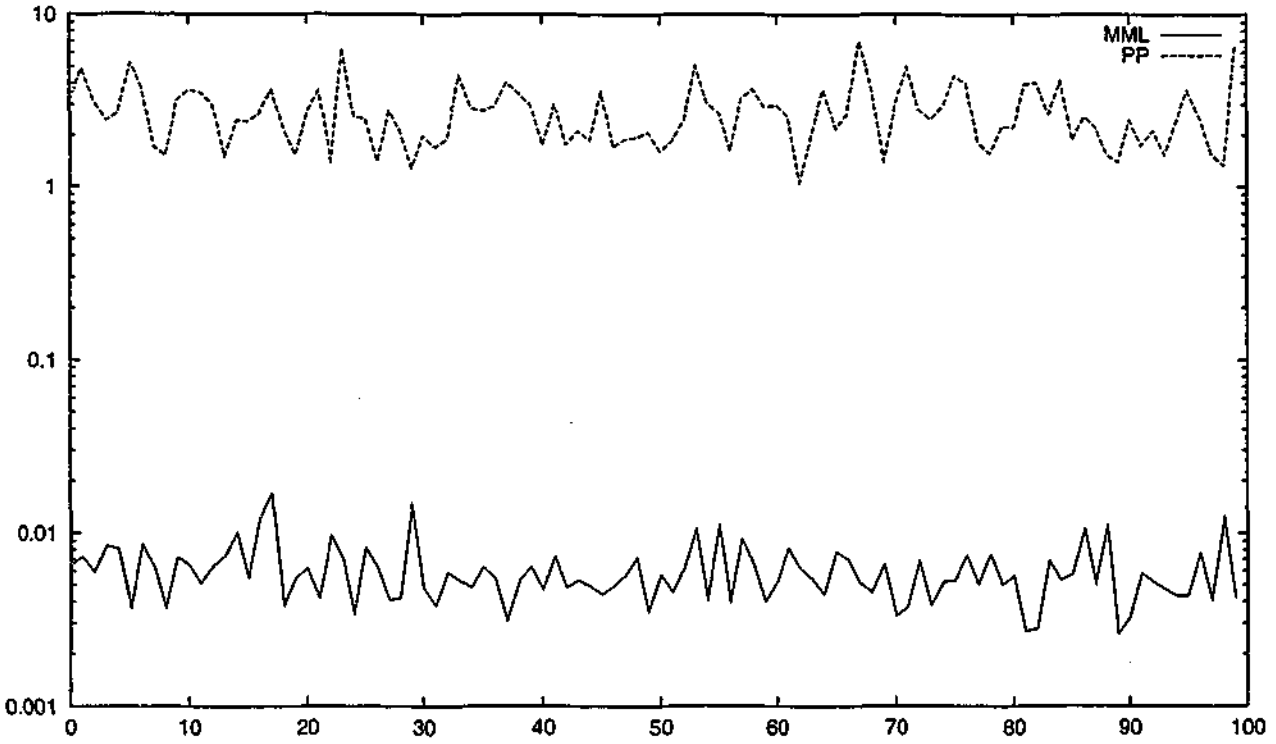


Figure 5.12: Comparing MML and Piecewise Polynomial Fitting on a High Order Polynomial. X axis represents the run count (up to 100) while on the Y axis we have the squared prediction error (SPE).

Chapter 6

Segmentation of Binary Sequences

The chapter describes another significant contribution of this thesis. That is, the development of a system for learning from binary sequences. A specialised framework is used in defining the problem domain. The MML principle is employed in deriving an efficient scheme for model selection. The learning technique is again presented in the context of a comparison with other modern methods, including a detailed empirical evaluation.

6.1 Introduction

The segmentation problem occurs when there is a need to partition some data into distinct homogeneous regions. The specialized segmentation framework considered in this work was introduced by Kearns et al. [51] and employed in an empirical and theoretical comparison of model selection methods. Their motivation for selecting this problem was that, while being non-trivial, it appeared to Kearns et al. to permit exact solution of the optimizations re-

quired by the different methods and hence seemingly offered a comparison of the methods untainted by questionable mathematical approximations. The development of our technique for this specialised segmentation problem resulted in two primary conclusions:

- The earlier application of the Minimum Description Length (MDL) principle as employed by Kearns et al. in their empirical evaluation was sub-optimal.

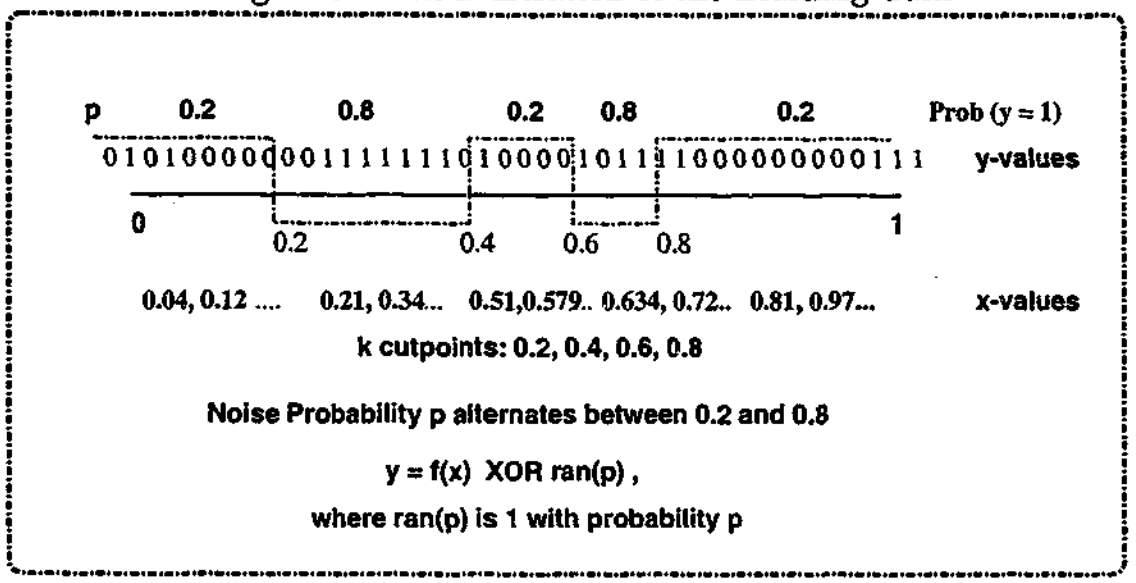
- Optimal coding resulted in the superior performance of two methods, the former based on a revised version of the MDL method (from [51]) and the latter based on the Minimum Message Length (MML) principle.

We commence with a brief summary of the problem. An unknown Boolean function $f(x)$ is defined on the real interval $0 < x < 1$. The interval is partitioned into $(k + 1)$ sub-intervals by k "cut points" $\{c_j : j = 1..k\}$ which are uniformly and randomly distributed in [0,1] and indexed so that $c_j < c_{j+1}$, $(j = 1..k-1)$. The function $f(x)$ is defined to be 0 in even-numbered sub-intervals, and 1 in odd-numbered sub-intervals; the sub-intervals are numbered from 0 to k so that cutpoint c_j separates sub-intervals $j-1$ and j. Data is generated from this model at N sample points $\{x_i : i = 1..N\}$. The Boolean datum y_i generated at x_i differs from $f(x_i)$ with probability $p < (1/2)$. Thus, the probability that $y_i = 1$ alternates between p and $(1-p)$, depending on whether x_i lies in an even or odd sub-interval.

The inference task, termed the "intervals model selection" problem by Kearns et al. [51], is to infer an approximation to the function $f(x)$ from the data $\{x_i,\ y_i :\ i = 1..N\}$. That is, we wish to infer the number and position of the cutpoints (and, incidentally, estimate the unknown "noise" rate, p).

The intervals model selection problem was originally employed by Kearns et al. [51] in their evaluation of different model selection methods. These se-

Figure 6.1: An Illustration of the Learning Task



lection methods included the two penalty-based methods — Vapnik's Guaranteed Risk Minimization (GRM) [114] and a version (we designate KMDL) [51] of Rissanen's Minimum Description Length (MDL) Principle [85, 12], and also Cross Validation (CV) [103]. Kearns et al. [51] reported that MDL performed no better than cross validation in this task.

This chapter presents a novel technique applying MML for binary seqmentation. The experiments of [51] are replicated, omitting their implementation of Vapnik's method. The chapter also corrects an approximation in KMDL, obtaining a slightly improved method which we term "CMDL", although it is still a sub-optimal application (see section 6.6) of the MDL principle. Consistent with [51], we find both KMDL and CMDL to perform relatively poorly unless the sample is large (with CMDL somewhat superior). Finally, we develop a more correct MDL method, using the theoretical framework of the Minimum Message Length principle [122, 127, 125], with which we are more familiar. Our results confirm the poor behaviour of KMDL as seen in [51], but the MML method works well, and compares well with the cross-validation (CV) method which we implement in the same form as was done

in [51]. Readers interested in a comparison of MML, CV [103], and Vapnik's Structural Risk Minimization (SRM) [114] applied to univariate polynomial regression are referred to work by Wallace and Viswanathan [120, 115]. The MML principle has been previously applied to the problem of linear segmentation [6, 4, 73, 74, 56] and also in a different context (that of identifying superior sports performance) to segmentation of binary sequences [102, 54].

6.2 Definitions

This section presents standard definitions for all the terms used in this chapter.

1. S : training set, $\langle x_i, y_i \rangle i = 1, \ldots, N$.
2. N : size of S.
3. p : true probability (noise rate).
4. k : number of cuts.
5. d : number of alternations of label in S ($d = k + 1$).
6. m : number of sample points at which $h(x)$ differs from $f(x)$.
7. $f(x)$: true Boolean function from which S is generated.
8. $h(x)$: learning algorithm's estimate of $f(x)$ from S.
9. $H(x)$: binary entropy function given by $-(x \log x + (1-x)\log(1-x))$
10. $\epsilon(h)$ represents the generalization error of a hypothesis $h(x)$ with respect to the target function $f(x)$. $\epsilon(h) = KL(f(x) \parallel h(x))$, which is the Kullback-Leibler distance from $f(x)$ to $h(x)$.

11. $\hat{\epsilon}(h)$ denotes the training error of h on sample S.
$\hat{\epsilon}(h) = |\langle x_i, y_i \rangle \in S : h(x_i) \neq f(x_i)| / N.$

6.3 Kearns's Intervals Model Selection Problem

To test the various methods, Kearns et al. [51] chose a function $f(x)$ with 100 intervals each of length 0.01 (99 equally-spaced cutpoints). This function is the easiest to learn among all such functions with 99 cuts. A randomly spaced set of cuts would increase the chance that some subintervals would contain few (or no) sample points, making them much harder to detect. In our study, we employ generating functions with randomly-placed cuts — as in the original problem specification [51]. Note that none of the learning methods themselves assume approximate or exact equality of subinterval lengths: they all assume the locations of the cuts to be random.

A single test problem is generated from $f(x)$ by fixing a sample size N and a noise probability, p. Then, N x-values are selected from the uniform distribution in (0,1), and for each x_i a Boolean datum y_i is generated as $f(x_i)$ XOR $ran(p)$, where $ran(p)$ is a random noise bit with probability p of being 1. In other words, $y_i = f(x_i)$ with probability $1 - p$ and $y_i = 1 - f(x_i)$ with probability p. Many replications of a problem with given N and p are generated by making different random selections of the sample points and noise bits.

For a given sample S, Kearns et al. [51] see the "*essence*" of the learning problem as being the selection of a *model class*, where a class F_k is the set of all alternating functions with k cuts. That is, the essence is, to them, the estimation or selection of the number of cutpoints k. Within a class F_k, a

simple dynamic programming algorithm suffices to find the model function $h'_k(x)$ with maximum likelihood, i.e. with minimum training error $\hat{\epsilon}(h'_k)$. Of course, the locations of the cutpoints of $h'_k(x)$ are determined only to within the interval between two adjacent sample points. In this work, we take the cutpoint of $h'_k(x)$ which lies between x_i and x_{i+1} to be midway between the sample points.

The learning task thus reduces to selecting a model from the set of model functions $\{h'_k(x);\ k = 1..k_{max}\}$, where k_{max} is the largest number of cuts resulting in any reduction in training error. This set of models was then given to the two model selection methods based on GRM [114] and a version (KMDL [51]) of the MDL [85] principle. For the cross-validation method, the protocol was slightly different. The sample S was divided randomly into a 90% training set and a 10% validation set. A set of maximum-likelihood models $\{h_k(x);\ k = 1..k^c_{max}\}$ was developed from the training sample, and the cross-validation method selected from that set, the model with the lowest error on the validation sample. It was claimed that this represents an optimal cross-validation, but it was chosen as a simple and representative application of the method [51]. The generalization error for the model selected by each method was computed with respect to the problem target function $f(x)$. Having summarized the study done by Kearns et al. [51], we proceed to introduce the KMDL version [51] of MDL (see sections 6.4 and 6.6), a 'corrected' version - CMDL (see section 6.5), and Minimum Message Length (MML) (see section 6.7). We then present empirical results and discuss these.

6.4 The KMDL method

We replicate the MDL encoding scheme employed by Kearns et al. in their study. To distinguish it from a modified method we consider later (see section

6.5), we term it KMDL. According to Kearns et al., MDL [85] is a broad class of algorithms with a common information theoretic motivation and each MDL algorithm is determined by a specific encoding scheme for both functions and their training errors. Kearns et al. [51] present one such encoding scheme for the binary sequence problem, although this is far from an optimal code.

Let h be a function with exactly k cut points. Description of $h(\cdot)$ first requires specification of its number k of cutpoints. The length of this description is neglected in KMDL. Given k, we can sufficiently describe $h(\cdot)$ by specifying the k sample points immediately before which $h(\cdot)$ changes value. Note that it makes sense for $h(\cdot)$ to have a cutpoint before the first sample point x_1 (because as in section 6.1, $f(x)$ is defined to be 0 in even-numbered sub-intervals), but not after the last, x_N. Thus, there are N places where $h(\cdot)$ may have cuts. Assuming, as in [51], that the cuts are equally likely to occur in any of these places, specifying their locations takes $\log_2 \binom{N}{k}$ bits. Given $h(\cdot)$, the training samples can simply be encoded by correcting the mistakes implied by $h(\cdot)$. Suppose $h(x)$ differs from $f(x)$ at m sample points, where $m = N \times \hat{\epsilon}(h)$. (KMDL neglects the cost of describing m.) Given m, the identity of the m sample points where y_i differs from $h(x_i)$ can be specified with $\log_2 \binom{N}{m}$ bits. Thus, KMDL arrives at a total description length of

$$\log_2 \left\{ \binom{N}{k} \times \binom{N}{m} \right\} = \log_2 \binom{N}{k} + \log_2 \binom{N}{m} \text{ bits}$$

In [51], expressions such as $\log_2 \binom{N}{m}$ are approximated by $N \times H(m/N)$ where $H(\cdot)$ is the binary entropy function from section 6.2. Dividing by N leads to the KMDL choice of k:

$$k = argmin_k \{H(k/N) + H(\hat{\epsilon}(h'_k))\} \tag{6.1}$$

6.5 The CMDL method

We attempt a modest correction to the KMDL method, which we term CMDL. The CMDL encoding scheme includes the lengths needed for the description of k and m. An important point that we note in developing our MDL approach is that, as we are only considering the maximum-likelihood model in each class, there cannot be any misses adjacent to a cutpoint. It then follows that we can safely assume that the number of cuts $k \leq N - 2m$. The training error count m can certainly not exceed $N/2$, so the cost of encoding m is $N/2$ bits. The cost of specifying a cutpoint from $N - 2m$ potential cuts is $\log(N - 2m)$ bits. Although the simplifications of KMDL relative to CMDL appear to be typical of much of the use of MDL in practice, the refinements of CMDL are similar to the complete coding advocated by Rissanen and Dom [88, 31, 118] and are justified (for this problem) by superior performance in practice, as demonstrated by our results. We also replace the binary entropy approximation in KMDL by the accurate log-combinations expression. Thus, CMDL selects

$$k = argmin_k \left\{ \log(N/2) + \log(N - 2m) + \log \binom{N}{m} + \log \binom{N - 2m}{k} \right\} \tag{6.2}$$

6.6 The Flaw in KMDL

While the minor correction of KMDL to CMDL has a clear beneficial effect, there remains a serious flaw in the coding scheme used in both methods. The use of a *Minimum* Description Length principle to select among competing model-based encodings of the data can make sense only if the coding scheme

used with each competing model indeed *minimizes* the length of the description employing that model. The schemes used in KMDL and CMDL do not come close.

These methods specify the location of the cutpoints of a model to within the interval between adjacent sample points. On average, this is a precision of about $(1/N)$. Except for very low noise rates, such precise specification is unwarranted (see section 6.7.2 and below), leads to an over-long description, and vitiates the comparison of competing models.

It is an essential feature of efficient model-based coding (whether MDL or MML) that *no estimated parameter be specified more accurately than it can be estimated.* To illustrate this point, suppose, in this problem, that we decide to encode the cutpoints so that they are always required to precede an odd-numbered sample point. That is, we encode them to an average precision of $(2/N)$. What effect will this have on the description length? First, we save approximately k bits, because for each of the k cutpoints there are now only $N/2$ possible locations to be selected among. Second, for each cutpoint, there is a probability $(1/2)$ that it is where we wanted it to be. If it is not, then one y-value will be encoded using probability p instead of $(1-p)$ or vice-versa. The final result is that the description length is increased on average by $k(\frac{1}{2}\log_2 \frac{1-p}{p} - 1)$ bits. This quantity is *negative* for $p > 0.2$, so by lowering the precision of cutpoint specification we actually *shorten* the description unless the noise rate is less than 0.2. In other words, unless the noise is low, even the arbitrary decision to rule out $N/2$ possible cutpoints will actually *improve* the minimum description length achieved over the faulty method of assuming an *exact* specification of cutpoints. The MML method now described generalizes this point. (For a detailed recent comparison between MML and MDL, and their relation to the works of Solomonoff [98],

Kolmogorov [53] and Chaitin [13] on algorithmic probability and algorithmic information theory, see Wallace and Dowe [125] and other articles in that special issue of the *Computer Journal*.)

6.7 MML Based Model Selection

In essence, MML [122, 127, 125] seeks to minimize a message length defined by the joint encoding of the model and the data given the model. The MML principle aims to find a specific model that explains the data — as opposed to MDL-based methods that prefer to infer a model class. It will be more convenient now to measure lengths in nits rather than bits (1 nit = $\log_2 e$ bits), so we now switch to natural logs. (Henceforth, we use k and p to denote the estimated model quantities.)

The MML principle [127, 125] offers the following general expression for computing the MML message length for parameter vector $\vec{\theta}$ and data $\vec{x}$:

$$MessLen = -\log g(\vec{\theta}) - \log f(\vec{x}|\vec{\theta}) + 0.5 \log F(\vec{\theta}) - \frac{D}{2}\log 12 + \frac{D}{2} \qquad (6.3)$$

where $g(\vec{\theta})$ is the prior density on $\vec{\theta}$, $f(x|\vec{\theta})$ is the likelihood of data $\vec{x}$ given $\vec{\theta}$, D is the number of parameters and $F(\vec{\theta})$ is the "Fisher Information". (Slightly better approximations exist [125], but the above should suffice for this problem.)

6.7.1 Encoding the Estimated Probability

In our encoding scheme we include a statement of p, the estimated noise rate. This value determines that y-values agreeing with $h(x)$ will be encoded with length $-\log(1-p)$ each, and those disagreeing, with length $-\log p$.

In this application of the MML principle, the parameter p is one element of $\vec{\theta}$ and it is the parameter p of a binomial sequence with N trials and m disagreements, so the relevant likelihood function is

$$f(m|p) = p^m(1-p)^{N-m} \tag{6.4}$$

Since we assume $0 \leq p \leq 0.5$ but have no other prior information, we assume a uniform prior $g(p)$.

The Fisher information $F(p)$ is easily shown [127, 126] to be

$$F(p) = \frac{N}{p(1-p)} \tag{6.5}$$

The value of p that minimizes the message length can be derived [127, 126, 122] by differentiating the expression for *MessLen* to obtain,

$$p = (m + 0.5)/(N + 1.0) \tag{6.6}$$

6.7.2 Encoding the Cutpoints

In specifying the encoding for the number of estimated cuts k we note an earlier observation that, since we are only considering the maximum-likelihood model in each class, there cannot be any misses adjacent to a cutpoint. Therefore, we can safely assume that the number of cuts $k \leq N - 2m$. This results in a codelength of $\log(N - 2m)$ nits. The training error count m certainly cannot exceed $N/2$, so the cost of encoding m is $N/2$ nits.

As the cutpoints are real-valued, it would require an infinite number of bits to specify them precisely. Thus, we need to find an optimal precision for these parameters — recall section 6.6. Let δ be the precision (range) with which we want to specify a cutpoint. We assume that the true cutpoint is uniformly distributed within this range. Let ϵ be the difference between the

true cutpoint and the estimated one. Since we assume that our cutpoint is uniformly distributed in δ, the difference ϵ is uniform in $[-\frac{\delta}{2}, +\frac{\delta}{2}]$.

The expected number of data points in a region of width δ is given by $N\delta$. Since our training values are uniformly distributed, the expected number of data points coded with the wrong probability (put on the "wrong" side of the cutpoint) is given by $N \cdot E(|\epsilon|)$. The expected value of $|\epsilon|$ can be derived as

$$\begin{aligned} E(|\epsilon|) &= \frac{1}{\delta}\int_{-\delta/2}^{\delta/2} |x|\,dx = \frac{2}{\delta}\int_{0}^{\delta/2} x\,dx = \frac{2}{\delta}\left[\frac{1}{2}x^2\right]_0^{\delta/2} = \frac{2}{\delta}\times\frac{\delta^2}{8} \\ &= \frac{\delta}{4} \end{aligned} \tag{6.7}$$

The expected excess cost (in message length) of encoding a single data item with the wrong probability can be calculated by computing the difference between the expected cost of encoding the data item with the correct and incorrect probabilities. Thus, from section 6.6, the expected excess cost per wrong item C_n can be derived as follows, noting that the code words of length $-\log p$ and $-\log(1-p)$ are interchanged for each wrong item.

$$\begin{aligned} C_n &= (-p\log(1-p) - (1-p)\log p) - (-p\log p - (1-p)\log(1-p)) \\ &= p(\log(p) - \log(1-p)) + (1-p)(\log(1-p) - \log(p)) \\ &= p\log\frac{p}{1-p} + (1-p)\log\frac{1-p}{p} \\ &= (2p-1)\log\frac{p}{1-p} \\ &= (2(1-p)-1)\log\frac{1-p}{p} \end{aligned} \tag{6.8}$$

which is, as we would expect, symmetrical between p and (1 - p).

From (6.7) and (6.8), the total increase in the cost C_t of encoding the $N\delta/4$ y-values expected to be encoded with the "wrong" probability is

$$C_t = \frac{N\delta}{4}C_n = \frac{N\delta}{4}(2p-1)\log\frac{p}{1-p} \tag{6.9}$$

The cutpoints of $h(\cdot)$ are parameters of the model and so must be encoded. To encode the position of a cut to precision δ within the range (0,1) requires length $-\log\delta$. Hence, from (6.9), for each of the k cutpoints, the total excess cost C_k incurred by encoding its position to precision δ is

$$C_k = -\log\delta + C_t = -\log\delta - \frac{N\delta}{4}(2p-1)\log\frac{p}{1-p} \tag{6.10}$$

Differentiating C_k from (6.10) and setting the result to zero, we find the value of δ that minimises C_k and hence the message length as follows:

$$-\frac{1}{\delta} + \frac{N}{4}(2p-1)\log\frac{p}{1-p} = 0$$

Hence,

$$\delta = \frac{4}{N(2p-1)\log\frac{p}{1-p}} \tag{6.11}$$

With this choice of δ, from (6.9) and (6.11), the expected excess cost is $C_t = 1$, just one nit per cutpoint. Given m, from (6.10) the total cost of encoding all k cutpoints to precision δ is

$$\log(N-2m) + k(1-\log\delta) - \log(k!)$$

since $k \leq (N-2m)$ and the order in which the k cutpoints are specified is immaterial. As mentioned earlier in section 6.6, the use of an overly precise specification for δ is only justified when the noise rate p is low. How low p must be to justify Kearns et al's exact specification we can find by substituting $1/N$ for δ in expression (6.11) above, obtaining $p \approx 0.018$. In other words, the precision specified by Kearns et al. [51] is only justified when the noise parameter is an *extremely* low value, less than 2%!

6.7.3 Encoding the Data

Given m, the identity of the m sample points where y_i differs from $h(x_i)$ can be specified with code length $\log \binom{N}{m}$, which is included in the data part of our message.

6.7.4 The total message

The length of the entire MML message can now be computed. The components of the message are

- the statement of k and m,
- the statement of p to precision $\sqrt{12/F(p)}$ within the range (0,1/2),
- the positions of the cutpoints to precision δ, and finally,
- the encoding of the data.

In estimating the noise parameter p, the number of mistakes m made by the maximum-likelihood model $h'_k(x)$ is increased by the expected additional number of disagreements resulting from the imprecision of cutpoints. The resulting estimated error count is used in place of m in estimating p, which affects the choice of δ (from equation (6.11)), and hence in turn the estimated error count (from equation (6.7)). A few iterations of these calculations converge quickly. As a result, our estimate of the noise rate exceeds m/N. The effect seems to correct for the overfitting of the maximum-likelihood model which, in KMDL and CMDL, leads to an underestimate of p. Figures 6.11 and 6.12 plot the noise rate or probability as estimated by the different methods from the data when the cutpoints are evenly-spaced and random respectively.

6.8 Experimental Protocol

A dynamic programming algorithm was used to find the maximum-likelihood model for each value of the k classes of cutpoints. The estimated k and training error count m for each class was then supplied to the MML, KMDL and CMDL methods and the cutpoint model preferred by each method was noted. All four methods — MML, KMDL, CMDL and CV — were compared on noise rates of $p = 0, 0.1, 0.2, 0.3, 0.4$ with sample sizes ranging from $N = 100$ to 3000 in steps of 100. For each choice of p and N, 100 replications were performed. Our true cutpoint models consisted of 100 either evenly-spaced or randomly generated cutpoints.

All methods were compared on the basis of

- the number of estimated cuts,
- the Kullback-Leibler (KL) distance between the true and estimated model, and,
- the expected prediction error (EPE).

The Kullback-Leibler distance (also known as the "relative entropy") measures the expected excess cost of using an encoding based on the estimated model rather than the true model. Formally, the Kullback-Leibler distance between the true distribution $p(y)$ and the estimated distribution $q(y)$ is:

$$KL\ \ (p \parallel q) = \sum_{y} p(y) \log \frac{p(y)}{q(y)} \tag{6.12}$$

6.9 Analysis of Results

Although the original cutpoint problem analysis by Kearns et al. [51] from section 6.3 employs evenly-spaced cutpoints, it is important to remember that

the original statement of the problem [51] and all the models implicitly assume that the cutpoints are randomly distributed. The problem of learning evenly-spaced cutpoint models is relatively easy since there is then no need at all to specify the location of the cutpoints. Since in fact none of the learning models were optimized for that easier problem, it was entirely inappropriate for Kearns et al. to employ it in their empirical analysis. Therefore, we address the original problem of inferring models of data generated from randomly generated cutpoints.

6.9.1 Randomly-spaced Cutpoints

Figures 6.2-6.5 include comparisons of the KL distance and the number of estimated cuts for noise rates of 0, 0.1, 0.2, 0.3 and 0.4 respectively. Each figure plots the generalization error as measured by the KL distance and the estimated number of cuts collated. All plots represent averages over 100 replications. With *no* noise (i.e., $p = 0$), all methods understandably performed well.

The robust performance of MML for small and medium sample sizes in the presence of increasing noise values can be observed plainly in Figures 6.2-6.5. There is clear statistical significance in the superior performance of MML in comparison to the KMDL and CMDL methods (e.g., with sample sizes from 200 to 1000 computed p-values (from classical statistical significance testing) for KL distance were always less than 0.05 and usually less than 0.0001). The superiority of MML to CV in the KL-distance metric, despite not generally being statistically significant at individual points (of noise level and sample size), is consistent and enduring across noise levels and most sample sizes examined. Thus, in general, it is clear that the MML approach performs better than all the other methods evaluated when the cutpoints are randomly

distributed (as assumed in the problem framework).

6.9.2 Estimating the Number of Cutpoints

Despite this evident superiority in the KL-distance metric, it can be observed that MML tends to be slightly conservative in estimating the number of cuts in comparison with CV. This conservatism of MML is due in large part to the fact that the MML technique we have employed here is directed at finding the simplest model for explaining the data at hand and is not specifically directed at estimating the number of cutpoints. In particular, our MML program ignores the possibility that two or more cutpoints may fall between adjacent data values, since any even number of cuts between two adjacent data points will show no discernable cut, and any odd number $(1, 3, 5, 7, \ldots)$ of such cuts will show as only one cut. If we really wanted to estimate the number of cutpoints, then — noting that we can add any arbitrary even number of cutpoints to the MML model without any discernable difference — we would use a Bayesian integration along the lines of those in linear Gaussian segmentation work [74, 40]. (See also section 6.9.3 and discussion comments in section 6.10.)

6.9.3 Expected Prediction Error

Figures 6.14 and 6.15 present comparisons of all methods based on the expected prediction error (EPE) [1] for random cutpoint models. The expected prediction error measures the "right"/"wrong" predictive accuracy of the different methods. It is important to note that the MML method does notably better at minimizing the KL-distance than expected prediction error in comparison with the other methods. This implies that even if the other methods

[1] A similar measure, called the 'generalization error', was examined by Kearns et al.[51].

are sometimes more accurate in estimating the number of cuts, the other methods must then certainly be putting the cuts in the wrong places. The conservatism on small data-sets of the MML method in estimating cutpoints is warranted by the above results in the sense that the MML method is trying to get closer to the true model (including the *location* of cutpoints *as well as* the noise rate, p) as opposed to finding incremental improvements in prediction error. Ironically enough, this holds even though (as in sections 6.6 and 6.7.2) MML encodes cutpoints more cheaply than KMDL. Figures 6.12 and 6.13 compare the different methods in their ability to estimate the noise rate p with random and evenly-spaced cutpoints. The MML and CV methods seem to converge faster to the true noise rate.

6.9.4 Evenly-spaced Cutpoints

For completeness, Figures 6.7-6.11 present comparisons of the KL distance and estimated cutpoints when the true cutpoint model is evenly-spaced, although, as discussed, the original problem specification was for randomly-spaced cuts. Our results replicate the performances of KMDL and CV as discussed by Kearns et al. [51].

6.9.5 Tabulated Results

This section includes some results from the experimental runs. The following tables present comparisons of the different methods for varied sample sizes with a noise rate of $p = 0.2$. All the tables include for each method the number of cutpoints estimated *Ek*, the training error count *Em*, the estimated probability *Ep* and the Kullback-Leibler distance *KLD* — all averaged over 100 runs. The true cutpoint model consists of 100 randomly-spaced cutpoints.

Method	Ek	Em	Ep	KLD
KMDL	72.67	10.23	0.056	1.3800
CMDL	43.11	40.74	0.207	0.9068
MML	1.39	90.44	0.453	0.1903
CV	21.60	48.54	0.245	0.3125

Table 6.1: $N = 200$ and $p = 0.2$

Method	Ek	Em	Ep	KLD
KMDL	219.05	0.00	0.002	1.7186
CMDL	193.06	17.73	0.031	1.5182
MML	26.55	150.30	0.251	0.1505
CV	65.92	104.65	0.175	0.2586

Table 6.2: $N = 600$ and $p = 0.2$

6.10 Discussion

The results shown here demonstrate that the poor behaviour of the "MDL" method reported by Kearns et al. (KMDL) [51] was not inherent in the MDL principle or the MML principle. Rather, it was caused by a failure properly to consider the minimization of the description lengths for each model, and in particular the need to encode estimates (here, the cut positions) to an appropriate precision.

The MML method developed here (which may equally well be considered an MDL method) improves dramatically upon the performance reported by Kearns et al., in general outperforming all of the other methods in the true

Figure 6.2: Evaluation of Different Methods with Random Cutpoints

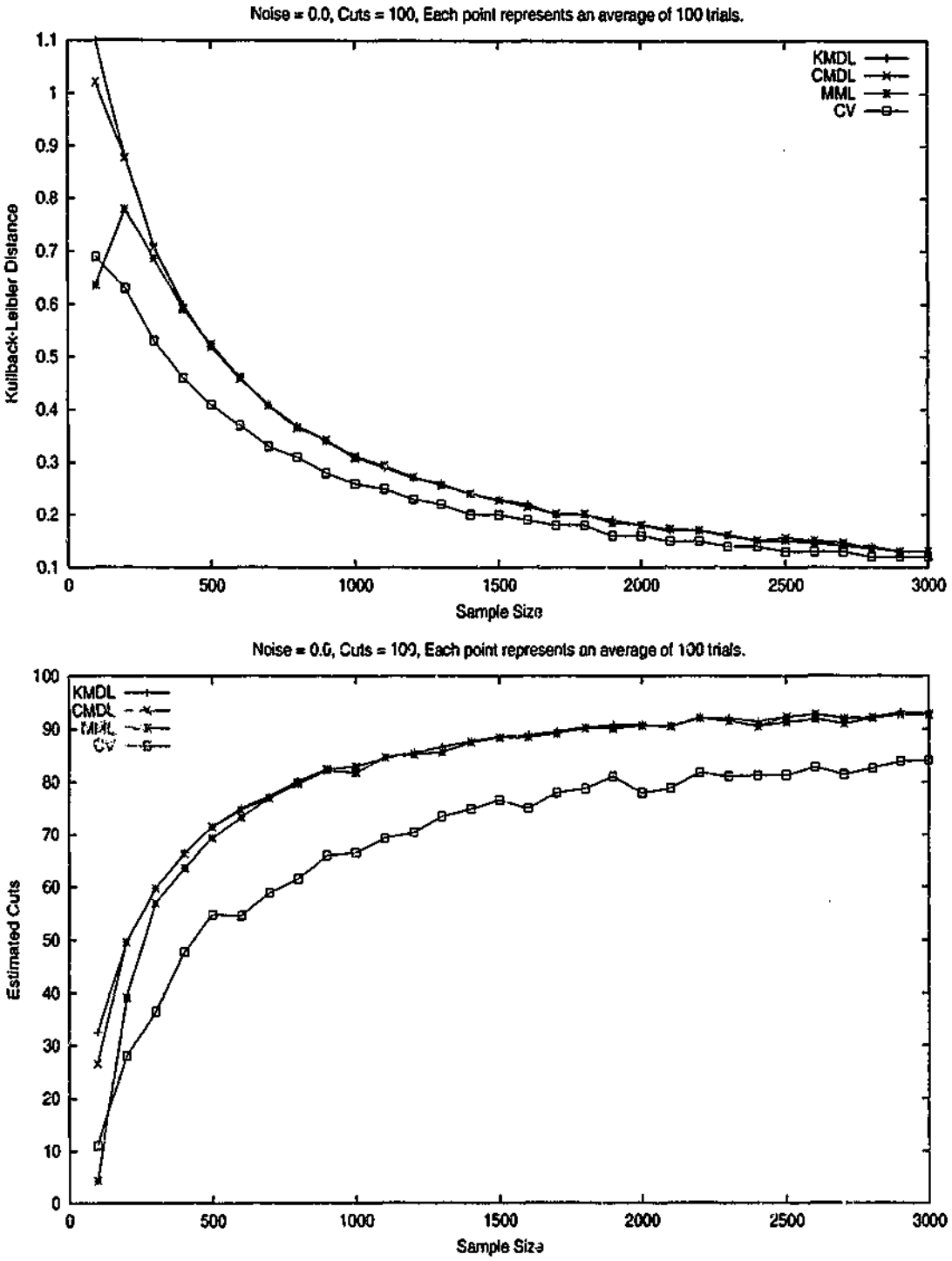
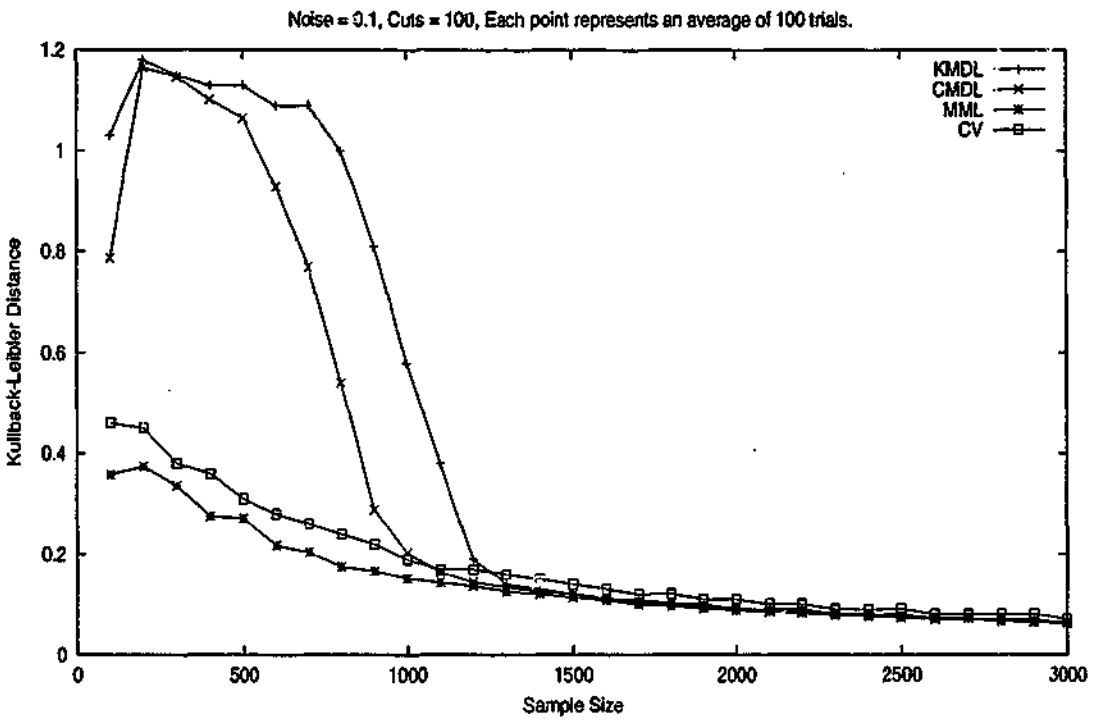


Figure 6.3: Evaluation of Different Methods with Random Cutpoints



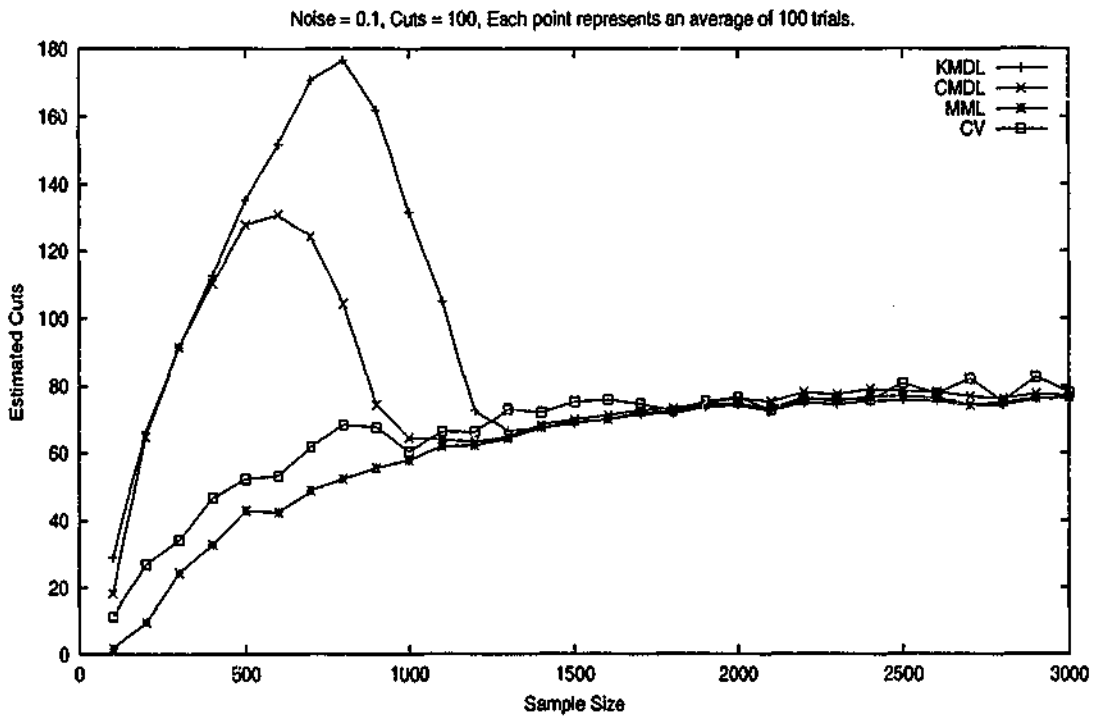


Figure 6.4: Evaluation of Different Methods with Random Cutpoints

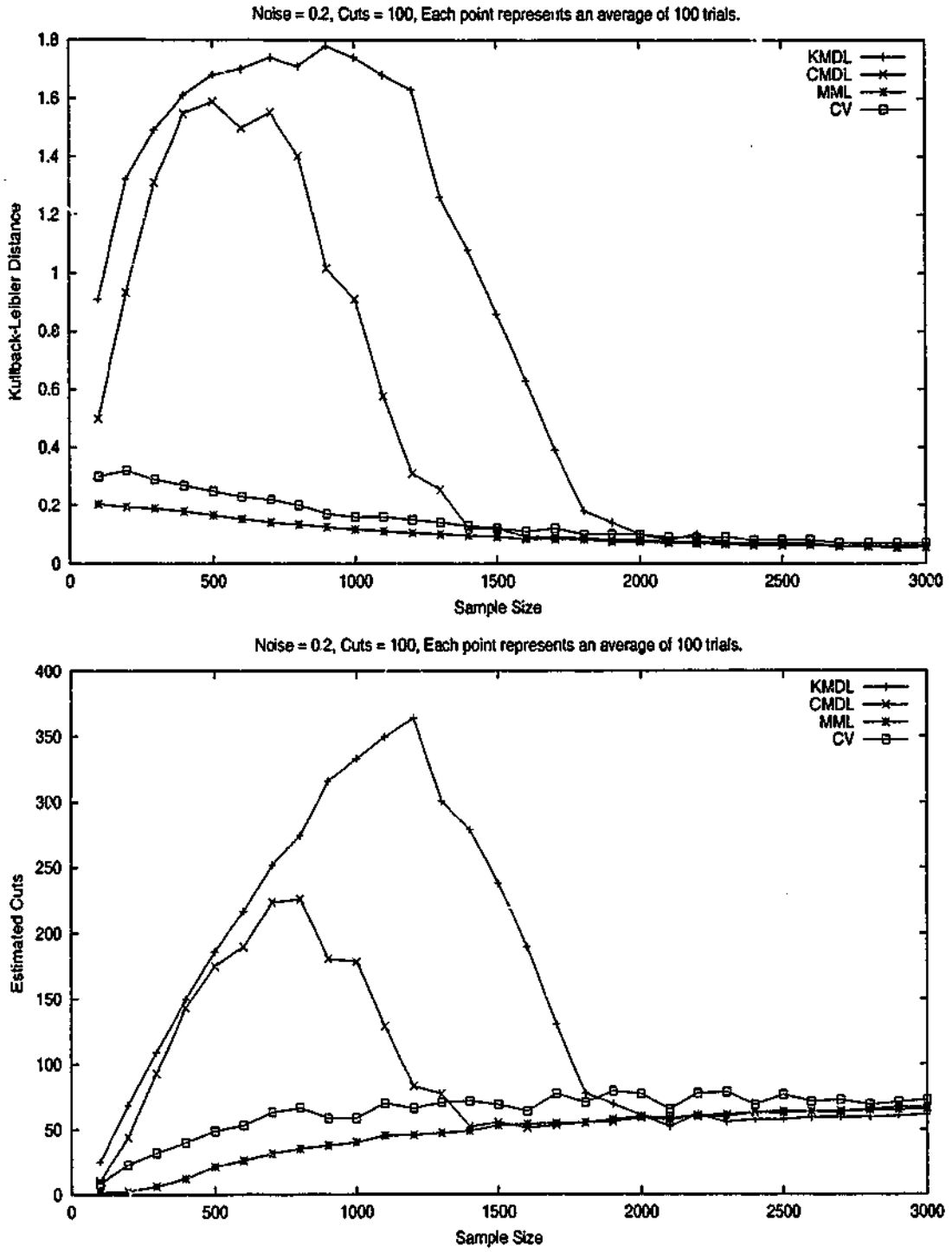
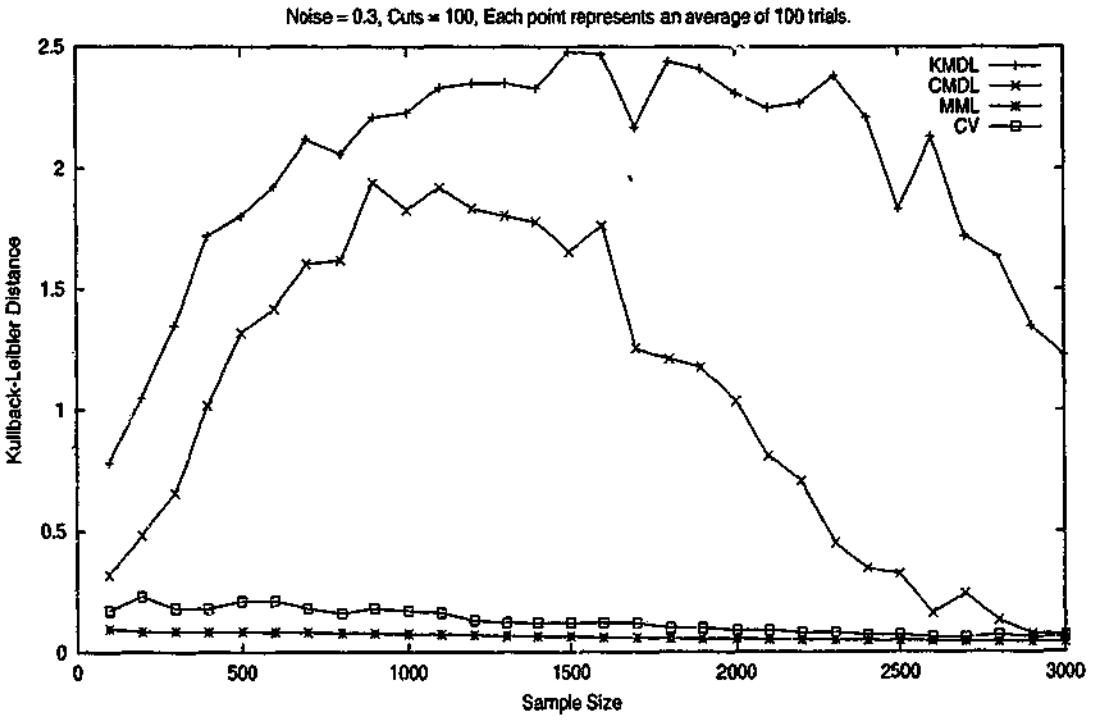


Figure 6.5: Evaluation of Different Methods with Random Cutpoints



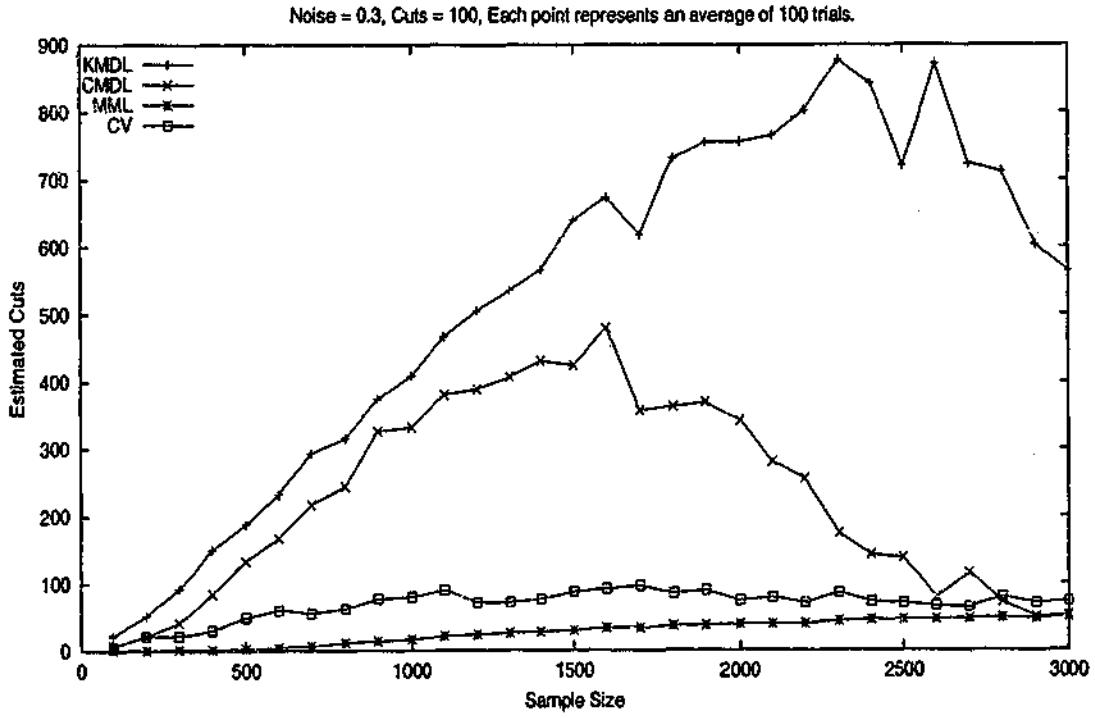


Figure 6.6: Evaluation of Different Methods with Random Cutpoints

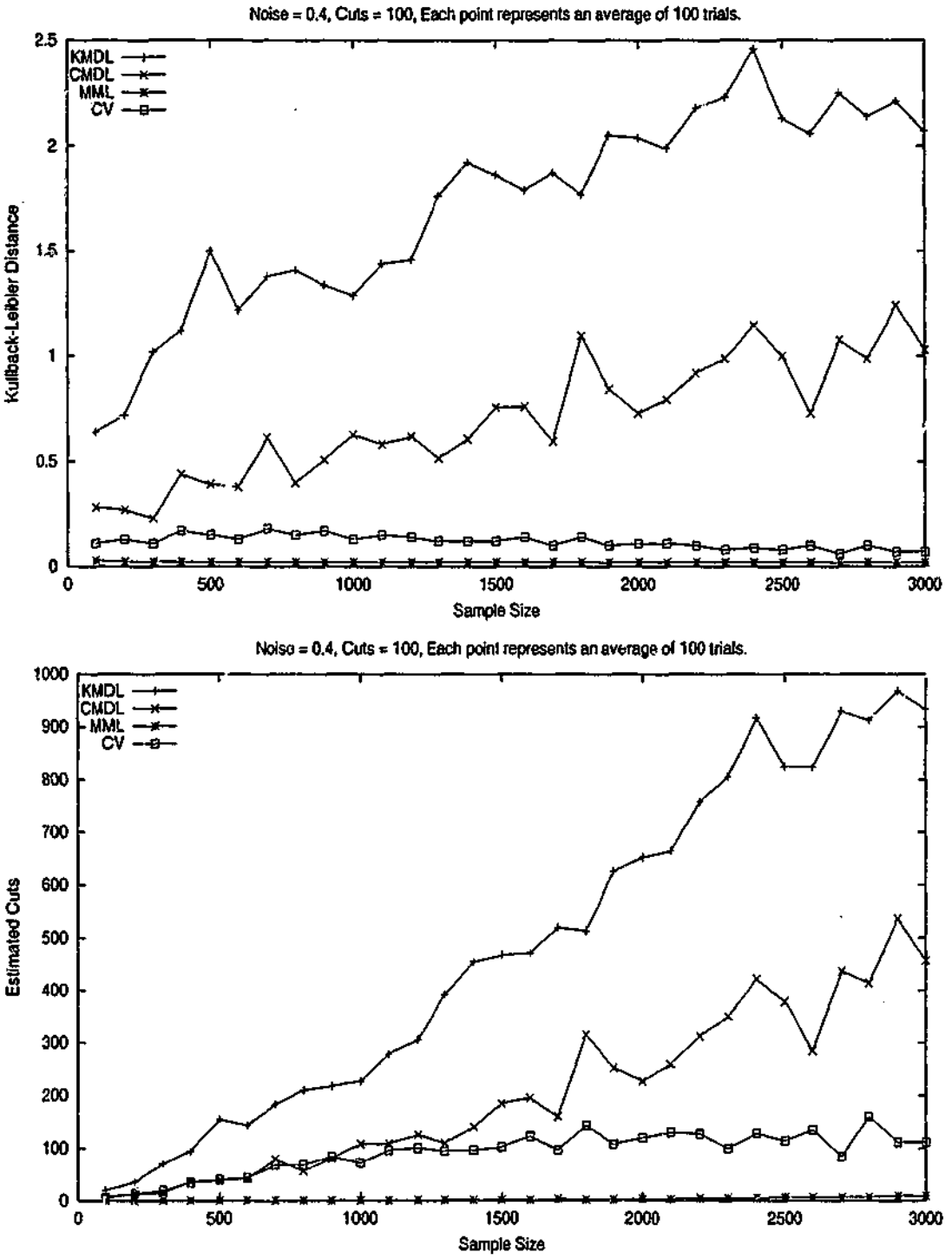


Figure 6.7: Evaluation of Different Methods with Evenly-spaced Cutpoints

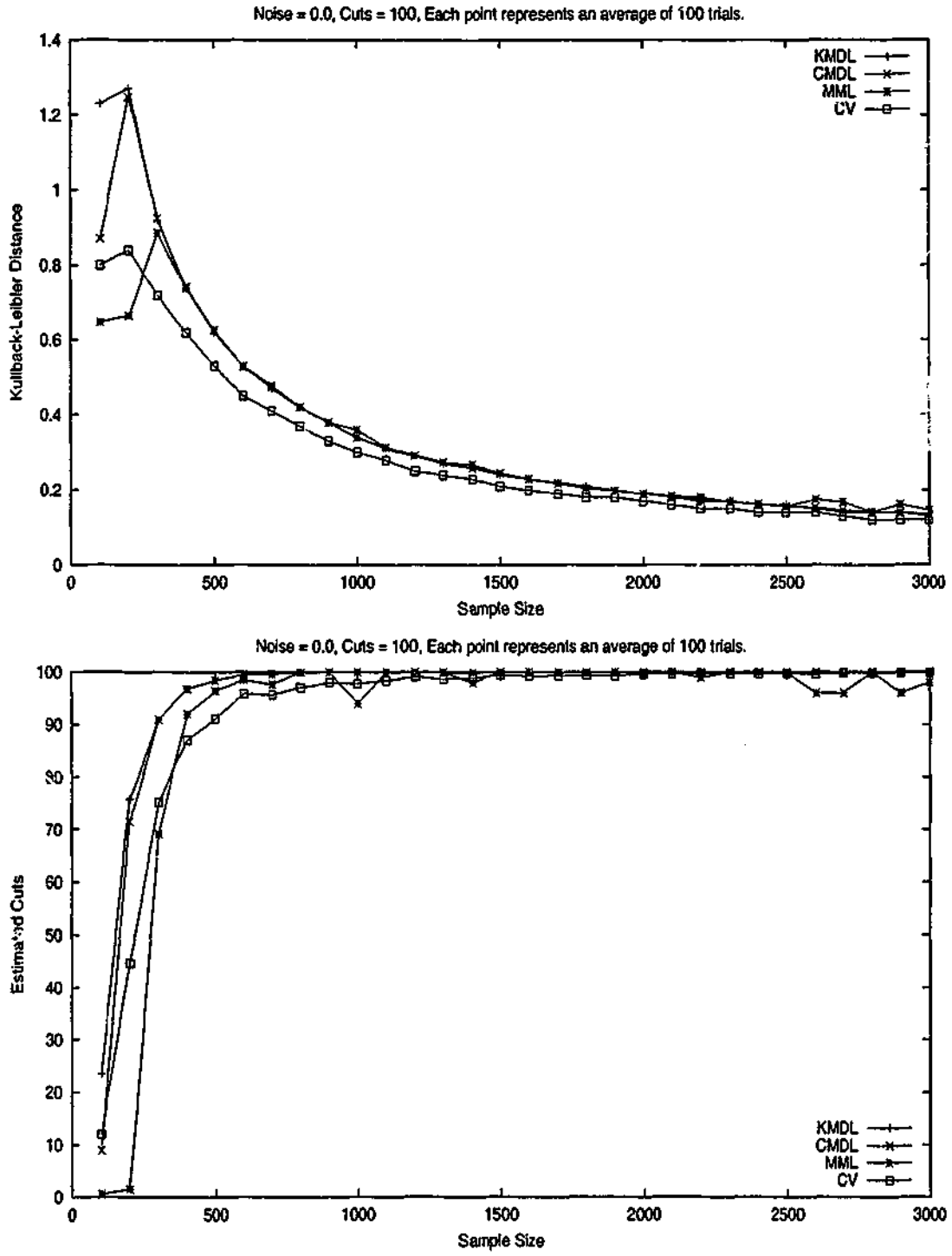


Figure 6.8: Evaluation of Different Methods with Evenly-spaced Cutpoints

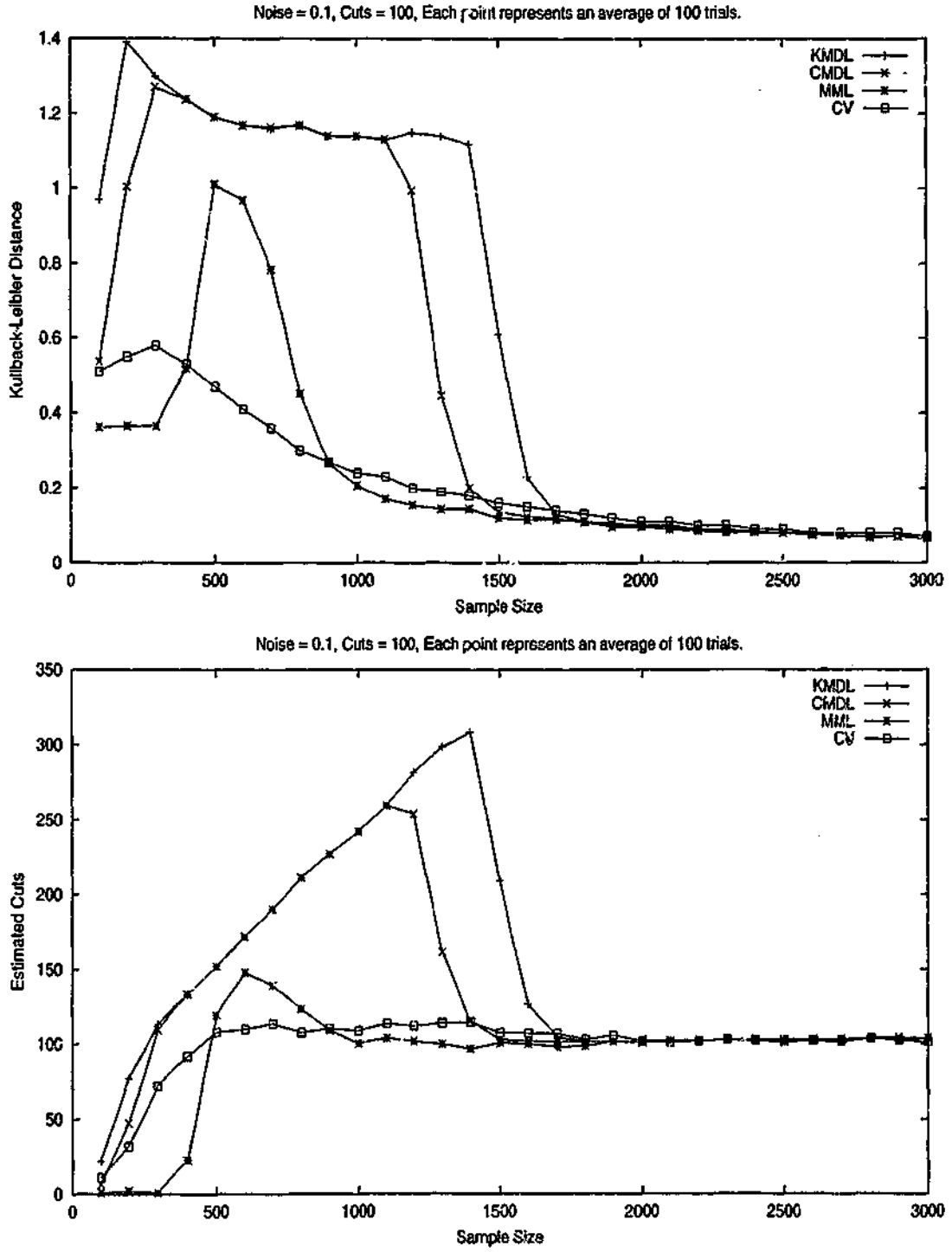
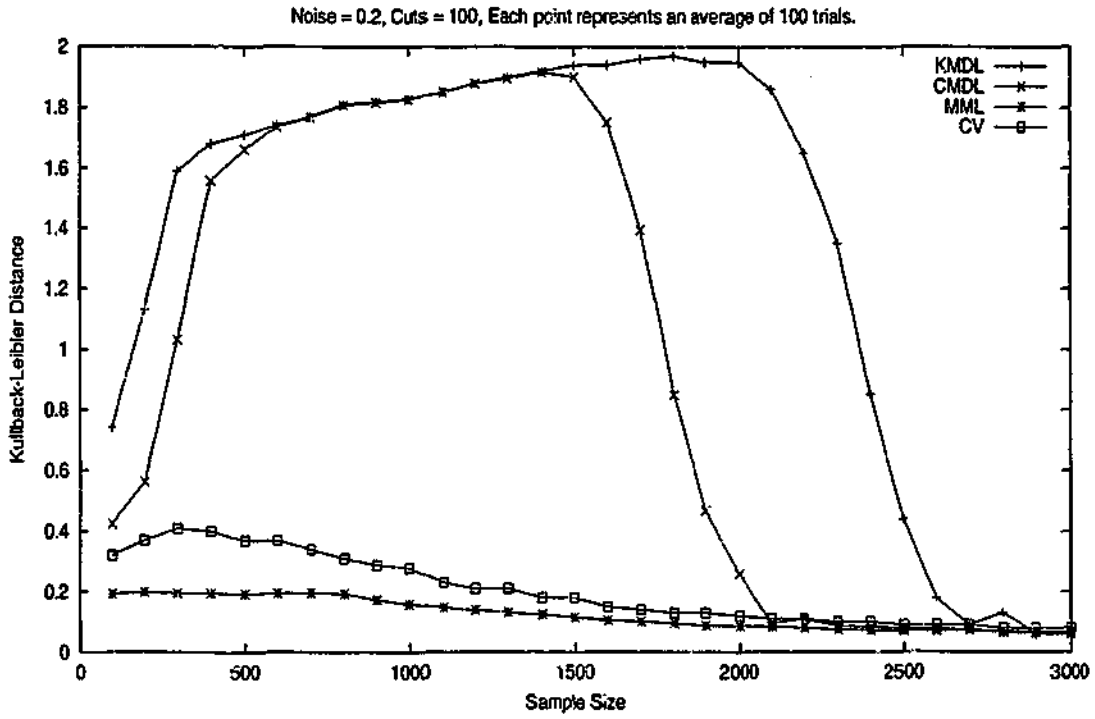
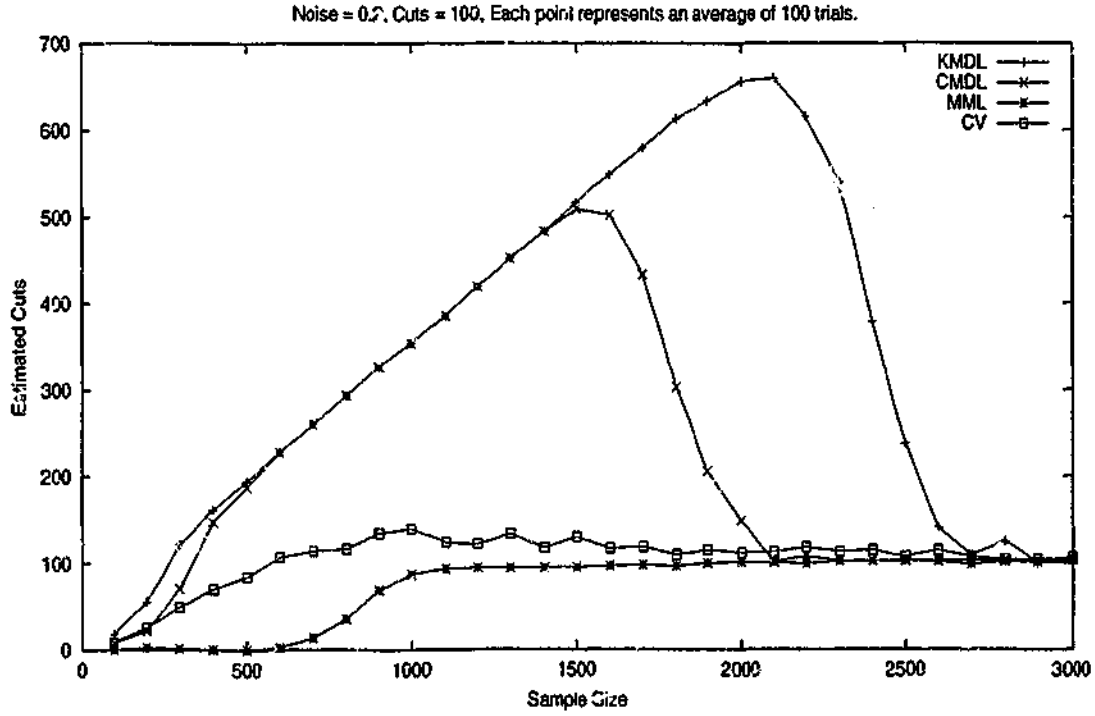


Figure 6.9: Evaluation of Different Methods with Evenly-spaced Cutpoints





Method	Ek	Em	Ep	KLD
KMDL	324.94	14.81	0.016	1.7011
CMDL	143.28	133.68	0.134	0.7230
MML	40.21	223.97	0.225	0.1161
CV	70.24	181.11	0.182	0.1706

Table 6.3: $N = 1000$ and $p = 0.2$

Method	Ek	Em	Ep	KLD
KMDL	225.72	186.20	0.125	0.8160
CMDL	49.67	294.48	0.197	0.1000
MML	52.39	319.90	0.214	0.0911
CV	68.34	287.17	0.192	0.1207

Table 6.4: $N = 1500$ and $p = 0.2$

problem environment. Despite that, this implementation of MML is itself only a rough application of the principle, and still uses a sub-optimal coding scheme. In particular, it uses a constant cutpoint precision δ for all cutpoints of the model, whereas our derivation of δ clearly implies that the precision used for each cutpoint should reflect the local density of sample points near the cut. A more carefully optimized MML method making proper use of knowledge of the sample point locations would improve results, especially perhaps in small samples. Previous experience with MML leads us to expect that such a development would lessen the over caution of the present method in finding cuts in small samples.

Figure 6.10: Evaluation of Different Methods with Evenly-spaced Cutpoints

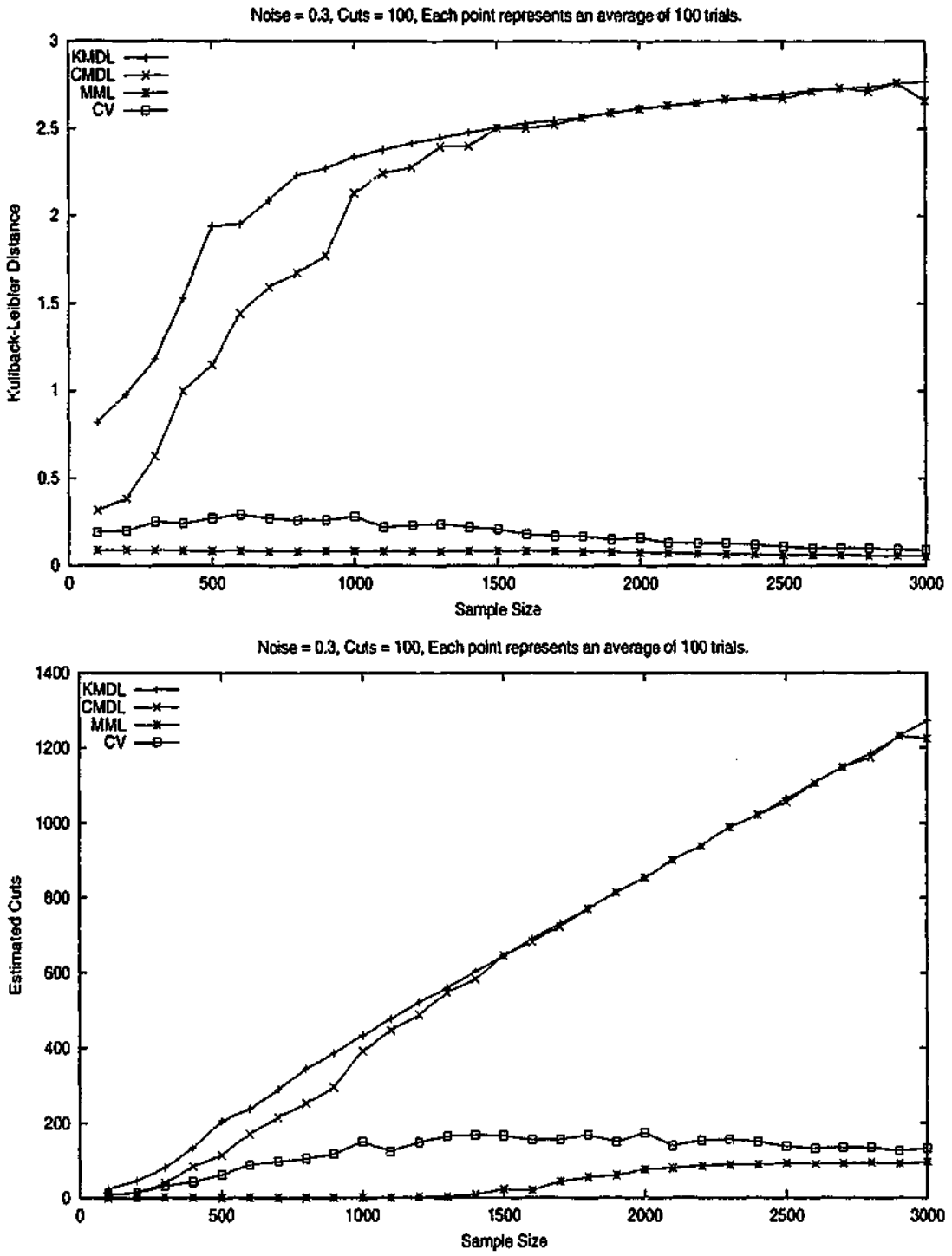
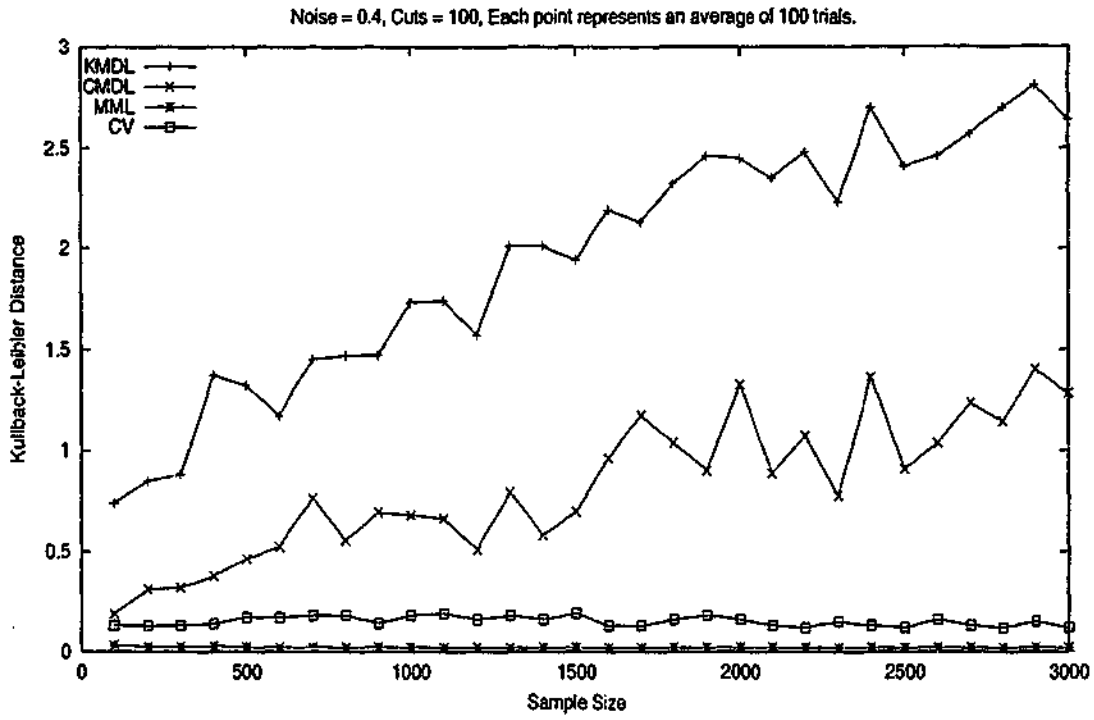


Figure 6.11: Evaluation of Different Methods with Evenly-spaced Cutpoints



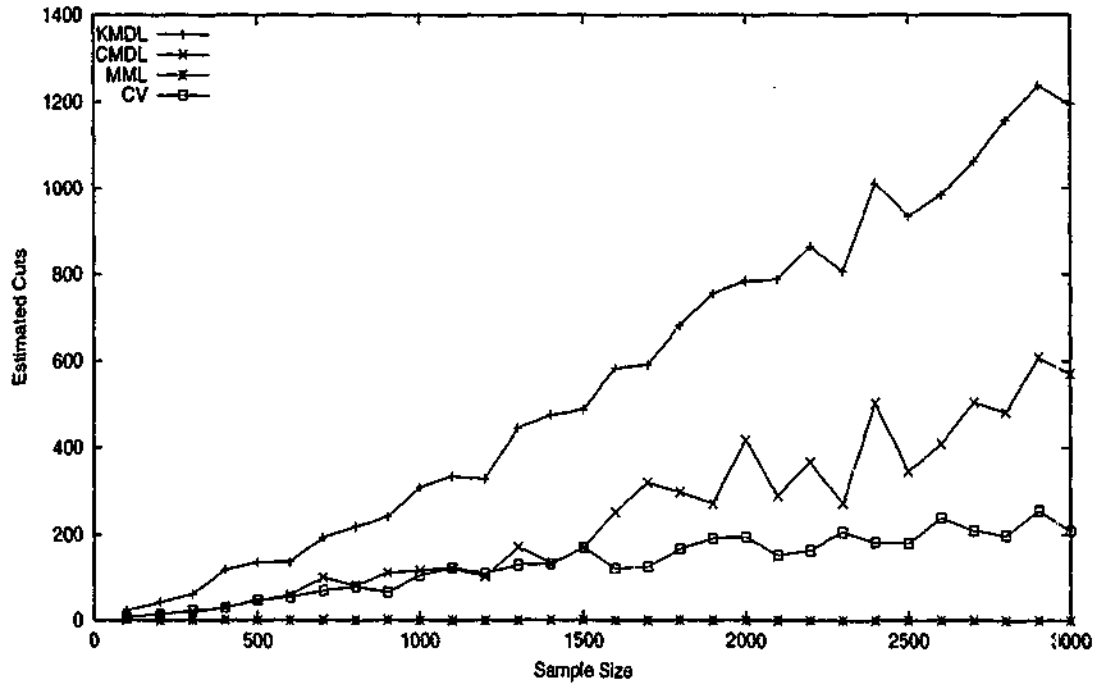


Figure 6.12: Comparison of Estimated p with Random Cutpoints

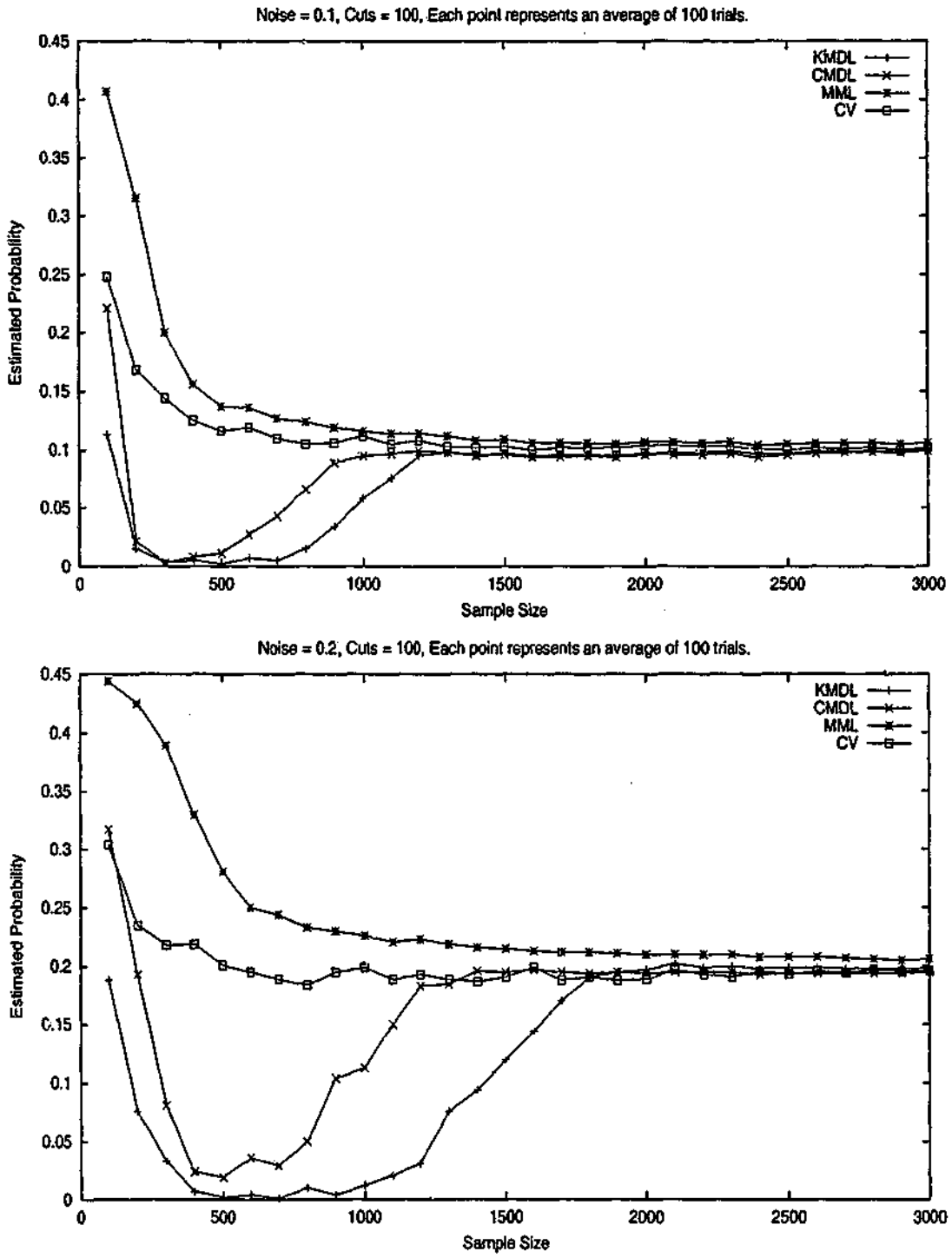


Figure 6.13: Comparison of Estimated p with Evenly-spaced Cutpoints

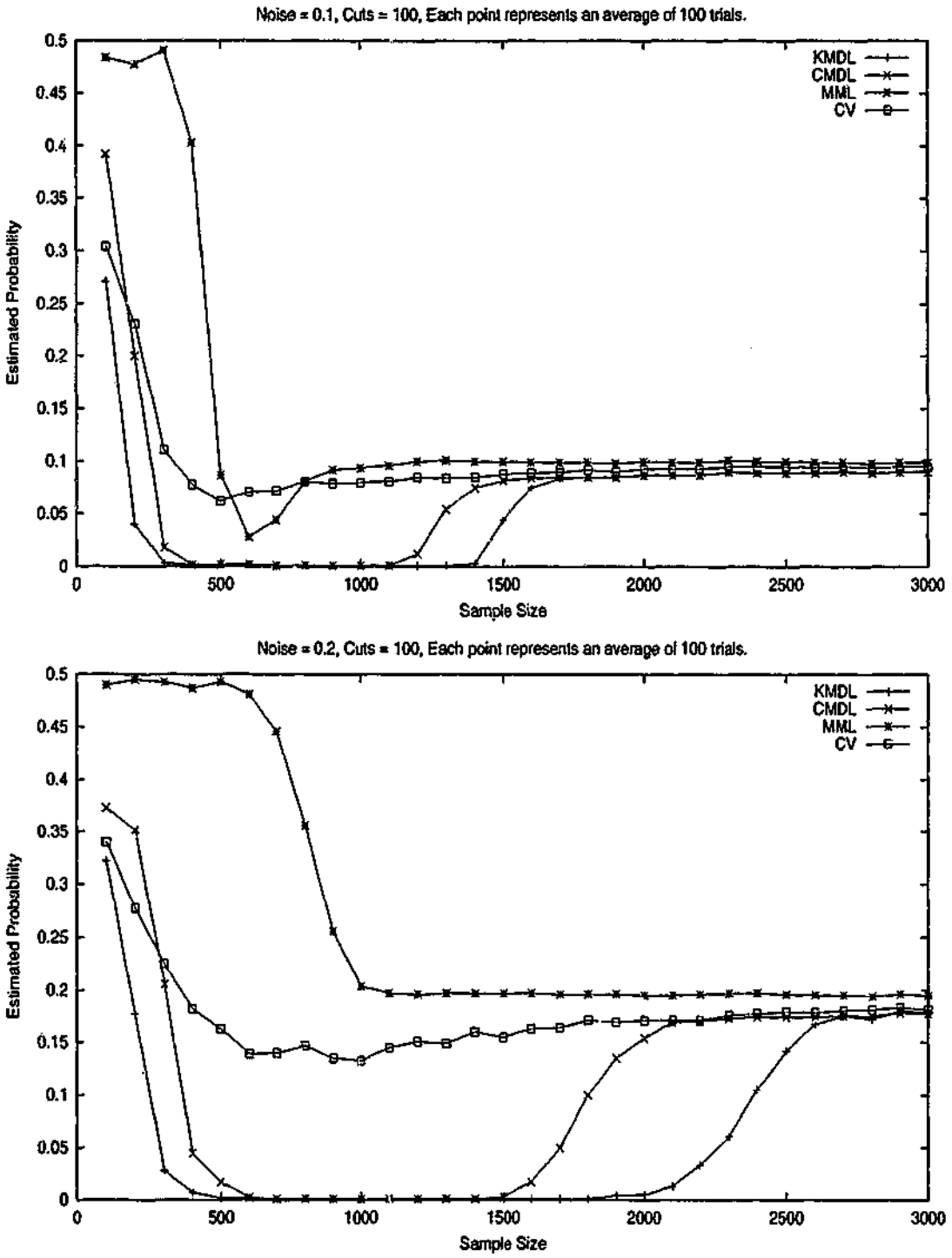


Figure 6.14: Comparison of Methods on EPE with Random Cutpoints

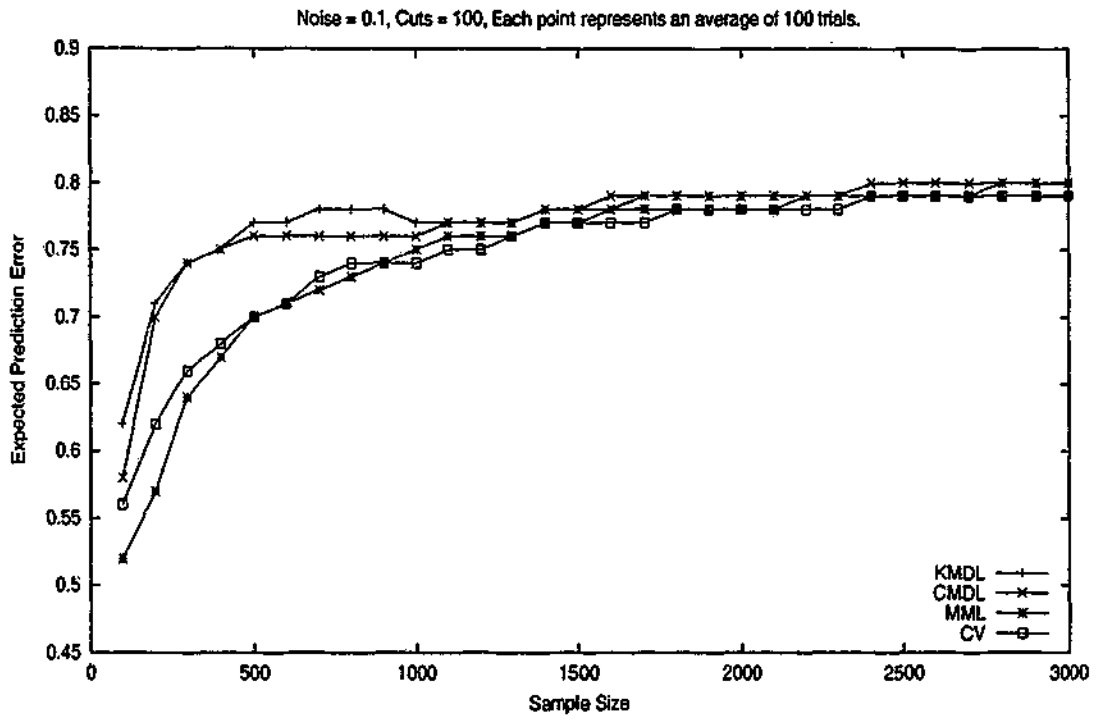
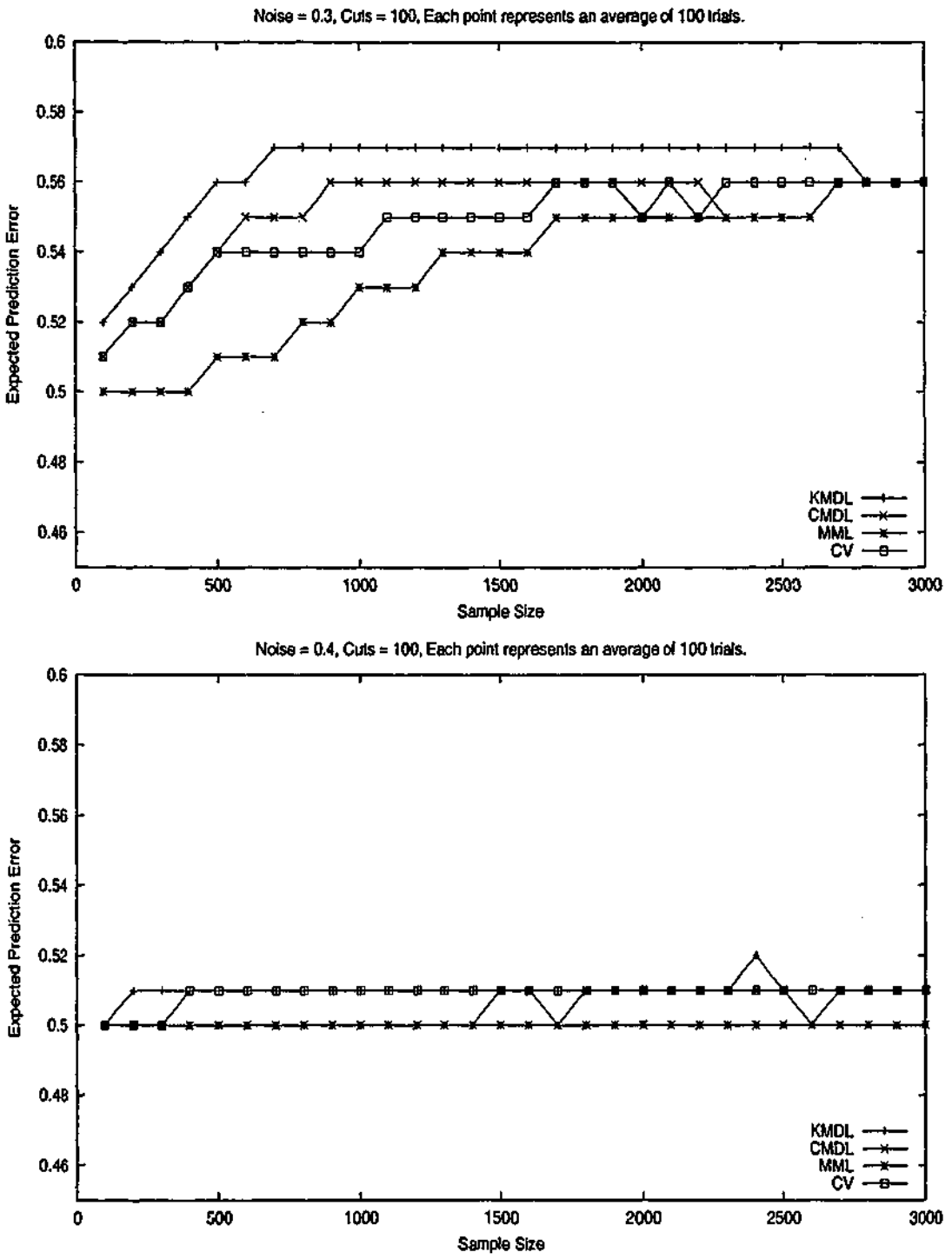


Figure 6.15: Comparison of Methods on EPE with Random Cutpoints



Chapter 7

Conclusions

The primary aim of this research endeavour is to present a methodology for model estimation and inference. The methodology commonly known as the Minimum Message Length is used to develop systems for learning from noisy data. The performance of these systems in the domains of polynomial regression and binary sequence segmentation is demonstrated through a comprehensive empirical study with other well-known approaches. In this process the research work also demonstrates the effectiveness of the Minimum Message Length principle as a principled metric for model preference. Based on our empirical evaluation we find that the MML approach in general provides more robust and accurate model selection than the other model selection metrics like Akaike's information criterion, Swartz's criterion, generalized cross-validation and the Minimum Description Length principle.

Chapters five and six presented novel systems for univariate polynomial discovery and binary segmentation. In addition to these contributions it is also important to consider the significance of the empirical results presented in these chapters. The empirical analysis of the bound on prediction error in chapter five and the demonstration of its looseness is a significant result. In a

similar manner chapter six points out the failure of minimum length encoding schemes that often lead to poor performance.

7.1 BODHI Project

The initial stages of this research were aimed at a system for integrated discovery (named BODHI) from noisy real-world data. The integration was in the context of covering all the stages of the discovery process discussed in chapter 3. While a basic schematic architecture was developed towards this aim the grander objective of building a system for automating the various stages of discovery needs time and future research. Stand-alone systems for univariate polynomial regression and the analysis of binary sequeces were developed as modules that could be integrated into such a larger system. Although component modules (from chapters 5 and 6) demonstrated plausible performance in learning models from noisy datasets the integration of these modules and others remains to be implemented. The Core Data Mining Software (CDMS) package, currently under development at Monash University, aims at the integration of such models together in a discovery environment. There is a substantial amount of scope for further research and development towards the BODHI system.

7.2 Future Research

Structural Risk Minimization and Minimum Message Length have been shown to be plausible inductive principles. The support vector machine (SVM) is a universal constructive learning procedure based on the statistical learning theory of Vapnik [112, 113]. The term "universal" implies that the SVM can be used to learn a variety of representations including polynomial estimators,

radial basis functions, decision trees, neural networks and splines [22, 111]. MML is also universal in this sense [125]. One of our research efforts in progress is to extend the current univariate regression problem presented here to a multivariate design, thus enabling a direct comparison of MML with support vector machines and classical multivariate approaches.

Piecewise polynomials and splines are popular representation schemes which offer a greatly flexible model space. One apparent idea is to extend the MML univariate polynomial model discussed in chapter four to an MML piecewise model for application in non-linear and non-parametric data analysis.

7.3 Remarks

What directions show promise for pragmatic automated discovery ? First, we must further elaborate discovery methods. Systems like Bacon, AM, IDS, Prospector, etc have only considered a handful of techniques. What other knowledge, search strategies, and other means for generating and testing hypotheses could have been used ? We need better ways to represent and reason with the knowledge used to make discoveries. With current methods much useful knowledge is difficult to represent in a computer. For example, how should we represent and use the three-dimensional structure of proteins to determine their behavior ? How do we replicate human visual reasoning ?

Second, we must further demonstrate the scientific feasibility of automated discovery by replicating more discoveries – particularly discoveries of different types. Can we replicate Bohr's formulation of atomic structure, for example, or the design of previously unknown computer algorithms, or Mendel's discovery of genetical laws? These appear to be qualitatively different types of

discovery. Can computers be taught more of the steps involved in scientific method – to observe, classify, quantify, hypothesize, and experiment ? Systems like IDS and BODHI are attempts to develop such integrated programs.

Third, we must seek practical applications of automated discovery. Prospector has indicated the potential. How can the techniques found so far be applied to other problems for economic payback? If we can explain techniques and problems in automated discovery in an explicit fashion, others may find applications and solutions.

Fourth, we must substantially improve our methods for knowledge acquisition and knowledge-base management. The cost of building knowledge bases greatly impedes automated discovery program development. The data warehousing technologies available today are a positive step in this regard. Making automated discovery economically feasible is difficult when knowledge bases require so much effort to build. This is particularly true of discovery knowledge bases, which ideally should contain much expert knowledge in the scientific subject area. Research leading to better knowledge acquisition and knowledge management can contribute substantially to bringing about automated discovery.

Finally, we must explain that discovery results from a set of principles that have been at least partially articulated – that discovery can result from a planned sequence of steps – to dispel the myth that it is a magical, incomprehensible event.

Bibliography

[1] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, January 1991.

[2] H. Akaike. Fitting autoregressive models for prediction. *Annals of the Institute of Statistical Mathematics*, 21:243–247, 1969.

[3] H. Akaike. Statistical predictor information. *Annals of the Institute of Statistical Mathematics*, 22:203–217, 1970.

[4] R. Baxter. *Minimum Message Length Inductive Inference - Theory and Applications.* PhD thesis, Dept. Computer Science, Monash University, Australia 3168, 1996.

[5] R.A. Baxter and D.L. Dowe. Model selection in linear regression using the MML criterion. In J.A. Storer and M. Cohn, editors, *Proc. 4'th IEEE Data Compression Conference*, page 498, Snowbird, Utah, March 1994. IEEE Computer Society Press, Los Alamitos, CA. Also TR 276 (1996), Dept. of Computer Science, Monash University, Clayton, Victoria 3168, Australia.

[6] R.A. Baxter and J.J. Oliver. The kindest cut: minimum message length segmentation. In S. Arikawa and A.K. Sharma, editors, *Proc. of the 7th*

Int. Workshop on Algorithmic Learning Theory, pages 83–90. Springer-Verlag Berlin, 1996. Lecture notes in computer science; Vol. 1160: Lecture notes in artificial intelligence.

[7] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Occam's razor. In Jude W. Shavlik and Thomas G. Dietterich, editors, *Readings in Machine Learning*, pages 201–204. Morgan Kaufmann, 1990.

[8] Margaret A. Boden. *The Creative Mind: Myths and Mechanisms*. Basic Books, New York, 1990.

[9] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Statistics/Probability Series. Wadsworth Publishing Company, Belmont, California, U.S.A., 1984.

[10] Carla E. Brodley. Recursive automatic bias selection for classifier construction. *Machine Learning*, 20(1/2):63–94, 1995.

[11] K. Korb C. Wallace and H. Dai. Causal discovery via mml. Technical Report 96/254, Dept. Computer Science, Monash University, Australia 3168, Feb 1996.

[12] R. M. Cameron-Jones. Minimum description length instance-based learning. In *5th Australian Joint Conference on Artificial Intelligence*, pages 369–373, 1992.

[13] G.J. Chaitin. On the length of programs for computing finite sequences. *J.A.C.M.*, 13:547–549, 1966.

[14] Gregory J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13(4):547–569, October 1966.

[15] Gregory J. Chaitin. On the length of programs for computing finite binary sequences: Statistical considerations. *Journal of the ACM*, 16(1):145–159, January 1969.

[16] P. Cheeseman. On finding the most probable model. In J. Shrager and P. Langley, editors, *Computational Models of Scientific Discovery and Theory Formation.*, pages 73–95. Morgan Kauffman, 1990.

[17] P. Cheeseman et al. Autoclass: A bayesian classification system. In *Fifth International Conference of Machine Learning*, 1988.

[18] V. Cherkassky and M. Mulier. *Learning from Data: Concepts, Theory, and Method*, chapter 4, pages 119–127. Wiley and Sons, 1998.

[19] A. Church. On the concept of a random sequence. *Bull. Amer. Math. Soc*, 1940.

[20] Peter Clark and Tim Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261, 1988.

[21] Cleveland and Grosse. Computational methods for local regression. *Statistics and Computing*, 1(1):47–62, 91.

[22] Corinna Cortes and Vladimir Vapnik. Support vector networks. *Machine Learning*, 20:273, 1995.

[23] P. Craven and G. Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerical Math.*, 31:377–403, 1979.

[24] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines (and Other Kernel-Based Learning Methods)*. CUP, 2000.

[25] L. H. Ungar D. E. Schuurmans and D. P. Foster. Characterizing the generalization performance of model selection strategies. In *Proceedings of the Fourtcenth International Conference on Machine Learning*, 1997.

[26] M. J. Pazzani D. H. Fisher and P. Langley. *Concept Formation: Knowledge and Experience in Unsupervised Learning.* Morgan Kaufmann, 1991.

[27] Lindley Darden. Recent work in computational scientific discovery. In *Nineteenth Annual Conference of the Cognitive Science Society*, 1997.

[28] Carle de Boor. *A Practical Guide to Splines.* Number 27 in Applied Mathematical Sciences. Springer-Verlag, New York, 1978.

[29] D.G.T. Denison, B.K. Mallick, and A.F.M. Smith. Automatic Bayesian curve fitting. *J. Roy. Statist. Soc. Series B*, 60:333–350, 1998.

[30] T. Detterich and R. Michalski. A comparative review of selected methods for learning from examples. In *Machine Learning: An Artificial Intelligence Approach*, pages 41–81. Tioga Publ. Co., 1983.

[31] B. Dom. MDL estimation with Small Sample Sizes including an application to the problem of segmenting binary strings using bernoulli models. Technical Report RJ 9997 (89085) 12/15/95, IBM Research Division, Almaden Research Center, 650 Harry Rd, San Jose, CA, 95120-6099, 1995.

[32] D. L. Donoho, I. M. Johnstone, G. Kerkyacharian, and D. Picard. Wavelet shrinkage: Asymptopia? *J. R. Statist. Soc. B.*, 57(2):301–337, 1995.

[33] D.L. Dowe, R.A. Baxter, J.J. Oliver, and C.S. Wallace. Point Estimation using the Kullback-Leibler Loss Function and MML. In *Proc. 2nd Pacific Asian Conference on Knowledge Discovery and Data Mining (PAKDD'98)*, pages 87–95, Melbourne, Australia, April 1998. Springer Verlag.

[34] D.L. Dowe, J.J. Oliver, and C.S. Wallace. MML estimation of the parameters of the spherical Fisher distribution. In A. Sharma et al., editor, *Proc. 7th Conf. Algorithmic Learning Theory (ALT'96), LNAI 1160*, pages 213–227, Sydney, Australia, October 1996.

[35] T. Edgoose and L. Allison. MML Markov classification of sequential data. *Statistics and Computing*, 9:269–278, 1999.

[36] R. L. Eubank. *Spline smoothing and nonparametric regression.* Dekker, New York, 1988.

[37] Brian Falkenhainer and Ryszard S. Michalski. Integrating quantitative and qualitative discovery, the ABACUS system. *Machine Learning, 1986*, 1(4):367–401, 1986.

[38] S. J. Farlow, editor. *Self-organizing Methods in Modelling (Statistics: Textbooks and Monographs, vol.54.* Marcel Dekker Inc., 1984.

[39] L. J. Fitzgibbon, D. L. Dowe, and L. Allison. Univariate polynomial inference by Monte Carlo message length approximation. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002)*, July 2002.

[40] Leigh J. Fitzgibbon, Lloyd Allison, and David L. Dowe. Minimum message length grouping of ordered data. In *Algorithmic Learning Theory,*

11th International Conference, ALT 2000, Sydney Australia, December 2000, Proceedings, volume 1968 of *Lecture Notes in Artificial Intelligence*, pages 56–70. Springer, Berlin, 2000.

[41] Kenneth D. Forbus. Qualitative process theory. In D. Bobrow, editor, *Qualitative reasoning about physical systems.* MIT press, 1985.

[42] J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19(1):1–67, 1991.

[43] Jerome H. Friedman and Werner Stuetzle. Projection pursuit regression. *Journal of the American Statistical Association*, 76(376):817–823, December 1981.

[44] E. Gold. Language indentification in the limit. *Information and Control*, 10:447–474, 1967.

[45] Diana F. Gordon and Marie desJardins. Evaluation and selection of biases in machine learning. *Machine Learning*, 20(1/2):5–22, 1995.

[46] P. Grünwald. Model selection based on minimum description length. *Journal of Mathematical Psychology*, 2000.

[47] J. Hajek. *A Course in Nonparametric Statistics.* Holden-Day, San Francisco, CA, 1969.

[48] D. J. Hand. *Kernel Discriminant Analysis.* Research Studies Press, Chichester, 1982.

[49] P. J. Hayes. The naive physics manifesto. In D. Michie, editor, *Expert Systems in the Electronic Age*, pages 242–270. Edinburgh University Press, Edinburgh, Scotland, 1979.

[50] R. Jones. Generating predictions to aid the scientific discovery process. In *Fifth National Conference on Artificial Intelligence*, 1986.

[51] Michael Kearns, Yishay Mansour, Andrew Y. Ng, and Dana Ron. An experimental and theoretical comparison of model selection methods. *Machine Learning*, 27:7–50, 1997.

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

[53] A.N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1:4–7, 1965.

[54] K. Korb and M. Stillwell. The story of the hot hand: Powerful myth or powerless critique. *under submission*, 2002.

[55] D. kulkarni and H.A. Simon. *Computational models of scientific discovery and theory formation*, chapter Experimentation in Machine Discovery. Morgan Kaufmann, 1990.

[56] D. L. Dowe L. J. Fitzgibbon and L. Allison. Change-point estimation using new minimum message length approximations. In *Proceedings of the Seventh Pacific Rim International Conference on Artificial Intelligence (PRICAI-2002)*, LNAI. Springer-Verlag, August 2002.

[57] P. Langley. Data-driven discovery of physical laws. *Cognitive Science*, 1981.

[58] P. W. Langley. BACON: A production system that discovers empirical laws. In Raj Reddy, editor, *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pages 344–344, Cambridge, MA, August 1977. William Kaufmann.

[59] Pat Langley. The computer-aided discovery of scientific knowledge. In *First International Conference on Discovery Science*, 1998.

[60] Michael Lebowitz. Not the path to perdition: The utility of similarity-based learning. In Tom Kehler and Stan Rosenschein, editors, *Proceedings of the 5th National Conference on Artificial Intelligence. Volume 1*, pages 533–537, Los Altos, CA, USA, August 1986. Morgan Kaufmann.

[61] Michael Lebowitz. Experiments with incremental concept formation: UNIMEM. *Machine Learning*, 2:103–138, 1987.

[62] D. B. Lenat. Automated theory formation in mathematics. In *Fifth International Joint Conference on Artificial Intelligence*, 1977.

[63] Per Martin-Löf. The definition of random sequences. *Information and Control*, 9(6):602–619, December 1966.

[64] R. S. Michalski. A theory and methodology of inductive learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, volume I, pages 83–134, Palo Alto, CA, 1983. Tioga.

[65] R.S. Michalski and R. Stepp. *Machine Learning: An artificial intelligence approach*, chapter Learning from observation: Conceptual Clustering. Morgan Kaufmann, 1983.

[66] Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell. *Machine learning : an artificial intelligence approach, vol.1, (ed.)*. Palo Alto, Calif. : Tioga Pub. Co., 1983, CALL NUMBER: Q325 .M32 1983, 1983.

[67] Marvin L. Minsky. A framework for representing knowledge. In Patrick Henry Winston, editor, *The Psychology of Computer Vision*, pages 211–277. McGraw-Hill, New York, 1975.

[68] Tom M. Mitchell. The need for biases in learning generalizations. Technical Report CBM-TR-117, Department of Computer Science, Rutgers University, New Brunswick, New Jersey, May 1980.

[69] M. Moulet. The role of measurement uncertainty in numeric law induction. *Lecture Notes in Computer Science*, 945:619–??, 1995.

[70] Keiichi Noe. Philosophical aspects of scientific discovery: A historical survey. In *Discovery Science, First International Conference*, 1998.

[71] B. Nordhausen and P. Langley. An integrated discovery system. In *Qualitative Reasoning Workshop Abstracts*. Qualitative Reasoning Group, University of Illinois at Urbana-Champaign, 1987.

[72] Bernd Nordhausen and Pat Langley. An integrated framework for empirical discovery. *Machine Learning*, 12:17–47, 1993.

[73] J. J. Oliver, R. A. Baxter, and C. S. Wallace. Minimum message length segmentation. *Res' and Dev' in Knowledge Discovery and Data Mining (PAKDD-98)*, pages 222–233, 1998.

[74] J. J. Oliver and C. S. Forbes. Bayesian approaches to segmenting a simple time series. Technical Report 97/336, Dept. Computer Science, Monash University, Australia 3168, dec 1997.

[75] Jan Zytkow Pat Langley et al. *Computational Explorations of the Creative Processes*. MIT Press, Cambridge, 1987.

[76] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1991. (Revised 2nd Edition).

[77] Joseph Phillips. Towards a method of searching a diverse theory space for scientific discovery. *Lecture Notes in Computer Science*, 2226:304–??, 2001.

[78] Karl R. Popper. *The Logic of Scientific Discovery.* Basic Books, New York, 1959.

[79] J. R. Quinlan. Learning logical definitions from relations. *Machine Learning.* 5(3):239–266, ? 1990.

[80] J. R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, San Mateo, CA, 1993.

[81] H. Reichenbach. *Experience and Prediction.* University of Chicago Press, 1938.

[82] L. Rendell. A general framework for induction and a study of selective induction. *Machine Learning*, 1(2):177–226, 1986.

[83] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, 1996.

[84] J. Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[85] J. Rissanen. Stochastic complexity and modeling. *Ann. of Statist.*, 14(3):1080–1100, 1986.

[86] J. Rissanen. *Stochastic Complexity in Statistical Inquiry.* World Scientific, N.J., 1989.

[87] J. Rissanen. Hypothesis selection and testing by the MDL principle. *The Computer Journal*, 42(4):260–269, 1999.

[88] J. J. Rissanen. Fisher Information and Stochastic Complexity. *IEEE Trans on Information Theory*, 42(1):40–47, January 1996.

[89] Donald Rose and Pat Langley. Chemical discovery as belief revision. *Machine Learning*, 1:423, 1986.

[90] Donald Rose and Pat Langley. Chemical discovery as belief revision. *Machine Learning*, 1:423–451, 1986.

[91] Y. Sakamoto et al. *Akaike information criterion statistics*, pages 191–194. KTK scientific publishers, 1986.

[92] S. Schaal. Nonparametric regression for learning. In J.A. Storer and M. Cohn, editors, *Proc. Conference on Adaptive Behavior and Learning*, pages 123–133. University of Bielefeld, Bielefeld, Germany, 1994.

[93] Cullen Schaffer. A proven domain-independent scientific function-finding algorithm. In William Dietterich, Tom; Swartout, editor, *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 828–833, Hynes Convention Centre?, July 29–August 3 1990. MIT Press.

[94] Cullen Schaffer. Bivariate scientific function finding in a sampled, real-data testbed. *Machine Learning*, 12:167–183, 1993.

[95] Klaus Schmid. Creative problem solving and automated discovery — an analysis of psychological and AI research. Technical Memo TM-95-04, Deutsches Forschungszentrum für Künstliche Intelligenz, 1995.

[96] G. Schwarz. Estimating the Dimension of a Model. *Ann. Stat.*, 6:461–464, 1978.

[97] Herbert A. Simon, Valdés-Perez, and Sleeman. Scientific discovery and simplicity of method. *Artificial Intelligence*, 91(2):177–181, 1997.

[98] R. Solomonoff. A formal theory of inductive inference, I and II. *Information and Control*, 7:1–22 and 224–254, 1964.

[99] R. J. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7:1–22, 224–254, 1964.

[100] R. J. Solomonoff. Complexity-based induction systems: Comparisons and convergence theorems. *IEEE Trans. on Information Theory*, 1978.

[101] Ray J. Solomonoff. The discovery of algorithmic probability. *Journal of Computer and System Sciences*, 55(1):73–88, August 1997.

[102] M. Stillwell. *Investigating the "Hot Hand" Phenomenon.* PhD thesis, Dept. Computer Science, Monash University, Australia 3168, 1998.

[103] M. Stone. Cross-validatory choice and assessment of statistical predictions (with discussion). *Journal of the Royal Statistical Society series B*, 36:111–147, 1974.

[104] C. J. Thornton. *Truth from Trash: How Learning Makes Sense.* MIT press, Cambridge, MA, 2000.

[105] P. E. Utgoff. *Machine learning of inductive bias.* Kluwer, Hingham, MA, 1986. Reviewed in IEEE Expert, Fall 1986.

[106] P. E. Utgoff. Shift of bias for inductive concept learning. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An*

Artificial Intelligence Approach, volume II, pages 107–148, Los Altos, CA, 1986. Morgan Kaufmann.

[107] R. E. Valdes-Perez. Machine discovery in chemistry: New results. *Artificial Intelligence*, 1995.

[108] R. E. Valdes-Perez. Principles of human computer collaboration for knowledge discovery in science. *Artificial Intelligence*, 1999.

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

[110] P. J. van Heerden. A general theory of prediction. Technical report, Polaroid Corp., 1963.

[111] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation, and signal processing. In M. Mozer, M. Jordan, and T. Petsche, editors, *Advances in Neural Information Processing Systems 9*, pages 281–287, Cambridge, MA, 1997. MIT Press.

[112] V. N. Vapnik. *Estimation of Dependencies Based on Empirical Data.* Springer-Verlag, New York, 1982.

[113] V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, 1995.

[114] V. N. Vapnik. *Computational Learning and Probabilistic Reasoning*, chapter Structure of Statistical Learning Theory. Wiley and Sons, 1996.

[115] M. Viswanathan and C. Wallace. A note on the comparison of polynomial selection methods. In *Proc. 7th Int. Workshop on Artif. Intell. and Stats.*, pages 169–177. Morgan Kauffman, January 1999.

[116] M. Viswanathan, C.S. Wallace, D.L. Dowe, and K. Korb. Finding cut-points in noisy binary sequences - a revised empirical evaluation. In *Proc. 12th Australian Joint Conference on Artificial Intelligence*, pages 405–416, Sydney, Australia, December 1999.

[117] C. S. Wallace. *Computational Learning and Probabilistic Reasoning*, chapter 3, pages 43–66. Wiley, 1996.

[118] C. S. Wallace and D. L. Dowe. Refinements of MDL and MML coding. *The Computer Journal*, 42(4):330–337, 1999.

[119] C.S. Wallace. Multiple Factor Analysis by MML Estimation. Technical Report 95/218, Dept. of Computer Science, Monash University, Clayton, Victoria 3168, Australia, 1995. Accepted, to appear in J. Multiv. Analysis.

[120] C.S. Wallace. On the selection of the order of a polynomial model. Technical report, Royal Holloway College, London, 1997.

[121] C.S. Wallace. Intrinsic Classification of Spatially-Correlated Data. *Computer Journal*, 41(8):602–611, 1998.

[122] C.S. Wallace and D.M. Boulton. An information measure for classification. *Computer Journal*, 11(2):195–209, 1968.

[123] C.S. Wallace and D.M. Boulton. An invariant Bayes method for point estimation. *Classification Society Bulletin*, 3(3):11–34, 1975.

[124] C.S. Wallace and D.L. Dowe. MML estimation of the von Mises concentration parameter. Tech Rept TR 93/193, Dept. of Computer Science, Monash Univ., Clayton, Victoria 3168, Australia, 1993. prov. accepted, Aust. and N.Z. J. Stat.

[125] C.S. Wallace and D.L. Dowe. Minimum message length and Kolmogorov complexity. *Computer Journal*, 42(4):270–283, 1999.

[126] C.S. Wallace and D.L. Dowe. MML clustering of multi-state, Poisson, von Mises circular and Gaussian distributions. *Statistics and Computing*, 10(1):73–83, 2000.

[127] C.S. Wallace and P.R. Freeman. Estimation and inference by compact coding. *J. R. Statist. Soc B*, 49(3):240–265, 1987.

[128] C.S. Wallace and P.R. Freeman. Single factor analysis by MML estimation. *Journal of the Royal Statistical Society (Series B)*, 54:195–209, 1992.

[129] G. Wallas. *The Art of Thought.* Harcourt-Brace, New York, 1962.

[130] M. P. Wand and M. C. Jones. *Kernel Smoothing.* Chapman & Hall, London, 1995.

[131] Yihsiao Wang. Randomness, stochasticity, and approximations. *Theoretical Computer Science*, 32:517–529, 1999.

[132] Kenneth H. Wasserman. *Unifying Representation and Generalization Understanding Hierarchically Structured Objects.* PhD thesis, Columbia University, 1985.

[133] G. Webb. A heuristic covering algorithm outperforms learning all rules. In *Proceedings of ISIS'96: Information, Statistics and Induction in Science*, pages 20–30. Singapore: World Scientific, 1996.

[134] D. G. Willis. Computational complexity and probability constructions. *Journal of the Assoc. of Comp. Mach.*, 1970.

[135] R. Zembowicz and J. M. Zytkow. Automated discovery of empirical equations from data. *Lecture Notes in Computer Science*, 542:429–??, 1991.

[136] R. Zembowicz and J. M. Zytkow. Automated discovery of empirical equations from data. In M. Ras, Z.W.; Zemankova, editor, *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems (ISMIS'91)*, volume 542 of *LNAI*, pages 429–440, Charlotte, N.C., USA, October 1991. Springer Verlag.

[137] R. Zembowicz and J. M. Zytkow. Recognition of functional dependencies in data. In J. Komorowski and Z. W. Raś, editors, *Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems (ISMIS'93)*, volume 689 of *LNAI*, pages 632–641, Trondheim, Norway, June 1993. Springer Verlag.

[138] Robert Zembowicz and Jan M. Zytkow. Discovery of equations: Experimental evaluation of convergence. In William Swartout, editor, *Proceedings of the 10th National Conference on Artificial Intelligence*, pages 70–75, San Jose, CA, July 1992. MIT Press.

[139] J. M. Zytkow. Incremental discovery of hidden structure: Applications in the theory of elementary particles. In *Thirteenth National Conference on Artificial Intelligence*, 1996.

[140] J. M. Zytkow and P. J. Fischer. Constructing models of hidden structure. In M. Ras, Z.W.; Zemankova, editor, *Proceedings of the 6th International Symposium on Methodologies for Intelligent Systems (ISMIS'91)*, volume 542 of *LNAI*, pages 441–449, Charlotte, N.C., USA, October 1991. Springer Verlag.

[141] Jan Zytkow, editor. *Machine Discovery.* Kluwer Academic, Boston, USA, 1997.

[142] Jan M. Zytkow and Paul J. Fischer. Incremental discovery of hidden structure: Applications in theory of elementary particles. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 750–756, Menlo Park, August 4–8 1996. AAAI Press / MIT Press.

[143] Jan M. Zytkow and Herbert A. Simon. A theory of historical discovery: The construction of componential models. *Machine Learning*, 1:107, 1986.

[144] Jan M. Zytkow and Herbert A. Simon. A theory of historical discovery: The construction of componential models. *Machine Learning*, 1:107–136, 1986.