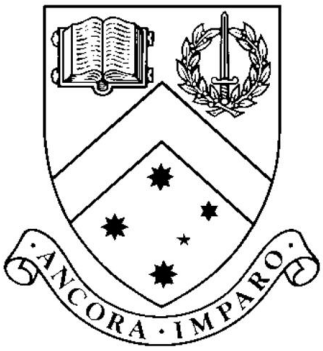
MML Inference of Decision Trees, Graphs and Forests

by

Peter J. Tan



Thesis

Submitted by Peter J. Tan

for fulfillment of the Requirements for the Degree of

Doctor of Philosophy (0190)

Supervisor: Assoc. Prof. David L. Dowe

Associate Supervisor: Assoc. Prof. Trevor I. Dix

Clayton School of Information Technology
Monash University

February, 2017

To my family and parents

Contents

List of Tables

List of Figures

MML Inference of Decision Trees, Graphs and Forests

Peter J. Tan
Monash University, 2017

Supervisor: Assoc. Prof. David L. Dowe
Associate Supervisor: Assoc. Prof. Trevor I. Dix

Abstract

The Minimum Message Length (MML) principle is a general inductive inference framework which recasts inductive inference as a coding process. Using assumptions identical to those from the well-known Bayesian inference, prediction and modelling, MML represents general inductive and statistical inference problems as a data encoding process which conveys the data to a receiver in a two-part message. The first part is the message to encode the inferred model. The second part is the message to encode the data in light of the inferred model. MML states that the model with the shortest two-part message is the best model approximating the true model.

Decision trees, decision graphs and decision forests are popular supervised learning methods in machine learning. In this dissertation, the MML principle is applied to build machine learning schemes for decision trees, decision graphs and decision forests. Two novel MML inference schemes are developed. One is a MML coding scheme for Oblique decision trees, which are decision trees with linear discriminate functions at their internal nodes. Another is a MML coding scheme for decision graphs with multi-way joins and dynamic attributes. A decision forests learning scheme based on MML oblique decision trees is also presented.

Experiments were conducted across a range of problems using data from University of California Machine Learning Repository and the Singapore Data Mining Centre. These experiments showed that compared to other popular decision tree models such C4.5 and C5, models generated by MML inference schemes achieved

favourable results in both classification and probabilistic predictive accuracy. The study showed that MML inference schemes are able to find the optimal trade-off between the complexity of these structure models and goodness-of-fit for a given set of data.

MML Inference of Decision Trees, Graphs and Forests

Declaration

I declare that this thesis is my own work and has not been submitted in any form for another degree or diploma at any university or other institute of tertiary education. Information derived from the published and unpublished work of others has been acknowledged in the text and a list of references is given.

Peter J. Tan
February 17, 2017

Acknowledgments

First of all, I would like to thank both of my supervisors, for their guidance, encouragement and selfless care. I have learned a lot from my main supervisor, Assoc. Professor David Dowe, who always impresses me with his great attention to details and sense of humour even in many adverse circumstances. Many thanks also go to my second supervisor, Assoc. Professor Trevor Dix, who provided resources for my studies when I most needed them and helped me polishing my writing with great patience.

I was very fortunate to be in company of a group of talented post graduate students. To Josh, Leigh, Rodney, Tim, Shane, Lara and Yudi, thank you not only for the valuable discussions on research topics, but also for all the fun memories we shared. Those are highlights during my postgraduate studies at Monash. I would also like to thank my employer, Geoscience Australia, who generously provided a three-month study leave for me to work on my thesis.

Lastly, I would like to thank my family, especially my wife Jenny, for her never-ending love and support, it makes this long journey a worthwhile experience.

Peter J. Tan

Monash University
February 2017

Chapter 1

Introduction

The aim of this research is to build machine learning systems with high classification and probabilistic predictive accuracy, using decision trees, graphs and forests inferred by the minimum message length (MML) principle. This chapter comprises four parts: brief explanations of the keywords and concepts mentioned above; a discussion on motivation of the research; outlines of thesis chapters; and a statement of the contribution of the work presented in this thesis. The intention is to provide a relatively informal background to the thesis before giving rigorous definitions, analyses and discussions.

1.1 Machine Learning

Few people argue against the importance of learning. Although quotes similar to Aldous Huxley's claim that the most important thing we learn from history is that we never learn from history often appear in the media, human learning can be considered as very successful so far. Human beings have always learned, and an enormous knowledge base has been built. After all, human beings are born with very limited knowledge or skill.

There are many definitions of learning. One from the Wikipedia states "Learning is the process of acquiring knowledge, skills, attitudes, or values, through study, experience, or teaching." Apparently such a definition is confined to learning performed by a human being. As such, questions like "what is machine learning and

why are such systems needed?" naturally arise. It also seems appropriate to be the starting point of discussions herein.

As for the term learning, there are inevitably many definitions of the term "machine learning". The definition used here is:

" Machine learning is a branch of artificial intelligence concerned with automatic processes of analysing data performed by computer programs."

In fact, machine learning is just another utility to perform often complicated and difficult tasks for human beings. From this perspective, the role of machine learning programs is not much different from machines in beer factories, except that the final product of a machine learning program is information. Machine learning can be viewed as a process of information transformation. Data are continuously collected and streamed into databases every day. Most of the data are eventually locked in so-called data warehouses. One of the impacts of rapidly advancing IT technologies is the seemingly exponential growth in data collection, which creates both unprecedented challenges and enormous opportunities for machine learning research. On one hand, large amounts of data are collected from sources where there used to be no records, so more industries are able to benefit from machine learning techniques. On the other hand, it demands that machine learning algorithms must be able to keep pace with newly created data sets, which are increasing in size and complexity. The great challenge for machine learning research is to develop computer systems that are capable of exploring such data sets automatically and intelligently.

In recent years, machine learning systems have kept advancing, often exceeding expectations. Some of the machine learning systems have already outperformed experts and have great potential for widening the gap. A well-known example is the Deep Blue computer program which plays chess. Since the world chess champion was first beaten by Deep Blue, human players have been unable to beat Deep Blue or its successors.

The operation of machine learning programs follows pre-defined instructions and procedures known as algorithms. In this respect, machine learning algorithms are

blueprints for machine learning systems, which can also be regarded as implementations of machine learning algorithms. In computer science, the main concern for machine learning research is to devise effective and efficient algorithms for various machine learning tasks.

This thesis investigates tasks arising from following machine learning paradigm: given a limited number of data sampled from a population, theories which are consistent with the whole (unseen) population are inferred. Such machine learning paradigms which infer theories from a body of known samples are often referred to as supervised learning. In practice, the inferred theories are mathematical models which are usually aimed at two tasks: classification and probabilistic prediction. To classify a piece of data, the learned models take values of features of the data as input and assign a class label to the data. In some situations, when probabilities on class labels are desired, the models output estimated probabilities on each class. The goal for supervised learning algorithms is to learn the optimal mathematical models which maximise the classification or probabilistic prediction accuracy on new data sampled from the same population.

The algorithms developed in this thesis are to construct decision trees, graphs or forests from data using MML inductive inference schemes. Depending on the purpose of the tasks (i.e., pattern discovery or classification), appropriate learning models and relevant algorithms are developed and selected to tackle the task.

1.2 Minimum Message Length Principle

The minimum message length (MML) principle creates a formal framework for inductive inference which, in this thesis, is confined to processes that infer mathematical models for given data sets. As such, the minimum message length principle is the theory underpinning the proposed decision tree, decision graph and decision forest learning algorithms. For a given fixed body of data, MML theory defines a term called the explanation message on all competing candidate inference models. The explanation message comprises two parts. The first is the message to encode the inferred model. The second part is the message to encode the data in light of the inferred model. The encoding of the first part takes into account the complexity

of the model and the second part the probability of each observed data item under the model. There are a great and often indefinite number of mathematical models inferred from the given data. MML states that the model with the shortest two-part message length is the model best approximating the true model.

1.3 Decision Trees, Graphs and Forests

Consider the following example. A power plant had a problem of chronic generator failures and the engineers in the plant were unable to determine when it typically occurs. The whole process was monitored via hundreds of sensors and readings from various meters were recorded regularly. The data from readings were logged and accumulated in a database. To simplify the illustration, suppose one record is comprised of readings from only four sensors and the temperature indication of the targeted engine at the time. A higher than normal temperature indicates a high risk of generator failures, while a generator running normally has a low temperature. A snapshot of a part of the database is shown below.

S1	S2	S3	S4	Temperature
23	23	23	34	High
32	43	32	43	Low
78	75	39	46	Low
123	36	11	23	Low
32	32	99	32	High
...	...	...	...	...

A data analysis specialist was asked to find when the problem typically occurs. That is, to find out under what combinations of conditions (e.g., reading from meters), the generator was at high risk of failure. One of the solutions was to construct a decision tree with data from the database. The inferred decision tree, as shown in Fig. 1.1, reveals the conditions that led to the power failures. This information can then serve as a guideline for operators in the plant to prevent future failures.

This is only a simplified example to show how business and industry could benefit from machine learning and decision tree techniques. The concept of decision trees

first appeared in expert systems, where trees are often drawn to illustrate the process of decision making by experts in specific domains. From this point of view, a decision tree is a form of knowledge representation. Decision tree learning plays an important role in machine learning research, mainly due to the powerful performance of decision trees yet simplicity of implementation. The wide-spread uses of decision trees could also be attributed to their intuitive forms. As shown in the above example, the underlying patterns revealed by decision trees are easily comprehensible to people without an artificial intelligence (AI) background. From the root node at the top of a decision tree, a specific test is performed at each internal tree node, which subsequently decides which route will be followed. A sequence of tests is performed until the data reaches a leaf node, at which point the data is associated with a label.

Decision trees are comprehensible representations that have been widely used in many machine learning domains. But in the area of supervised learning, decision trees have their limitations. Two notable problems are those of replication of similar subtrees below different branches, and fragmentation into smaller sets of instances

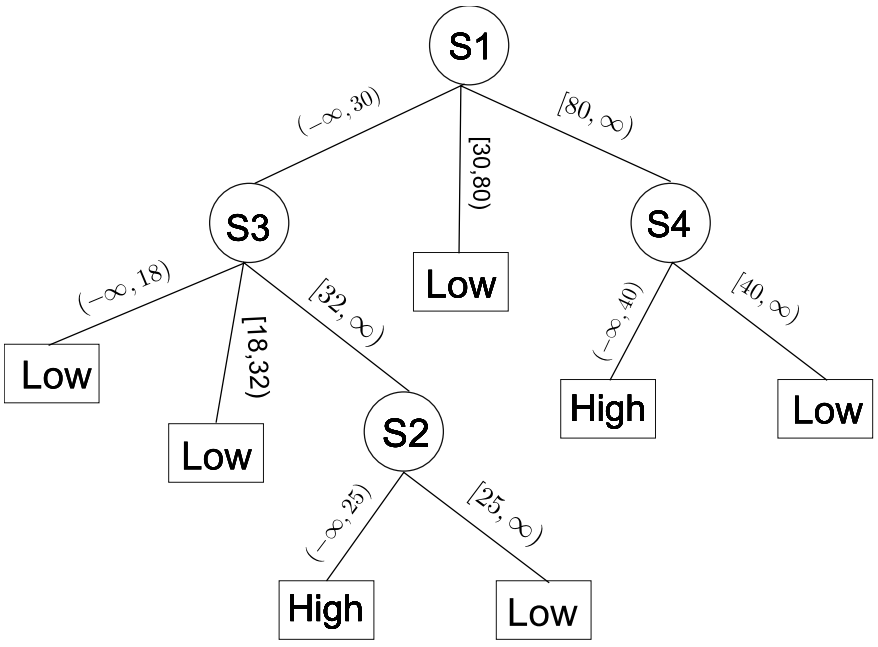


Figure 1.1: A sample decision tree inferred from power plant data

satisfying the tests as a route is followed. One way of solving these problems is to introduce decision graphs, a generalization of the decision tree, which address the above problems by allowing for disjunctions, or joins. Decision graphs are a natural extension of decision trees. Decision graphs could be viewed as generalizations of decision trees; both decision trees and decision graphs have decision nodes and leaves. The feature that distinguishes decision graphs from decision trees is that they may also contain joins (or disjunctions), which are represented by two or more nodes having a common child. Decision graphs are excellent learning models when inferring concepts with disjunctions from a limited number of data instances.

1.4 Ensemble Learning and Applications on Real World Data

Ensemble learning is one of the major advances in supervised learning research in recent years. The outputs of an ensemble classifier are determined by a committee, in which a group of classifiers cast (possibly weighted) votes on final predictions. Generally, ensemble learning schemes are able to outperform single classifiers in predictive accuracy. The intuitive explanation for the success of ensemble learning is that mistakes made by individual classifiers are corrected by complementary results submitted by other classifiers in the committee.

There are many variations of ensemble learning algorithms, depending on the types of base learners. The base learners are the learning algorithms which infer individual classifiers in an ensemble learning scheme. In this thesis, MML oblique decision trees are the chosen base learners. A decision "tree" forest is a group of decision trees whose individual decisions are combined and averaged to classify new incoming data. Due to complexity caused by model assembling, decision forests are not the best model to showcase underlying patterns from the data. However, predictive accuracy provided by decision forests on new test data are usually among the best, often performing better than individual decision trees and graphs. Hence,

decision forests are often selected as preferred inference models when there are machine learning problems that require only high quality predictors (also known as "black box" solutions).

Real world data sometimes differs vastly from the data tested in machine learning research. A real world data set may comprise a large number of instances with a large number of noisy input features. Noise, complexity and size of the data set pose new and different challenges to machine learning researchers. For many machine learning algorithms, computational time grows exponentially with the number of input features. The purpose of feature selection for a supervised learning algorithm is to identify attributes irrelevant to the target attributes. After processes of feature selection, data sets will only consist of input features most relevant to the target attribute, thus enabling learning algorithms to run more efficiently. The size of data sets is often a big obstacle for learning algorithms. Some databases consist of millions of instances of data. Most learning algorithms could not be applied to the whole data set directly.

1.5 Overview of the Chapters

Minimum Message Length: Chapter 2 presents the reader with background knowledge of the Minimum Message Length Principle. It begins with an introduction to MML inference and is followed by discussions of its relation to information theory, Bayesian inference and statistical learning, including properties and advantages of MML inference. In the latter part of the chapter, strict MML (SMML) is discussed in detail, then several important approximations to SMML estimators are introduced, as these estimators are more practical for developing MML inference schemes.

Decision Trees: Chapter 3 includes definitions of decision trees, algorithms that grow decision trees from given data, and differences and relations between some existing decision tree schemes. The following discussions focus on details of MML decision tree inference schemes. As the fundamental models for research in this thesis, various aspects of the MML decision tree inference scheme are examined in detail. Another important part of the chapter is dedicated to evaluation schemes for performances of decision trees. Probabilistic costing (or log-loss), a scheme for

testing performance of decision trees on probabilistic predictions, is also proposed. The final part of Chapter 3 concerns oblique decision trees which are decision trees with linear discriminate functions at their internal nodes. The motivation for oblique decision tree research and a novel MML oblique decision tree inference scheme are presented, and its relation with support vector machines are examined. Detailed discussions of the proposed MML oblique decision tree algorithm include a proposed MML coding scheme and search heuristics for finding optimal oblique trees.

Decision Graphs: The introduction of Chapter 4 reveals the motivation for inferring decision graphs and explains why the proposed novel decision graph algorithm with multi-way joins offers more flexibility than similar decision graph algorithms. It then continues with discussions on various topics, which include: the advantages and disadvantages of using decision graphs over decision trees; MML decision graph coding schemes; relations between MML decision tree and decision graph algorithms; dynamic attributes in decision graphs; comparisons with related decision graphs algorithms; and random search algorithms for MML decision graph inference; analysis of experimental results. MML decision graphs achieved the highest predictive accuracy on the "abalone" data set in the literature [186, 187].

Decision Forests: Decision forests group multiple decision trees to increase predictive accuracy, and are a form of ensemble learning. The first section of Chapter 5 explains ensemble learning and why it works better than the single model algorithms. A discussion of main elements of ensemble learning algorithms is then presented, followed by a review of well tested ensemble learning algorithms. A detailed analysis of a novel decision forest algorithm using MML oblique decision trees is then presented, followed by experimental results and comparisons with other ensemble learning algorithms.

The latter part of Chapter 5 is devoted to applications of the proposed machine learning algorithms to real world data sets. Strategies to resolve the issues of noise, complexity and size associated with the real world data sets are reviewed. Examples

of applying the proposed learning algorithms to some real world data sets are elaborated. There are also comprehensive studies on two specific cases: classification of gene expression data sets and probabilistic predictions on large data sets.

1.6 Contributions of the Thesis

The work presented here makes a number of contributions to machine learning research within minimum message length framework. All novel MML learning schemes and their implementations have been published by the author:

- MML Oblique decision trees and the importance of considering different or appropriate evaluation schemes appear in section 3.5 and section 3.3 and is published in [188].
- MML Multi-way join decision graphs are presented in section 4.2 and extensions for dynamic attributes in section 4.6 and is published in [186, 187].
- Decision forest of MML Oblique decision trees appear in section 5.3 and is published in [189].
- Applications of developed MML schemes to model various real world problems are published in [190, 189, 42, 116].

All these implementations have been applied to real world data and comparisons made against other machine learning programs (mainly decision tree program C4.5 and C5 [164]). The comparisons show the novel algorithms presented in the thesis are capable of inferring models with improved performance. This study was focused on MML inference of various machine learning models. After the study completed, algorithms and programs developed in this study continue playing important roles in author's recent research [185, 183, 184, 115, 125], which involve modelling remote sensing time series data.

Chapter 2

Minimum Message Length

The Minimum Message Length (MML) principle is a general inductive inference framework developed by Wallace et al.[209, 207]. Using assumptions identical to those from the well-known Bayesian inference, prediction and modelling, MML represents general inductive and statistical inference problems as a data encoding process by seeking a minimum two-part message length. This chapter first presents a concise review of the MML principle in order to provide a thorough explanation of the MML framework. It is not the author's intention, and is also beyond the capacity of a single chapter, to cover every aspect of the MML principle. The bulk of the chapter is then devoted to an overview of the theoretical foundation of the MML principle and the development of some important MML approximations. To obtain a thorough investigation of MML, readers are encouraged to read Chris Wallace's excellent book [207] on MML, PhD theses by MML research students [22, 11, 1, 83, 38] and many publications on various MML topics [218, 209, 214].

2.1 Introduction

In the field of statistical estimation, the goal of a Bayesian estimator is to estimate unknown population parameters from a set of sample data by using Bayes's theorem. Supposing there is a set of models each with a discrete parameter θ_i, let Θ be the countable set of all possible values of the parameter $\{\theta_1, \theta_2, \ldots\}$. Given a body of data x, Bayesian inference attempts to find $\theta_i \in \Theta$ which maximises the probability

$Pr(\theta_i|x)$, which is the probability of the parameter taking the value of θ_i, given the presence of observed data x. The probability distribution $\{Pr(\theta_i|x)\}$ is called *the posterior distribution*, which is a new distribution over the set Θ. Using Bayes's theorem, we have

$$Pr(\theta_i|x) = \frac{Pr(x|\theta_i)Pr(\theta_i)}{Pr(x)} \qquad \forall \theta_i \in \Theta$$

where

$$Pr(x) = \sum_i Pr(x|\theta_i)Pr(\theta_i)$$

is called the *marginal data probability* or *marginal probability* of x, and is independent of the choice of θ_i. Assuming a prior probability $Pr(\theta_i)$ is defined over every possible parameter value, and the likelihood function $Pr(x|\theta_i)$ is known for every possible parameter value, it is possible to find the value θ_i that maximises the posterior probability $Pr(\theta_i|x)$. In this case, Bayesian inference is able to choose the model with the highest posterior probability as the most probable model from the given data x.

When the unknown parameter is a (multi-dimensional) real value vector, the result of a Bayesian inference is a probability density function of the unknown parameter, $p(\theta|x)$, called the *posterior (density) function.* In such cases, the prior probability distribution $\{Pr(\theta_i)\}$ is replaced by $h(\theta)$, which is a prior probability density over Θ. Using Bayes's theorem we arrive at a posterior density on Θ

$$p(\theta|x) = \frac{Pr(x|\theta)h(\theta)}{Pr(x)} \qquad \forall \theta \in \Theta$$

where the marginal probability of data x now becomes

$$Pr(x) = \int_\Theta Pr(x|\theta)h(\theta)d\theta$$

It becomes impossible to assign non-zero prior probabilities to any possible value of the parameter θ. Often, the posterior density function is interpreted in conjunction with loss functions to infer the best point estimate, i.e., find the point estimate

minimising

$$\int_{\Theta} l(\theta - \hat{\theta}) p(\theta|x) d\theta$$

However, in cases where the "real" loss function is unavailable because the nature of the circumstance is unknown, generic loss functions are chosen to fulfill the task. One popular choice is the negative of a delta function. Minimising

$\int_{\Theta}(1 - \delta(\theta - \hat{\theta}))p(\theta|x)d\theta$, where

$$\lim_{\epsilon \to \infty} \delta(\theta - \hat{\theta}) = \begin{cases} \frac{1}{2\epsilon} & : \quad |\theta - \hat{\theta}| < \epsilon \\ 0 & : \quad |\theta - \hat{\theta}| \geq \epsilon \end{cases}$$

leads to the maximum a posteriori (MAP) estimator. Another commonly used loss function is the squared difference of coordinates. Minimising $\int_{\Theta}(\theta - \hat{\theta})^2 p(\theta|x)d\theta$ leads to the minimum mean squared error (MMSE) estimator. One may be tempted to select the model with the highest probability density. But unfortunately the MAP estimator and the MMSE estimator are not statistically invariant under reparameterisation. The following example illustrates how the values of the point estimates by MAP vary when the coordinates are changed. Considering the problem of estimating the mean of the bivariate Normal distribution with an identity covariance matrix, in polar coordinates (r, θ), the likelihood function is $p(r, \theta|\hat{r}, \hat{\theta}) = \frac{1}{2\pi} exp\big(-|r - \hat{r}|^2\big)$. In rectangular coordinates (x, y), the likelihood function becomes $p(x, y|\hat{x}, \hat{y}) = \frac{1}{2\pi} exp\big(-(x - \hat{x})^2 - (y - \hat{y})^2\big)$. Given that $x = r\cos\theta, y = r\sin\theta$, the prior density function $h(x, y) = J^{-1} \cdot h(r, \theta)$, where J is the Jacobian determinant and is given by the formula

$$J \quad = \quad \det \begin{bmatrix} \frac{\partial x}{\partial r} & \frac{\partial x}{\partial \theta} \\ \frac{\partial y}{\partial r} & \frac{\partial y}{\partial \theta} \end{bmatrix} \quad = \quad \det \begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix} \quad = \quad r$$

For a given set of data X containing N items, the posterior functions under polar coordinates and under rectangular coordinates are respectively

$$posterior(r, \theta) \propto h(r, \theta) \cdot \prod_{i=1}^{N} p(r_i, \theta_i|\hat{r}, \hat{\theta})$$

$$posterior(x, y) \propto \frac{h(r, \theta)}{r} \cdot \prod_{i=1}^{N} p(x_i, y_i | \hat{x}, \hat{y})$$

It is obvious the point estimate made by MAP is not identical under the two different coordinates because of the Jacobian determinant introduced in the process of transformation of the prior. Detailed examples which show such lack of invariance in conventional Bayesian inference methods (such as MAP) can be found in [65], [11, section 3.4.1] and [49, footnote 158].

Part of the motivation behind MML is that conventional Bayesian estimators such as MAP and MMSE are unable to make invariant parameter estimations for models with continuous parameters. MML overcomes this issue by being statistically invariant. It also has other advantages over conventional Bayesian inferences. As shown in Fig 2.1, MML can be illustrated by the following hypothetical communication scenario: A sender attempts to convey a body of data to a receiver who listens at the other end of a noiseless communication channel. The aim for the sender is to transmit the set of the observed data with the least number of two-part code words, i.e., the shortest two-part message length in a lossless manner. To simplify the investigation, assume that there is some prior information which is agreed upon and shared by both the sender and receiver. The shared information, such as the range of attributes of the data and the inferred model class, is essential for the communication, as the receiver needs such information to recover the encoded data. The sender may send the data in one part, in which case the sender only encodes the data using the shared prior information but does not encode any inferred hypothesis. Alternatively, the sender may send the data in two parts. In the first part, the sender states a hypothesis or theory about the observed data. In effect, the sender builds a code book which is used to encode the data later in the second step. Then the sender transmits the code book to the receiver, who recovers the code book with the help of the shared prior information. In the second step, the data is encoded by the code book generated in the first step and then sent to the receiver. At the other end of the channel, the receiver decodes the received code words by referring to the shared codebook. In this way, the original data set is conveyed in a two-part message. If the length of the two-part message is shorter than that resulting from the one-part approach, the data is in effect compressed and the hypothesis or theory

stated in the first step represents an inference from MML. This suggests that the theory which leads to the shortest two-part message is the best inference from the data. It has also been proved that both SMML and MML87 are invariant [207, Section 3.4, 5.2][218, 214, 40, 58].

The following sections of this chapter are organized in the following way. Firstly, the theoretical foundation of MML theory, the Strict Minimum Message Length (SMML) principle is reviewed. The discussion includes mathematical derivation of the optimal relations and key properties of SMML. The subsequent section is devoted to various techniques to approximate SMML. These approximations are important for implementing MML inference in practical statistical inference problems. Lastly, a comparison of MML inference with other statistical inference schemes and a brief review of MML literature are presented.

2.2 Information and Coding

Information is often defined as knowledge of specific events or objects. Gaining information is equivalent to reducing uncertainty about the events and objects. From this point of view, measurement of information must be examined in the context of communication, i.e., information is transmitted from a sender to receiver.

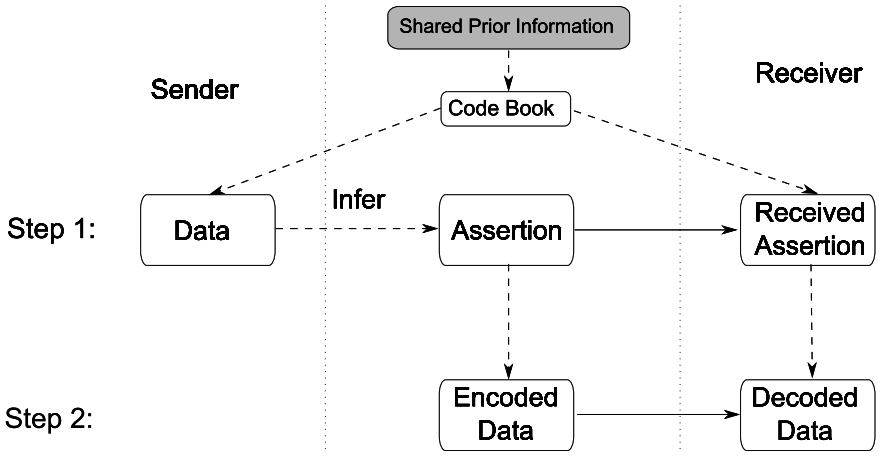


Figure 2.1: A hypothetical communication scenario

Shannon and Weaver [176] defined the information content, I, of an event that has probability p of occurring as $I = -\log(p)$. Most information is recorded and conveyed as sequences of symbols, which are often known as a message. The length of message can actually be used as a measurement of information. It is revealed by the coding theory discussed below.

Encoding is a procedure for mapping a given set of symbols $\{s_1, s_2, \ldots\}$ onto a set of code words $\{c_1, c_2, \ldots\}$. There are coding schemes which perform one-to-many or many-to-one transformations. However, for investigations here, coding schemes are confined to one-to-one transformations. Only the total code length of the transformation is of interest and how to develop coding schemes for data sets sampled from various statistical distributions.

It has been proved that for a finite set of events $\{x_i\}$ and a set of associated probabilities $\{p(x_i)\}$, there exist optimal codes which would minimise the expected code length. One such optimal coding scheme is to encode each x_i with a code length of $-\log_2(p(x_i))$ bits [175]. It is identical to the definition of the information content. As such, code lengths are often regarded as measures of information.

Coding theory is not only limited to applications of encoding data sets. It can also be applied to problems of encoding possible theories. The choice of a code for theory description implicitly specifies a prior probability distribution over the set of possible hypotheses, Θ, which is $Pr(\theta_i) = 2^{-length\_of\_code\_for\_\theta_i}$. There are some conditions on constructing a proper prior, such as $\sum Pr(\theta_i) < 1$, $Pr(\theta_i) = 0$ for impossible theories and $Pr(\theta_i) > 0$ for possible theories.

The information content of a given body of data is subjected to prior expectations shared by the sender and the receiver. To devise an optimal coding scheme for the data, it is important to clarify the prior information shared by both the sender and the receiver. The optimal coding scheme, which is often known as the optimal code, is determined and agreed upon by both the sender and the receiver before the procedure of encoding and transmitting data can begin.

2.3 Strict Minimum Message Length (SMML)

The main concern of MML inference is to construct a short two-part message containing an explanation for the given body of data. (Recall the hypothetical communication scenario in section 2.1.) In fact, obtaining accurate estimations of the length of explanations is more important than performing an actual encoding of the data for the purpose of this investigation. The estimator functions, which are able to exactly minimize the expected length of explanations, are called *Strict Minimum Message Length* (SMML) estimators [207, Chapter 3][76, 214, 40, 58]. The SMML method lays the theoretical foundations of the framework of MML inference. This section is devoted to an investigation of the characterization of SMML and the derivation of certain relations which must be satisfied by SMML estimators. Although for most statistical distributions SMML is impractical for calculating the exact length of an explanation message from given data sets, the study and analysis of the SMML method are crucial for the development of the more general (and practical) MML estimators, which themselves are approximations of SMML in various ways.

2.3.1 I_0 and I_1

There are two ways to convey a body of data. It can either be transmitted directly (as it is in a one-part code); or alternatively a theory can be inferred from the data, then the set of data is transmitted with the help of the theory (in a two-part code). The expected length of a one-part code is often denoted as I_0 ($I_0 = \sum_x r(x) I_0(x)$, where $I_0(x) = -\log r(x), r(x) = \int_{\theta \in \Theta} h(\theta) f(x|\theta) d\theta$), while the expected length of a two-part code is denoted as I_1 ($I_1 = \sum_x r(x) I_1(x)$).

Assuming that the parameter θ is a one dimensional vector and the target point estimate θ lies in a range $(\hat{\theta} - \Delta/2, \hat{\theta} + \Delta/2)$, such that $\theta \in (\hat{\theta} - \Delta/2, \hat{\theta} + \Delta/2)$, then the probability of obtaining a datum x is given by

$$Pr(x|\hat{\theta} \pm \Delta/2) = \frac{\int_{\hat{\theta}-\Delta/2}^{\hat{\theta}+\Delta/2} f(x|\theta)h(\theta)d\theta}{\int_{\hat{\theta}-\Delta/2}^{\hat{\theta}+\Delta/2} h(\theta)d\theta}$$

If the value of Δ is increased until the range $(\hat{\theta} - \Delta/2, \hat{\theta} + \Delta/2)$ spans the whole space of Θ, the probability $Pr(x|\hat{\theta} \pm \Delta/2)$ becomes the marginal probability of x, denoted as r(x), which is the integral of the likelihood function over the whole parameter space and thus given by

$$r(x) = \int_{\theta \in \Theta} f(x|\theta)h(\theta)d\theta$$

Then the length of the optimal code for encoding an element $x \in X, r(x) > 0$ is given by,

$$I_0(x) = -\log r(x)$$

The expected one-part message length for the set X using the one-part optimal code is

$$\begin{aligned} I_0 &= E(-\log r(x)) \\ &= -\sum_{x \in X} r(x) \log r(x) \end{aligned}$$

According to information theory, the minimum expected code length to encode x is achieved when data x is encoded with a code length of $-\log r(x)$. In such a scheme, the message only consists of encoded data without giving any estimation of the parameter θ. Such a one-part code is often called a *non-explanation code* as it does not give an assertion of a model for the data.

The motivation of constructing two-part messages by MML is to use the two-part message length to select the best point estimate for the target parameter. The first step to convert inductive inference problems into coding problems is to establish probabilistic models of possible data and inference. These models and assumptions

are similar to those developed in the Bayesian framework. Let X be the set of possible data, then the set X is regarded as a countable and discrete set in the MML framework. With $\hat{\theta} \in \Theta$ an inference from $x \in X$, $f(x|\hat{\theta})$ is defined as the probability of having data x given the assertion $\hat{\theta}$.

When the parameter θ is a D-component vector, i.e., $\theta \in R^D$, then the set Θ is a D-dimensional real vector space. Since a non-zero probability cannot be assigned to any arbitrary point in Θ, partitioning of the set Θ is required in order to encode an estimated value of $\hat{\theta}$ for the inferred model. In practice, there are a few approaches to convey an estimation $\hat{\theta}$ used in MML. One, which is implemented in and often known as MML87 approximation [218], is to encode the estimated value of $\hat{\theta}$ to some finite precision Δ. A second, which is due to Dowe and known as the $I1D$ approximation (or MMLD), is to encode a defined area in Θ [49][207, Chapter 4][86, 87, 40]. Both approximations use prior density to calculate the coding probability to the model. With the same assumptions used in calculating I_0, taking the first approach and the parameter θ as a one dimensional vector, the length of the two-part message for coding x, which consists of the message to code the assertion and the message to encode the data x given the assertion, is given by

$$I_1(x) \quad = \quad -\log Pr(\hat{\theta} \pm \Delta/2) - \log Pr(x|\hat{\theta} \pm \Delta/2)$$

The expected two-part message length for the set X using the two-part optimal code, I_1, is given by

$$I_1 = -\sum_{x \in X} r(x) \Big(-\log Pr(\hat{\theta} \pm \Delta/2) - \log Pr(x|\hat{\theta} \pm \Delta/2) \Big)$$

Usually, a non-explanation code is of little interest for inductive inference. However, I_0 is the most efficient code for encoding a body of data and gives a lower bound for the expected code length of the MML two-part message I_1. On the other hand, a MML explanation containing an assertion $\hat{\theta}$ and a body of encoded data in light of the assertion, is less efficient.

When the set of possible models Θ are discrete, the difference between I_1 and I_0 is $I_1 - I_0$, which is given by,

$$I_1 - I_0 = -\sum_{x \in X} \log \frac{Pr(\hat{\theta}) f(x|\hat{\theta})}{r(x)}$$

$$I_1(x) - I_0(x) = -\log \frac{Pr(\hat{\theta}) f(x|\hat{\theta})}{r(x)}$$

is identical to the negative log of the Bayesian posterior probability. Therefore the difference in length between two explanation messages associated with two distinct hypotheses identifies the relative posterior probabilities of the two hypotheses. Assuming that the log has natural base, then the relation is given by

$$\frac{Pr(\theta_i|x)}{Pr(\theta_j|x)} = e^{I_1(x|\theta_i) - I_1(x|\theta_j)}$$

However, if parameter θ is continuous, then no non-zero probability can be assigned to $Pr(\hat{\theta})$, the above statement is not valid. In this case, the expectation of $I_1 - I_0$ is a good measure of the "believability" of the estimates.

2.3.2 The Minimum Length of SMML Explanations

The SMML method encodes data $x_i \in X$ into a two-part message, with the assertion as the first part of the message and the detail of the data given by the assertion as the second part. The first step of the SMML method is to construct subsets which consist of countable point estimates from the parameter space Θ. Let $\Theta^* = \{\theta_1, \theta_2, \theta_3, \ldots\}$ be one of these subsets, then assume that a prior probability distribution $\{Pr(\theta_j)\}$ is defined over Θ^*. With $\sum Pr(\theta_j) = 1$, an optimal code encodes an assertion θ_j into a string with the length $-\log Pr(\theta_j)$. Given an assertion $\hat{\theta} = \theta_j$, the code length for encoding data $x_i \in X$ is $-\log f(x_i|\theta_j)$. The core of SMML is an "estimator function" m() which maps data $x_i \in X$ into one of $\hat{\theta} \in \Theta^*$, so that the code length

of the two-part message for encoding x_i is minimized, i.e.,

$$\begin{aligned} -\log Pr(m(x_i)) - \log f(x_i|m(x_i)) \quad &\leq \quad -\log Pr(\theta_j) - \log f(x_i|\theta_j) \\ &\forall j : \theta_j \in \Theta^*, \theta_j \neq m(x_i) \end{aligned}$$

The set Θ^* and the estimator function m(x) are regarded as prior information and agreed upon by both the sender and the receiver before any data is transmitted. The sender and the receiver deduce this "coding prior" (see [58]) from the original prior and likelihood [58]. How this is done is shown below. The goal of the investigation is to find the minimum length of SMML explanations and the conditions under which such minimum length is achieved. Given a set of data X, prior $h(\theta)$ and likelihood function $f(x|\theta)$, which are assumptions found in general Bayesian frameworks, how can the SMML estimator and the set of point estimates for common statistical distributions be constructed? It is easily observed that mapping X into Θ^* by the estimator function m() implies partitioning in X. If such a partitioning has partitions given by $t_j = \{x : m(x) = \theta_j\}$, where t_j is the set of data which are mapped into assertion θ_j by m(x), then the expected length of a two-part message for x, I_1, is given by

$$\begin{aligned} I_1 &= -\sum_{x_i \in X} r(x_i)[\log Pr(m(x_i)) + \log f(x_i|m(x_i))] \\ &= -\sum_{\theta_j \in \Theta^*} \sum_{x_i \in t_j} r(x_i)[\log Pr(\theta_j) + \log f(x_i|\theta_j)] \\ &= -\sum_{\theta_j \in \Theta^*} \left(\sum_{x_i \in t_j} r(x_i) \right) \log Pr(\theta_j) - \sum_{\theta_j \in \Theta^*} \sum_{x_i \in t_j} r(x_i) \log f(x_i|\theta_j) \end{aligned} \tag{2.1}$$

Transforming I_1 into the above form leads to finding out relations which must be satisfied to achieve the minimum length of the two-part SMML explanation [207, Section 3.2.2]:

- **R1:** The first relation is obvious, it is the definition of the estimator function $m()$ itself. For any given data $x_i \in X$, the estimator $m()$ must select a point estimate $\hat{\theta} \in \Theta^*$ which encodes x_i into a two-part explanation with the

minimum length. In effect, $m(x_i)$ selects the point estimate $\hat{\theta} \in \Theta^*$ with the highest posterior probability.

$$\begin{array}{ccc} -\log Pr(\hat{\theta})) - \log f(x_i|\hat{\theta}) & \leq & -\log Pr(\theta_j) - \log f(x_i|\theta_j) \\ & \forall i : x_i \in X & \forall j : \theta_j \in \Theta^*, \theta_j \neq \hat{\theta}, \hat{\theta} = m(x_i) \end{array}$$

- **R2:** The second relation is derived by minimising the first term in expression 2.1. By information theory, Θ^* and the distribution $\{Pr(\theta_j)\}$ are chosen so that the coding probability assigned to assertion θ_j is the sum of the marginal probabilities of data belonging to the partition t_j mapped to the assertion θ_j,

$$Pr(\theta_j) = \sum_{x_i \in t_j} r(x_i)$$

- **R3:** Similarly, the third relation is derived by minimising the second term of the expression 2.1. The choice of Θ^* must also minimise $-\sum_{x_i \in t_j} r(x_i) \log f(x_i|\theta_j)$

The SMML estimator is a general statistical estimator rooted in the Bayesian framework which also possesses the following properties [207, Section 3.4]:

- Data Representation Invariance [207, Sec 3.4.1]: The length of the two-part message, I_1, and the assertion given by the estimator m() remain unchanged when the data is transformed by a one-to-one function, i.e., if $y = u(x)$, then $m_y(u(x)) = m(x)$.

- Model Representation Invariance: I_1 and the assertion given by the estimator m() remain unchanged when the model under one-to-one transformation, i.e., if $\phi = g(\theta)$, then $m_\phi(x) = m(g(x))$.

- Generality: The SMML method can be applied to a wide range of statistical inference problems.

- Dependence on Sufficient Statistics: If x can be presented as y via y=y(x) and y contain contains all the information of data x relevant to θ, then the SMML estimator depends only on y.

- Efficiency: Given a finite and noisy data set, SMML attains the best explicit separation of pattern and noise information with the coding method utilised in SMML.

The SMML method is capable of constructing the shortest two-part explanation for a body of data and possesses many desirable properties for statistical estimators (although the SMML estimator typically prefers the models of highest available orders, Dowe [49, footnote 153] explained why this happens and gave a modified format to rectify the issue). However, the SMML method is less desirable for most practical statistical inference tasks due to the following reasons.

Firstly, SMML estimators are difficult to construct. Under the SMML framework, the best inference from a body of data is the estimate resulting in the shortest two-part explanation. In general, the set of point estimates drawn from a body of data contains multiple distinct point estimates, each of which is associated with a partition t_i in the data space X. The partitions t_i are complementary and not overlapping, i.e., $X = \bigcup t_i$. But, unfortunately, satisfying the above relations is not a sufficient condition to define the SMML estimator [207, Section 3.2.2]. In fact, given only the likelihood function, a Bayesian prior and following the rules R1, R2 and R3, the construction of the SMML estimator is NP-hard for most distributions. So far, SMML estimators have been successfully constructed for the binomial distribution, the trinomial distribution [210][207, sec 3.2][75][76], mean of multivariate normal distribution with known variance[207, sec 3.3.3][11, Page 33] and a simple binomial change-point problem [83, Chapter 5][86], survey in [49], see also [60]. More recently, Dowty [71] gave a method for calculating the SMML estimator for 1-dimensional exponential families with continuous sufficient statistics.

Secondly, SMML estimators divide the data space into a set of groups and give each group of data one distinct point estimate. As a result, two adjacent data points in X may have different point estimates. Therefore the resulting parameter estimate functions are discontinuous functions over the data. For practical purposes, one

would prefer a continuous estimation function and single point estimate returned by the estimator.

For these reasons, the Strict Minimum Message Length (SMML) estimator in general has not been applied directly to solve most practical inference problems. However, the SMML method establishes the theoretical foundation of the minimum message length framework. It also provides a starting point for pursuing more practical message length approximations. The five properties (data representation invariance, model representation invariance, generality, dependence on sufficient statistics, efficiency) possessed by the SMML estimator and the subsequent approximations give MML methods significant advantages compared to other statistical estimators. In the next section, three approximations to the SMML method are presented. These SMML-like estimators retain most merits of the SMML estimators, but are mathematically more feasible to construct.

2.4 Approximations to SMML

This section presents three "SMML-like" estimators, MMLA, Ideal Data Group and (MMLD or) I1D, which are approximations to SMML in various ways. Rather than seeking the absolute minimum of two-part message length, the aim of these approximations is to return smoother estimator functions with some approximation errors in calculating two-part messages than those returned by the SMML estimator. As such, they not only avoid the difficulties of constructing complementary partitions in the data space, but also produce smoother point estimate functions and message length functions on the data. MMLA builds the entire code book before seeing any data, and thus demands a construction of complementary partitions in the parameter space. The other two are only concerned with the area in which the estimated value of the parameter most likely lie, and thus do not require such partitions.

2.4.1 MML Approximation A (MMLA)

The Minimum Message Length Approximation (MMLA) is an approximation to SMML and was due to Wallace [207, Section 4.1.1]. Similar to the SMML method,

MMLA defines a set of point estimates $\Theta^* = \{\theta_1, \theta_2, \ldots\}$. However, MMLA does not construct a set of partitions in data space X (like SMML does), each of which is mapped to one $\theta_j \in \Theta^*$. Instead, MMLA attempts to define a set of partitions in the parameter space, $\{R(\theta_j) \in \Theta\}$, each of which is associated with a point estimate in $\theta_j \in \Theta^*$. These partitions $R(\theta_j)$, like the data groups t_j defined in SMML method, are complementary and not overlapping, i.e., $\Theta = \bigcup R(\theta_j)$. The prior probability distribution $\{Pr(\theta_j)\}$ is no longer calculated by the sum of the marginal probability of the data in a set of predefined data groups, but are given by integrating the prior probability density $h(\theta)$ over the region $R(\theta_j)$,

$$Pr(\theta_j) = \int_{R(\theta_j)} h(\theta) d\theta$$

In this way, the prior probability distribution $\{Pr(\theta_j)\}$ is determined by the area of the partition set $\{R(\theta_j)\}$. In the SMML method, the estimator function m(x) gives each data $x \in X$ a point estimate $\theta_j \in \Theta^*$. There is also no such estimator function defined in the MMLA method. For a given data x and in each region $R(\theta_j)$, MMLA encodes data x with assertion θ_j and then average the code length over the region $R(\theta_j)$. The MMLA estimator does not give a point estimate for a given datum x, but instead attempts to minimize the expected two-part message length over the set $\{R(\theta_j) \in \Theta\}$ for the whole data set X.

Therefore, the goal of MMLA estimators is to construct the codebook, i.e., a set of point estimates Θ^*, and find the optimal partition of the parameter space, i.e., a set of regions associated with each member θ_j in Θ^*, to minimize the expected two-part message length which is given by

$$\begin{aligned} I_{1A} &= -\sum_{\theta_j \in \Theta^*} Pr(\theta_j) \left(\log Pr(\theta_j) + \frac{1}{Pr(\theta_j)} \int_{R(\theta_j)} h(\theta) \sum_{x \in X} f(x|\theta) \log f(x|\theta_j) d\theta \right) \\ &= -\sum_{\theta_j \in \Theta^*} Pr(\theta_j) \log Pr(\theta_j) - \sum_{\theta_j \in \Theta^*} \int_{R(\theta_j)} h(\theta) \sum_{x \in X} f(x|\theta) \log f(x|\theta_j) d\theta \end{aligned}$$

I_{1A} gives an expected two-part message length averaged over the entire parameter space via a set of complementary regions. In each region, MMLA calculates the average code length over the entire data set X. In order to find the relations which lead to the optimal Θ^* and the associated partition of the parameter space for the MMLA method, a new expression called I'_{1A}, which is equivalent to I_{1A} plus a constant term, $\int\limits_{\Theta} h(\theta) \sum\limits_{x \in X} f(x|\theta) \log f(x|\theta) d\theta$, is minimised. It is obvious that the set of point estimates Θ^* and the set of regions associated with the Θ^* that minimise I'_{1A} also minimise the target two-part message length I_{1A}. The message length I'_{1A} is then given by

$$\begin{aligned} I'_{1A} &= I_{1A} + \sum_{\theta_j \in \Theta^*} \int\limits_{R(\theta_j)} h(\theta) \sum_{x \in X} f(x|\theta) \log f(x|\theta) d\theta \\ &= -\sum_{\theta_j \in \Theta^*} Pr(\theta_j) \log Pr(\theta_j) \\ &\quad + \sum_{\theta_j \in \Theta^*} \int\limits_{R(\theta_j)} h(\theta) \sum_{x \in X} f(x|\theta) \log \frac{f(x|\theta)}{f(x|\theta_j)} d\theta \end{aligned} \tag{2.2}$$

$\sum\limits_{x \in X} f(x|\theta) \log \frac{f(x|\theta)}{f(x|\theta_j)}$ is the Kullback-Leibler distance from the distribution $\{f(x|\theta)\}$ to the distribution $\{f(x|\theta_j)\}$, which are the likelihood functions given by the candidate model θ and the point estimate θ_j, respectively. To simplify the notation, let $KL(\theta, \theta_j) = \sum\limits_{x \in X} f(x|\theta) \log \frac{f(x|\theta)}{f(x|\theta_j)}$, then the equation 2.2 can be rewritten as

$$I'_{1A} = -\sum_{\theta_j \in \Theta^*} Pr(\theta_j) \log Pr(\theta_j) + \sum_{\theta_j \in \Theta^*} \int\limits_{R(\theta_j)} h(\theta) KL(\theta, \theta_j) d\theta \tag{2.3}$$

$$= -\sum_{\theta_j \in \Theta^*} Pr(\theta_j) \left(\log Pr(\theta_j) + \frac{\int\limits_{R(\theta_j)} h(\theta) KL(\theta, \theta_j) d\theta}{Pr(\theta_j)} \right) \tag{2.4}$$

However from 2.4, the relations which minimise I'_{1A} cannot be derived. Observing that the contribution from the region $R(\theta_j)$ is independent from the other regions

associated with any given $\hat{\theta} \in \Theta^*$, the conditions are sought which minimise the contribution from the region associated with θ_j. It has been pointed out [218, pp 263-264] and proved [83, Appendix A] that the optimal region $R(\theta_j)$ which minimises the expression,

$$-\log Pr(\theta_j) - \frac{\int\limits_{R(\theta_j)} h(\theta) KL(\theta, \theta_j) d\theta}{Pr(\theta_j)}$$

possesses the following property

$$\theta \in R(\theta_j) \iff KL(\theta, \theta_j) \leq \frac{\int\limits_{R(\theta_j)} h(\theta) KL(\theta, \theta_j) d\theta}{\int\limits_{R(\theta_j)} h(\theta) d\theta} + 1$$

The relation is also known as the MMLA boundary rule, which states that a point $\theta \in \Theta$ belongs to the region $R(\theta_j)$ if and only if the Kullback-Leibler distance from the distribution $\{f(x|\theta)\}$ to the distribution $\{f(x|\theta_j)\}$, which are the likelihood functions given by the candidate model θ and the point estimation θ_j, is less than or equal to the average prior-weighted Kullback-Leibler distance over the region $R(\theta_j)$ plus 1 nit, where nit is the natural digit, defined as 1 nit = $\log_2 e$ bits [49, sec. 0.2.3, p531, col. 1]. By the MMLA boundary rule, the optimal region $R(\theta_j)$ associated with a given point estimate $\theta_j \in \Theta^*$ can be defined. Although the problem of finding a "local" (i.e., for a single region) optimal solution has been solved, derivation of the relations which minimise the "global" two-part message length I'_{1A} has not.

MMLA attempts to obtain an approximation with a simpler code construction and more obvious relation to prior beliefs than SMML. The effort is partly successful. Compared to the SMML method, MMLA eliminates the need to construct the SMML estimator function which maps data x into one of the point estimates in Θ^*. Furthermore, MMLA simplifies the estimation of message length by avoiding the calculation of the marginal probability r(x), which is difficult to obtain in most cases. On the other hand, MMLA does not give a point estimate for a given body of data. However, MMLA is still not a suitable candidate for solving most practical statistical inference problems. To seek the optimal solution via MMLA, one still has

to rely heavily on searches to find the number and the values of the point estimates $\{\theta_j\}$ and the associated boundaries of the regions [11, Appendix E].

Although it was no longer considered to be advantageous by Wallace [207, Section 4.1.1], MMLA serves as an intermediate step towards other important approximations of SMML. In fact, MMLA can be regarded as a generalised version of the other two approximations of SMML, Dowe's I_{1D} approximation and MML87, which are more practical and will be discussed in the following subsections. The MMLA approximation has also been discussed in [11, Page 23] (in which MMLA was referred to as Fairly Strict MML, FSMML), [1, sec. 3.3.4] and [83, sec. 2.2.2].

2.4.2 Dowe's "Ideal Group" (IG) Estimator and I_{1D} Approximation

Dowe proposed the two following approximations to SMML. These approximations are discussed in detail in [207, chapter 4]. Both approximations avoid the need to construct a partition of the entire set X of possible data.

The first one is called "Ideal Group" (IG) Estimator. For a given data $x \in X$, the IG estimator selects a point estimate $\hat{\theta}$ and a group of data t, which is called the ideal data group. The boundary of the ideal data group is defined as the following. For the set X of possible data and an estimation of parameter $\hat{\theta}$, a member x belongs to an ideal data group t if the following rule is satisfied: the difference between the length of the two-part message $I_1(x|\hat{\theta})$ and the length of the optimal non-estimating one-part message $I_0(x)$ is less than the average difference of $I_1(x|\hat{\theta}) - I_0(x)$ of all data x belonging to the group t plus 1.

$$x \in t \iff \log r(x) - \log f(x|\hat{\theta}) < \frac{1}{\sum\limits_{y \in t} r(y)} \left(\sum_{y \in t} r(y) \left(\log r(y) - \log f(y|\hat{\theta}) \right) \right) + 1$$

The IG estimator then encodes x with $\hat{\theta}$, the length of the two-part message for encoding x is thus given by $DI_1(x|\hat{\theta}) = -\log Pr(\hat{\theta}) - \log f(x|\hat{\theta})$, where $Pr(\hat{\theta}) = \sum\limits_{y \in t} r(y)$. The prior (and coding) probability for $\hat{\theta}$, $Pr(\hat{\theta})$, is approximated by the

sum of the marginal probability of the data in the ideal data Group t. The choice of $\hat{\theta}$ and the group t should minimise $DI_1(x|\hat{\theta})$.

The IG estimator eliminates the need to construct a set of complementary partitions in the data space X. Within an ideal data group, the IG estimator returns a smoother estimator function (smaller variance of $I_1(x|\hat{\theta}) - I_0(x)$) compared to the SMML estimator function. The IG estimator can be applied to approximate the optimal message length for x given by SMML. However, the above $DI_1(x|\hat{\theta})$ cannot be used to calculate the expected two-part message length for the set X. As for each x$\in X$, the IG estimator tends to "cherry pick", i.e., select with a strong bias towards the point estimate $\hat{\theta}$ an ideal data group t only optimal for encoding x. While SMML estimators only possess a limited number of point estimates Θ^*, most data will be encoded with a $\hat{\theta} \in \Theta^*$ optimal for a group of data but not for each specific one. As such, the true SMML message length is in general longer than that given by IG estimators.

Wallace suggested a correction [207, sec. 4.1][1, sec. 3.3.3]][49, sec. 0.2.2, p529, col. 1]. The rectified $DI_1()$ is given by

$$DI_1(x|\hat{\theta}) = \frac{1}{\sum\limits_{y \in t} r(y)} \left(\sum_{y \in t} r(y) \left(\log r(y) - \log f(y|\hat{\theta}) \right) \right) + \log \frac{f(x|\hat{\theta})}{r(x)}$$

The Ideal Group estimator has been successfully applied and has given good results to difficult statistical inference problems such as the Neyman-Scott problem [207, sec 4.2][213, 132].

Another approximation to SMML proposed by Dowe is the $I1D$ approximation [207, sec. 4.10], which was also known as the MMLD [49, sec. 0.2.2][51, sec. 6.3][86, 1] approximation. For data x, the $I1D$ approximation is a simple method of approximating the two-part message length $I_1(x)$ given by an SMML code. Given data x, a prior $h(\theta)$ and a defined parameter space Θ, the $I1D$ approximation encodes x in a two-part explanation in the following way. A region $R \in \Theta$ is chosen as the "coding region" , in a sense that every $\hat{\theta} \in R$ would be employed to code x in the second part message. The prior probability and the coding probability of the

region R is calculated by integrating the prior density $h(\theta)$ over R, thus

$$Pr(R) = \int_R h(\theta) d\theta$$

and the length of first part message (for coding the region R) is given by $-\log(Pr(R))$. As every $\hat{\theta} \in R$ would be employed to code x, therefore only the expected value of the length of the second part message is given by

$$-\frac{1}{Pr(R)} \int_R h(\theta) \log f(x|\theta) d\theta$$

Therefore, the length of the two-part explanation $I_1(x)$ estimated by the $I1D$ approximation is

$$I_1(x) = -\log Pr(R) - \frac{1}{Pr(R)} \int_R h(\theta) \log f(x|\theta) d\theta$$

The aim of the $I1D$ approximation is to search for a region $R \in \Theta$, which minimises the $I_1(X)$ given by above equation. It has been shown that the minimum expected message length is achieved when the following rule is satisfied [207, sec 4.10.2].

$$\theta \in R \iff -\log f(x|\theta) \leq -\frac{\int_R h(\phi) \log f(x|\phi) d\phi}{\int_R h(\phi) d\phi} + 1$$

Namely, model θ belongs to the region R if and only if the negative log-likelihood given x is less than the average negative log-likelihood within the region minus one.

The $I1D$ approximation is only concerned with one region in parameter space. In this respect, the $I1D$ approximation is similar to the IG estimator, which concentrates on one region in data space. On the other hand, the $I1D$ approximation can be regarded as a direct approximation of MMLA, which seeks complimentary partitions in the model space. The three discussed approximations (MMLA, IG estimator and $I1D$) have played important roles in developing and comprehending other more practical SMML approximations. Many ideas and concepts introduced

in these approximations proved fruitful and constructive. They are of great value to research on and development of the MML principle. One among them is the concept of the "uncertainty region" in the parameter space. In general, selecting a larger uncertainty region results in a shorter code length of assertion but a longer code length of encoded data. Thus finding the minimum two-part message for a set of data is equivalent to seeking the best trade-off between model complexity and fitting the data. In the next section one more approximation to SMML, MML87, which has been successfully applied to solve a wide range of practical statistical inference problems is introduced.

2.4.3 MML87: Quadratic Approximation to SMML

SMML and the approximations discussed in the previous sections are not mathematically flexible enough to be applied to most ordinary statistical or inductive inference problems. Among the approximations discussed in the previous section, Dowe's I1D approximation works well numerically at a cost of high computational complexity. However, to apply MML inference on real-world data sets, there is a need to find a SMML approximation that is able to derive estimations on the minimum two-part message length analytically for common distributions. The MML87 approximation was due to Wallace and Freeman [218], which is also a refined (or generalised) version of the MML68 approximation published in an earlier paper by Wallace and Boulton [209].

In order to simplify the calculation, the MML87 approximation does not attempt to construct estimators which minimise the expected message length for all possible values of X. Instead, only the presented data is taken into consideration. MML87 also assumes that the uncertainty area in which the target point estimate lies is a single symmetric contiguous region in (multiple) continuous parameter spaces.

To calculate the prior (and coding) probability for $\hat{\theta}$, integrations of the probability density function over the range is usually needed. However, to simplify the calculation, MML87 assumes that the value of the parameter θ does not vary too much at $\hat{\theta}$ and the "real" value of the estimation θ' lies within a single symmetric contiguous region in (multiple) continuous parameter spaces, whose centre is

the point estimate $\hat{\theta}$. With these assumptions, the probability $Pr(\theta')$ could be approximated by multiplying the density and the length(s) of the hypercube in the (multiple) parameter spaces. For convenience of the illustration and not losing the generality, let θ be a one-dimensional parameter and the parameter space Θ span a finite range in the real line. The prior probability is $h(\theta)$. Assuming the target estimate $\theta' \in (\hat{\theta} - \frac{w(\hat{\theta})}{2}, \hat{\theta} + \frac{w(\hat{\theta})}{2})$, θ' can be encoded by constructing an optimal code for the probability distribution $\{Pr(\theta_j)\}$.

Similar to MMLA, a set of point estimates Θ^* is agreed upon by both sender and receiver before any data is sent. Suppose $\hat{\theta} \in \Theta^*$ is the closest member to θ', a data x is then conveyed by using $\hat{\theta}$ as the point estimate. Then the coding probability for $\hat{\theta}$, $Pr(\hat{\theta})$, could be approximated by

$$Pr(\hat{\theta}) = \int_{\hat{\theta}-\frac{w(\hat{\theta})}{2}}^{\hat{\theta}+\frac{w(\hat{\theta})}{2}} h(\theta)d\theta \approx h(\hat{\theta})w(\hat{\theta}) \approx h(\theta')w(\theta')$$

$$\forall \theta' \in (\hat{\theta} - \frac{w(\hat{\theta})}{2}, \hat{\theta} + \frac{w(\hat{\theta})}{2})$$

The function w() is called the spacing function, whose value indicates the precision that the point estimate $\hat{\theta} \in \Theta^*$ would be encoded with. The sender should choose the optimal value of $w(\hat{\theta})$ so that the two-part message length $I_1(x)$ is minimised. Given that $I_1(x) = -\log(w(\hat{\theta})h(\hat{\theta})) - \log f(x|\hat{\theta})$ and using the Taylor expansion, gives

$$\begin{aligned} I_1(x) &= -\log(w(\hat{\theta})h(\hat{\theta})) - \log f(x|\hat{\theta}) \\ &\approx -\log(w(\theta')h(\theta')) - \log f(x|\theta') \\ &\quad -(\theta-\theta')\frac{\partial}{\partial\theta'}\log f(x|\theta') - \frac{1}{2}(\theta-\theta')^2\frac{\partial^2}{(\partial\theta')^2}\log f(x|\theta') + O((\theta-\theta')^3) \end{aligned}$$

As the target estimation θ' can take any value between $\hat{\theta} - \frac{w(\hat{\theta})}{2}$ and $\hat{\theta} + \frac{w(\hat{\theta})}{2}$, so the sender should try to minimise the expectation of the two-part message length $I_1(x)$, omitting the terms which are higher than quadratic terms, $O((\theta-\theta')^3)$, giving

$$\begin{aligned} E(I_1(x)) &\approx -\log\left(w(\theta')h(\theta')\right) - \log f(x|\theta') \\ &\quad -E((\theta-\theta'))\frac{\partial}{\partial\theta'}\log f(x|\theta') - \frac{1}{2}E((\theta-\theta')^2)\frac{\partial^2}{(\partial\theta')^2}\log f(x|\theta') \end{aligned}$$

Assuming θ' takes a uniform prior over $(\hat{\theta} - \frac{w(\hat{\theta})}{2}, \hat{\theta} + \frac{w(\hat{\theta})}{2})$, means $E(\theta - \theta^{'}) = 0$ and $E((\theta - \theta^{'})^2) = w(\theta')^2/12$, therefore the expectation of the two-part message length $I_1(x)$, $E(I_1(x))$, is given by

$$E(I_1(x)) \approx -\log\left(w(\theta')h(\theta')\right) - \log f(x|\theta') - \frac{w(\theta')^2}{24}\frac{\partial^2}{(\partial\theta')^2}\log f(x|\theta') \quad (2.5)$$

The equation 2.5 is minimised with respect to $w(\theta')$ when

$$w(\theta')^2 = -\frac{12}{\frac{\partial^2}{(\partial\theta')^2}\log f(x|\theta')} \quad (2.6)$$

However, the above expression cannot be substituted into equation 2.5 directly as the code book shared by the sender and the receiver must be decided without any knowledge of the data x. While the value of $-\frac{\partial^2}{(\partial\theta')^2}\log f(x|\theta')$ is unavailable to the receiver, its expectation over X can be determined. Let

$$F(\theta', x) = -\frac{\partial^2}{(\partial\theta')^2}\log f(x|\theta')$$

then the expectation of $F(\theta, x)$ over X is given by

$$\begin{aligned} F(\theta') &= -E\Big(F(\theta', x)\Big) \\ &= -\sum_{x\in X} f(x|\theta')\frac{\partial^2}{(\partial\theta')^2}\log f(x|\theta') \end{aligned} \quad (2.7)$$

The function $F(\theta')$ is the well known "Fisher Information", while $F(\theta', x)$ is often known as "empirical" Fisher Information. The Fisher information reveals the rate

of the variation of the negative log likelihood with respect to the estimate θ'. The larger the Fisher Information is, the sharper and narrower the peak of the curve of the negative log likelihood is. Equation 2.6 shows that the minimum message length given by MML87 is approximated by setting the value of spacing function to $\sqrt{\frac{12}{F(\theta')}}$. As such, the optimal value of the spacing function $w(\theta')$ is inversely proportional to the square root of the value of the Fisher information. An intuitive derivation from the above statement is that optimal coding precision for encoding θ' is determined by the shape of the curve of the negative log likelihood. The sharper the curve is, the higher precision is needed to encode θ'.

Choosing the appropriate θ' to minimise the expectation of the $I_1(x)$, $E(I_1(x))$ and substituting $w(\theta') = \sqrt{\frac{12}{F(\theta')}}$ into equation 2.7, gives

$$\begin{aligned} I_1(x) &\approx \arg\min_{\theta'} E(I_1(x)) \\ &\approx -\log\left(\sqrt{\frac{12}{F(\theta')}}h(\theta')\right) - \log f(x|\theta') + \frac{1}{2}\left(\frac{F(x,\theta')}{F(\theta')}\right) \end{aligned} \tag{2.8}$$

The first term and the second term of formula 2.8 give the length of assertion and the detail of data respectively. The third term could be regarded as a penalty term incurred from encoding θ' by the nearest point estimate $\hat{\theta} \in \Theta^*$.

Formula 2.8 can be further simplified by assuming that the empirical Fisher information is roughly identical to Fisher information. So let $F(x,\theta') \approx F(\theta')$, the formula becomes

$$I_1(x) \approx -\log\left(\sqrt{\frac{12}{F(\theta')}}h(\theta')\right) - \log f(x|\theta') + \frac{1}{2} \tag{2.9}$$

Formula 2.9 generalises gracefully when the parameter θ becomes a multi-dimensional real-value vector. Supposing θ is a D-dimensional

$$I_1(x) \approx -\log\left(\frac{1}{\sqrt{F(\theta')(\kappa_D)^D}}h(\theta')\right) - \log f(x|\theta') + \frac{1}{2}D \tag{2.10}$$

where κ_D is a lattice constant [102, 103] which is a result from estimating the most efficient way a D-dimensional space can be spanned.

The MML87 approximation not only greatly reduces the computational complexity of constructing the entire code book, but also retains the five desirable statistical properties (data representation invariance, model representation invariance, generality, dependence on sufficient statistics, efficiency) of SMML. It enables derivation of analytical estimation functions and formulas of the message length for many common statistical distributions. So although it has less coding efficiency compared to SMML (which it approximates), the MML87 approximation is a more suitable candidate for general method of estimation in practice.

2.5 MML Literature

This section provides a categorized list of publications in MML research.

- MML theories [207, 206, 76, 214, 215, 216, 140, 75, 139, 202, 212, 201, 200, 218, 199, 198, 91, 220, 27, 210, 22, 26, 24, 209, 58, 87, 84, 59, 142, 128, 149, 193, 58, 71]
- String alignment [4, 5, 9, 8, 7, 6, 155]
- MML model selection [170, 205, 92, 86, 85, 54, 13, 144, 3, 124, 2]
- MML inference of decision trees, graphs and forests [224, 203, 145, 147, 186, 187, 143, 141, 146, 189, 192, 191]
- MML inference of support vector machines [110, 188, 111]
- MML inference of Bayesian networks and causal models [131, 222, 109, 130, 41, 223, 65, 221, 39, 40]
- Point estimates for common statistical distributions [217, 55, 67, 211, 23, 2, 14]
- Regression [12, 37]
- Neural network [117]

- Factor Analysis [204, 219, 73]
- Solving famous statistical inference problems [69]
- Comparisons with other inference methods [25]
- Application on practical machine learning data [53, 105, 66, 119, 64, 152, 68, 157, 72, 156, 56, 106, 61, 57, 62, 230, 151, 112, 63, 158, 35, 17, 148, 36, 116]
- Opinions from other statistical learning approaches [196, 195]

2.6 Conclusion

This chapter discussed several important estimation methods in MML – SMML, MMLA, IG estimator, I1D (or MMLD) and MML87. It shows that MML goes one step beyond general Bayesian inference methods, which stops at and is content with derived posterior distributions. The MML framework has the advantages of not only retaining merits of general Bayesian inference methods, but also of possessing many desirable properties such as model invariance, which the other lacks.

MML estimation attempts to derive the most accurate assertion from a set of finite data by converting statistical inference problems into coding processes and then searching for the assertion resulting in the shortest two-part message length. However, to encode a group of data in the shortest code length is not the object of devising a MML coding scheme. Rather, construction of an efficient code book should always follow the task itself. In practice, MML and the subsequent Minimum Description Length (MDL) principle [166, 120] (see also [214] for a survey and [51, sec. 6.7] for a discussion of MML and MDL) are widely used for model selection in various machine learning problems, and both can be thought of as operational forms of Ockham's razor [128]. In the following chapters, MML coding schemes are applied to the problem of inference of decision trees, graphs and forests. The aim of these schemes is to find a trade-off between the complexity of these structure models and goodness-of-fit for a given set of data.

Chapter 3

MML Inference of Decision Trees

3.1 Introduction

Decision tree learning is a widely studied supervised learning method in machine learning research. From a given set of data – often known as the training data set – a supervised learning algorithm creates a function which maps a defined input space to some output values. There are two major purposes to inferring functions using supervised learning algorithms: the first is to extract hidden information (e.g., patterns) from the data, and the second is to predict unknown properties associated with new incoming data in future, assuming that new data and training data are identically and independently distributed (i.i.d.). The functions to be inferred by learning algorithms can take any valid form, such as decision trees, decision graphs, support vector machines, polynomial functions, Bayesian networks and neural networks, etc. Thus a supervised learning algorithm usually consists of two steps:

- Selecting the model class
- Searching for the best-fitting function with optimal parameters through the space of candidate functions

Decision tree learning uses decision trees as its model class. A decision tree learning algorithm infers a decision tree from a training data set. The popularity of

decision tree learning is due to its simplicity of learning procedures, easy implementations and yet excellent performance. Therefore, decision trees have been widely used as classifiers in many researches in other domains [150]. The intuitive nature of tree models make them well-suited for representations of underlying patterns among data.

One important goal of a decision tree learning algorithm is to find the optimal decision tree that will minimize predictive errors, i.e., generalise well on new data. As an indefinite number of trees can be generated, in practice tree learning algorithms define an objective function on candidate trees. The algorithm will then perform a limited search and the objective function is used to evaluate the candidate trees. It has been shown that the problem of finding the optimal decision tree with the smallest size is NP-complete [165]. Therefore, heuristic search algorithms such as greedy search are usually applied in order to find satisfactory trees within reasonable time.

The outline of this chapter is as follows. Section 3.2 introduces essential elements of decision tree learning problems, including a definition of classification and decision tree models (sec. 3.2.1); common procedures to infer a decision tree from data (sec. 3.2.2); split functions (sec. 3.2.3) and the problem of over-fitting (sec. 3.2.4); and a brief review of one baseline decision tree learning algorithm – C4.5 (sec. 3.2.5). Section 3.3 discusses issues of evaluating classifiers and introduces criteria to compare the performance of classifiers used in this study. Section 3.4 gives a review of MML coding methods applied in decision tree inference problems. Lastly, section 3.5 presents a novel multivariate decision tree inference scheme, which uses a MML coding scheme as an objective (goodness-of-fit) function to select multivariate splits at internal nodes. The goal is to extend MML-code based decision tree inference schemes to multivariate trees. The new objective function is capable of evaluating multivariate splits, taking generalization ability of splits into consideration.

While a novel decision tree algorithm is presented in this chapter, decision graph and decision forest learning algorithms introduced in subsequent chapters can be viewed as extensions of MML decision tree inference schemes. As such, decision tree coding schemes introduced in this chapter have also been applied in those learning algorithms.

Readers who would like to see a broader coverage of decision tree learning are advised to seek references in many extensive surveys of decision tree learning algorithms, such as [126, 127].

3.2 Decision Tree Learning

3.2.1 Classifiers and Decision Trees

Given vector x$\in R^D$, which is sampled from a set of random variables $(X_1, X_2, \ldots, X_D)$, a classifier takes the vector x as its input and outputs a value $y \in \{1, 2, \ldots, C\}$. Therefore, a classifier is a function

$$f : R^D \longmapsto \{1, 2, \ldots, C\}$$

When any random variable X_i only has a finite number of values or takes a limited range of values, the function f is restricted to a subset of X^D. In such cases,

$$f : dom(X_1) \times \ldots \times dom(X_D) \longmapsto \{1, 2, \ldots, C\}$$

Random variables X_i are often called the *independent variables* while the variable Y is called the *dependent variable* as the value of Y is determined by the classifier and the value of the vector x. In some articles, variables X_i are also known as the input attributes and variable Y is known as the target attribute.

Decision trees can be used as predictive models. When decision trees are viewed as classifiers, they take an input vector x and output a value $y \in \{1, 2, \ldots, C\}$. A decision tree is a directed acyclic graph, where each node only has in degree 1. However, in computer science literature, decision trees are often drawn in such top-down fashion, i.e., starting with a root node, using plain lines rather than arrows to connect nodes. In this way, relations between two immediately connected nodes are represented by their hierarchical level. The node situated higher in the hierarchy is called the parent node of the node situated lower in the hierarchy while the lower is called the child node of the former. There are two types of nodes in a decision tree. Nodes with child nodes are often denoted as *internal nodes* while nodes without child nodes are denoted as *leaf nodes*, whereas the node at the top of a decision tree without a parent is called the *root node*.

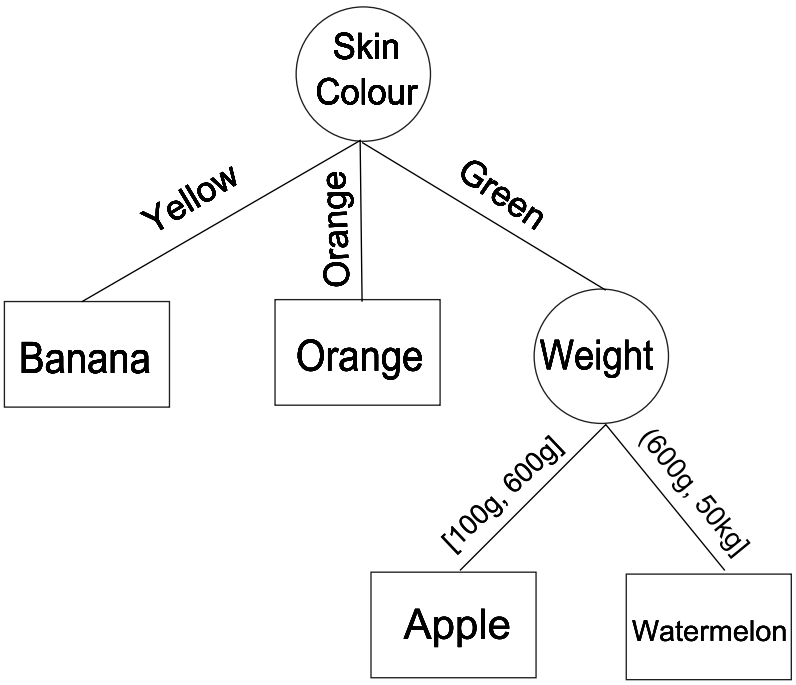


Figure 3.1: A decision tree model for classifying fruits

A function with a finite number of outcomes is associated with each of the internal nodes. When a decision tree is fed with a vector x, the case is routed down the tree from the root node until it reaches a leaf node. At each of the internal nodes it is routed through, the outgoing route is subsequently determined by the output of the function for that node, tested with the vector x as its input. A scalar $y \in \{1, 2, \ldots, C\}$ which is often known as a class label, is associated with the leaf node of a decision tree. A decision tree determines the class label associated with the leaf node which the case fell into as the class label of the test case.

A decision tree example is shown in Fig. 3.1, in which the internal nodes are represented as circles and the leaf nodes are represented as rectangles: a convention which is followed throughout this thesis.

3.2.2 Inferring a Decision Tree From Data

The problem of inferring a decision tree from a given data set can be described as follows: Given a set of training data consisting of l pairs of (x_i, y_i), where independent variable $x_i \in R^D$ and dependent variable $y_i \in \{1, 2, \ldots, C\}$ are drawn from a population with unknown probability density, the goal of a decision tree learning algorithm is to find a decision tree T that maximises (or minimises) the expected value of a pre-defined objective function $L(T(x), y)$ over the whole population, where $T(x)$ are outputs of the inferred decision tree. In practice, the performance of an inferred decision tree is measured by calculating the expected value of the objective function $L(\cdot)$ over a data set drawn separately from the population, such a score is given by

$$\frac{1}{N}\sum_{i=1}^{N} L(T(x_i), y_i)$$

where N is the size of the separately drawn data set, which is often known as the test data set. In the machine learning literature, such an inference process is often referred to as training a decision tree or growing a decision tree from a set of training data. While some algorithms infer decision trees through evolutionary algorithms [10], most top-down decision tree inference algorithms adopt the divide-and-conquer approach and are thus recursive in nature [180]. The reason is obvious – every subtree under an internal node of a decision tree is an independent decision tree itself, since it retains every property of a decision tree. As such, the same inference algorithm can be applied to these subtrees. In effect, a decision tree divides a given parameter space into a set of complementary and disjoint partitions recursively. The dividing operations continue until either every partition only contains instances (from the training set) with homogeneous values of the dependent variable or certain pre-defined conditions are satisfied. The procedure of a typical top-down decision tree inference algorithm can be summarised as immediately below.

Starting with a single leaf node (which subsequently becomes the root node of the inferred tree) and an entire training data set, the following processes are repeated until certain pre-defined conditions are satisfied in all leaf nodes.

1. Select one function class among a pre-defined set of allowable function classes (if there is more than one) as the split function
2. Perform searches on parameters for the split function, record values of the score function given by the outcome of the candidate split functions
3. Select the optimal split function by choosing the candidate parameter values corresponding to the best value of the score function
4. Generate child nodes and divide data at internal nodes in line with the outcomes of the optimal split function
5. For each of the child nodes generated in step 4, go to step 1 and repeat the above steps

Decision tree inference algorithms using the above procedure must have the following elements:

1. A set of allowable functions associated with internal nodes to perform the splits
2. A score function which evaluates candidate splits at internal nodes
3. Efficient heuristics that search for the optimal splits at internal nodes
4. A scheme for complexity control and a search strategy for finding the optimal decision tree

These elements are essential to development and analysis of decision tree inference schemes. The following subsections focus on details of these components of decision tree learning algorithms.

3.2.3 Split Functions and Split Selection

As discussed in section 3.2, there is a function associated with each of the internal nodes in a decision tree. The purpose of the function is to route an incoming datum to one of the subtrees under the internal nodes. In this way, a decision tree effectively

divides the input space into a set of adjacent partitions by routing incoming data into one of the leaf nodes. In decision tree research literature, such dividing processes are often called *splits*. As any function that takes a real vector and returns a finite number of outcomes is able to construct such partitions, there is an infinite number of candidate functions if we do not put further restrictions on the split function. To make decision tree inference problems tractable, a set of allowable functions at internal nodes is usually specified before the inference process begins. Figs 3.2, 3.3 and 3.4 show various kinds of split functions used in decision tree inference programs.

The simplest types of split functions are univariate functions, which are functions that operate on a single variable. These split functions include functions of one single variable taking category values and piece-wise functions with one single variable taking continuous values. The second type of split function consists of linear multivariate functions, which are functions of multiple variables. Decision trees with univariate split functions are called univariate trees while decision trees with multivariate split functions are often known as multivariate trees. In effect, univariate decision trees partition the data space with hyperplanes parallel to the axes where the hyperplanes for multivariate trees are not so restricted. The advantage of using linear multivariate functions as the split function class in a decision tree is that hyperplanes with arbitrary degrees of freedom are more flexible for separating the data than planes parallel to the axes. Therefore, such hyperplanes have the potential to achieve targeted data partitions with fewer cuts than axes-paralleled hyperplanes. Thus, multivariate decision trees are more expressive in the sense that they are able to represent the inferred model with more concise trees. However, using a linear multivariate split function also creates new challenges to decision tree inference algorithms: finding the optimal splits among the substantially larger set of candidate splits; controlling the increased complexity of inferred decision trees. These two crucial issues must be addressed by multivariate decision tree inference algorithms.

Some machine learning algorithms implement non-linear functions, such as Bayesian networks and support vector machines (SVMs), at internal tree nodes. Such functions, which are often capable of providing high quality of predictive performance, come with high complexity. Therefore, such decision trees are more often to be

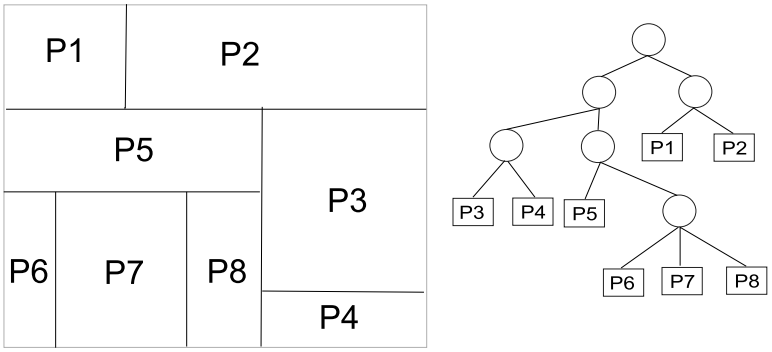


Figure 3.2: Data space partitioned by decision trees with univariate functions

regarded as hybrid models and the corresponding inference algorithms are largely different from the typical decision tree algorithms discussed in this chapter.

The choice of the function class for the split functions has a fundamental impact on the development of corresponding decision tree inference scheme. For example, algorithms restricting the split functions to predicates would only generate binary trees, whose nodes have either zero (leaf nodes) or two child nodes (internal nodes). Furthermore, the objective function for split selection which returns a merit defined on the candidate splits may have to take the split functions into considerations. Also search algorithms must be tailored for the specific function class implemented in a decision tree inference scheme. Some considerations include: a hill-climbing algorithm does not necessarily produce the best overall tree; the best tree may not have the best split in every split; and look-ahead, if introduced, may be of limited success. In general, the more expressive the split function, the simpler the tree would be. The simpler the objective function, the easier the search would be.

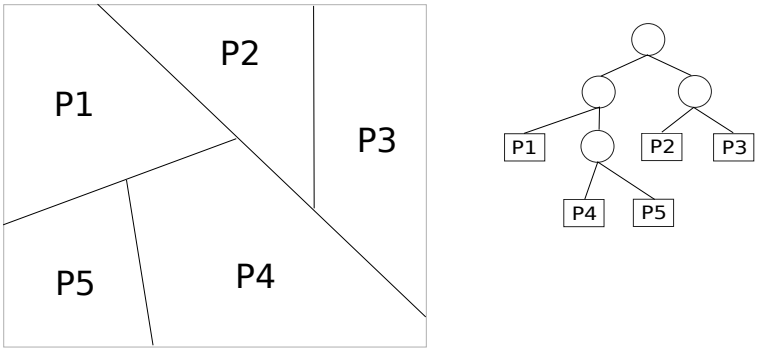


Figure 3.3: Data space partitioned by decision trees with multivariate functions

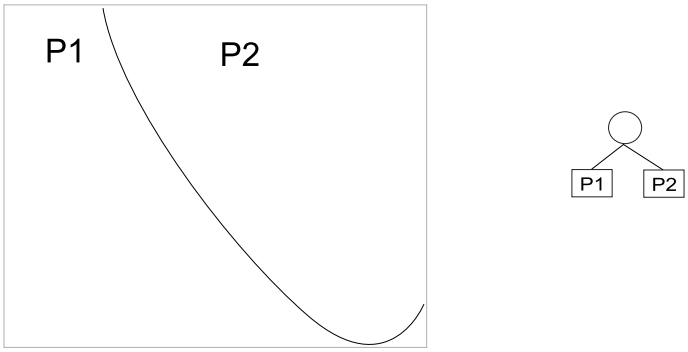


Figure 3.4: Data space partitioned by decision trees with quadratic functions

3.2.4 Over-fitting and Coding

Decision tree algorithms may be tempted to generate a decision tree that perfectly fits the training data as the optimal tree. In extreme cases, each leaf node of inferred decision trees contains only one or no instance from the training data set. Such decision trees achieve 100% predictive accuracy on the training data set. However, it is well-known that such complex decision trees often perform poorly on the incoming unseen data. There are two approaches to tackle this problem of over-fitting for top-down decision tree inference algorithms. One is to divide the tree inference procedure into two consecutive processes – tree-growing process and tree-pruning process. In the first step, tree inference algorithms grow a decision tree fully, i.e., to split nodes until every leaf node contains data belonging to one single class. Tree growing algorithms with such an approach tend to generate very complex trees. In order to simplify the resultant tree, these algorithms usually carry out a pruning process after the initial tree-growing process has been completed. The pruning process collapses certain subtrees and turns them into leaf nodes. However, the quality of the pruning process in such algorithms relies on various cross-validation schemes, such as leave-one-out, N-fold cross-validations, etc. The cross-validation schemes leave a portion of training data as a validation set, evaluate candidate trees and pick the tree with best performance on the validation set. In this way, the complexity of the inferred trees is implicitly controlled by the cross-validation schemes. The main drawback of such an approach is that part of the data has to be reserved and cannot be used to infer decision trees. For problems in which the size of data set is small, this has have adverse effects on the quality of the inferred trees. The cross-validating process also substantially increases the computational resource required by the algorithms.

Another approach is to explicitly control the complexity of the inferred trees by introducing objective functions which penalise over-complex decision trees. One way to implement such an approach is to construct coding schemes which assign longer code lengths to decision trees with complex topological structures and assign shorter code lengths to decision trees with simple topological structures. From a Bayesian point of view, having a scheme for encoding trees implicitly defines a prior probability distribution over a set of candidate trees. Unlike the former approach, it does not require any cross-validation scheme.

3.2.5 C4.5: a Decision Tree Learning Program

In this section, a brief review of the C4.5 decision tree inference program is presented. The focus is placed on several essential elements of a decision tree program: the tree growing procedure, the split selection criterion and the strategy for avoiding fitting the training data set with over-complex trees. Another reason for this is that experimental results are often compared to outputs from C4.5.

Both ID3 [161, 162] and C4.5 [164] are decision tree inference algorithms by Quinlan. While C4.5 can be regarded as a successor of ID3, they are similar in many respects.

Choosing parameters for split functions

Quinlan proposed an objective function called Gain as the split selection criterion for the ID3 decision program. Consider a set of data T with M distinct classes and a split which partitions the set T into n smaller subsets of data T_i, the Gain of such a split is defined as

$$Gain = G(T) - \sum_{i=1}^{n} \frac{|T_i|}{|T|} \times G(T_i)$$

where G() is the entropy of a data set. The entropy of a data set T with M distinct classes is defined as

$$G(T) = -\sum_{j=1}^{M} \frac{N_j}{N} \log(\frac{N_j}{N})$$

where N is the total number of instances in the set T and N_j is the number of instances belonging to the class j. Later in the C4.5 program, the Gain was replaced by another function called the Gain ratio. Both selection criteria are underpinned by information theory [175, 178, 108]. The Gain ratio is a rectified version of the Gain which eliminates serious biases towards attributes with a large number of outcomes. The Gain ratio is defined as

$$GainRatio(T) = \frac{Gain(T)}{SplitInfo(T)}$$

where

$$SplitInfo(T) = -\sum_{i=1}^{n} \frac{|T_i|}{|T|} \times \log \frac{|T_i|}{|T|}$$

The Gain ratio can then be regarded as a normalised version of the Gain.

Recently, Nowozin [136] proposed an improved information gain estimator for decision tree induction, given as

$$\hat{H}_G(h) = \log n - \frac{1}{n} \sum_{k=1}^{K} h_k G(h_k).$$

The function $G(h)$ is given as

$$G(h_k) = \psi(h_k) + \frac{1}{2}(-1)^{h_k} \left(\psi(\frac{h_k + 1}{2}) - \psi(\frac{h_k}{2}) \right).$$

In the above formula, ψ is the *digamma* function, where h_k is the occurrence of class k and $n = \sum_k h_k$. The paper [136] showed that the proposed entropy estimator produces more accurate estimates than the naive entropy estimator, especially for multinomial distributions with small sample sizes.

The procedure for constructing the C4.5 tree

Decision tree generation is naturally recursive. Starting with a single leaf node containing all the training data, the C4.5 program performs a tree growing process according to the following three rules.

- If all items at a node belong to one single class, the program stops growing subtrees under the node. The node then becomes a leaf node and is labelled.

- If no item is found at a node, the program stops and labels the leaf node with the majority class at the parent node.

- If items at a node belong to more than one class, the program searches for the optimal split function for the node, splits the node and for each of the child nodes, repeats the tree growing process.

Pruning scheme of C4.5

Quinlan developed a pruning method for ID3 called pessimistic pruning [163], which penalises trees with more leaf nodes by effectively increasing the number of errors observed at each leaf by 0.5. In C4.5, a far more pessimistic pruning scheme is adopted. After a decision tree is fully grown, i.e., no leaf nodes can be further split. Then the pruning scheme starts from the bottom of the tree. For a subtree with M leaf nodes and one internal node, the scheme calculates the predicted errors for this subtree given by

$$PE_s = \sum_{i=1}^{M} N_i * U_{CF}(E_i, N_i)$$

where N_i is the number of data at node i, and E_i is the number of data incorrectly classified at node i. Function $U_{CF}(E, N)$ represents the upper limit of (error) probability for the binomial distribution with observation (E,N) and a confidence level of CF. Then this number is compared to the predicted error for the pruned subtree given by

$$PE_p = N * U_{CF}(E, N)$$

where $E = \sum E_i$ and $N = \sum N_i$. If $PE_s > PE_p$ then the subtree is pruned back into a leaf node, otherwise the subtree is kept. The pruning process continues until no further pruning can be found.

3.3 Evaluation of Decision Trees

One essential part of any learning scheme is to assess the quality of inferred models generated by the scheme. As mentioned in section 3.2.2, the quality of the inferred model is defined as the generalisation ability of the model, i.e., how does the model perform on unseen data in the population. To compare candidate models objectively, one convention is to define loss functions with data and output of the model as its inputs. A loss function is defined as $c(x, f(x), y)$, where x is the input data to be tested, f(x) is the prediction made by the (learned) model and y is the real value of targeted variable associated with data x. In general, loss functions follow the rule which demands the output of a loss function to be zero when a correct prediction is

made:

$$c(x, y, y) = 0$$

Often the expected value of the loss function over the population is used to assess the generalisability of inferred models. Such a value is difficult to obtain. In practice, it is approximated by testing the model on a test data set independently sampled from the population. As such, the performance of an inferred model is measured by averaging outputs of the loss functions on every instance in the test data set. Depending on the goal of learning schemes, there are many choices of objective functions. This study concentrates on two criteria: classification accuracy and probabilistic prediction accuracy. In the following sections, loss functions associated with these two criteria are discussed in detail.

3.3.1 Classification Accuracy

One widely used loss function is misclassification error. The outcome of the function indicates whether the output of the model matches the real value of target variable associated with the data. The function compares the output of the model tested on incoming data and the label associated with the data. If these two values match, the function returns 0. Otherwise, the function returns 1.

$$I(x, f(x), y) = \begin{cases} 0 & \text{if } f(x) = y \\ 1 & \text{if } f(x) \neq y \end{cases} \tag{3.1}$$

Often in studies of supervised learning schemes, there is a need to evaluate and compare predictive performance of inferred models. The classification accuracy, often known as right/wrong accuracy, is a measure used to estimate the probability of an inferred model returning a correct class label for an incoming data. The classification accuracy is defined as the fraction of correctly classified data by the model on the whole test set, whose size is N.

$$1 - \frac{1}{N} \sum_{i=1}^{N} I(x, f(x), y)$$

The classification accuracy will be used as one of the criteria to compare performances of learning algorithms in this study.

3.3.2 Probabilistic Prediction Accuracy

Decision trees are often used as classifiers, which assign a class label from a predefined set of labels for a given set of values of input variables. When the target attribute is multinomial, each leaf node in a tree or graph is associated with a class label corresponding to the class with the highest inferred probability for this node. For such machine learning problems, the aim of decision tree learning algorithms is to construct an optimal decision tree or graph that maximizes the predictive accuracy. The Right/Wrong accuracy on test data is an obvious choice to evaluate the performance of the inferred decision tree or graph in this regard.

However, for many machine learning problems, like inferring from sociological and other medical data, not only the class predictions but also the probability associated with each class are essential to satisfactory solutions. In some domains, like finance, (long term) strategies heavily rely on accurate probabilistic predictions. Thus, machine learning problems with such requirements demand inferred decision trees providing optimal probabilistic prediction models.

Provost and Domingos [159] showed that with some modifications, tree induction programs can produce very high quality probability estimation trees (PETs).

In these cases, the multinomial distribution associated with each leaf node of the inferred tree or graph is interpreted as a probabilistic prediction model. Therefore decision trees and decision graphs are not only classifiers, but they can also provide a probabilistic prediction model.

Perlich, Provost and Simonoff [153] also observed that for large data sets, tree induction often produces probability-based rankings that are superior to those generated by logistic regression.

That raises the question of how to evaluate the probabilistic performances of classifiers generated from various algorithms. In the subsequent sections two most commonly adopted loss functions for this purpose in machine learning community will be discussed: Area Under Curve (AUC) and Log-Probabilistic Score (LP Score).

ROC graphs and AUC

A receiver operating characteristics (ROC) graph [74, 77] is a tool that is utilised to visualise and evaluate classifiers based on their performances. ROC graphs were originally used in signal detection theory to visualise the trade-off between hit rates and false alarm rates of classifiers [74]. Later, ROC graphs were widely used in analysis of the behaviour of diagnostic systems [97, 181].

For a classifier and a set of binary data (with positive and negative classes), the true positive rate (TPR) of a classifier is estimated by

$$TPR = \frac{\text{Positives instances correctly classified}}{\text{Total positive instances}} = \frac{TP}{TP + FN}$$

while the false postive rate (FPR) is estimated by

$$FPR = \frac{\text{Negative instances incorrectly classified}}{\text{Total negative instances}} = \frac{FP}{FP + TN}$$

A ROC graph is a two-dimensional graph in which the Y axis represents the true positive rate (TPR) and the X axis represents the false positive rate (FPR). A point in an ROC graph depicts relative trade-off between two important aspects of performance (true positives and false positives) of a classifier. Fig. 3.5 shows a ROC graph depicting the TPR and FPR of five classifiers. The point (0,1) represents the perfect classifier E while the point (1,1) represents a strategy of classifying every instance as positive, resulting in a 100% TPR and a 100% FPR. The points lying on the diagonal line $y = x$ represent classifiers with strategies of in effect randomly picking an instance as positive. For example, point (0.65, 0.65) shows a classifier B randomly guesses an instance as positive 65% of time while making 65% false negative errors. The points lying under the diagonal line (classifier A) represent performances worse than those of random classifiers. Informally, while comparing two points in a ROC graph, the point closer to the northwest corner indicates a better classification performance. In this example, classifier D performs better than classifier C as it is closer to the northwest corner. When a classifier can only output class labels (discrete numbers), then for a given test set, its performance could only be represented as a single point in a ROC graph. However, when a classifier outputs

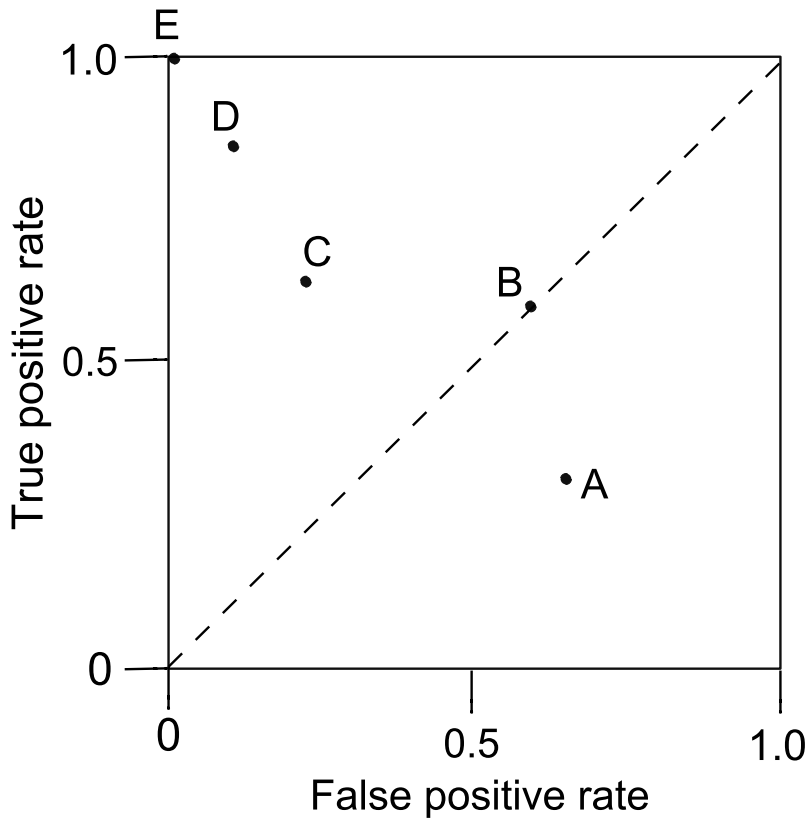


Figure 3.5: An ROC graph depicting 5 classifiers

probability estimations over class labels, then by varying the threshold of assigning positive labels, the classifier produces a series of (FPR, TPR) points in ROC space. The curve created by connecting these points while varying the threshold from 0 to 1 is called an ROC curve. An ROC curve depicts the probabilistic performance of a classifier in a two-dimensional graph. However, for the purpose of comparing probabilistic performances between classifiers, a scalar value is more desirable. A commonly used measure derived from the ROC curve is the area under the ROC curve (AUC). The values of AUC have a range of $(0, 1)$. However, in most cases the value of AUC of an inferred classifier is greater than 0.5 which is the score achieved by random classifiers. The AUC of a classifier is equivalent to the probability that a randomly chosen positive instance will be ranked higher than a randomly chosen negative instance [77, 78].

Area Under Curve (AUC) has been used by the machine learning community as a criterion to evaluate the performances of probabilistic prediction of inferred models. Some research [81, 160, 114] advocated that the ROC graph and the AUC replace the traditional right/wrong accuracy as criteria for decision tree inference algorithms. Ferri et al. [81] demonstrated that using AUC splitting criterion in decision tree inferences leads to trees with equal or better AUC values while retaining better or equal accuracy. However, one limitation of the AUC is that it was originally proposed as a measure of effectiveness of discriminating a pair of classes. As such, it cannot be directly applied in multi-class classification problems. Several attempts have been made to address this issue. Provost and Domingo [159] calculated AUCs for multi-class problems by performing M (1 versus the rest) binary classifications. Then the corresponding AUCs are weighted, averaged by the relative class abundances.

$$AUC_M = \sum_{c_i \in C} AUC(c_i, c_r)p(c_i),\ i \in M$$

where c_i represents class i and c_r represents the rest of the class except class i.

Hand and Till [96] derived another scheme to obtain the AUC for multi-class problems. It is based on AUCs of pairwise binary classifications. The result is therefore given by

$$AUC_M = \frac{2}{|C|(|C|-1)} \sum_{c_i, c_j \in C} AUC(c_i, c_j),\ i \in M$$

The Hand and Till method has a nice property that it is insensitive to changes in class distributions. Ferri et al. [82] proposed an extension to the AUC in the form of the Volume Under ROC Surface (VUS) for multi-class problems. It showed that with high computational cost in terms of $O(n^{|C|(|C|-1)})$, it is possible to compute the VUS for multi-class problem. Experiments to compare several approximations methods to compute the VUS were conducted, the study identified that the best approximation seems to be HT3 [96], which is a simplification of Hand and Till's M function.

Uniqueness of Log-Probabilistic Scoring

To address the issue of comparing probabilistic performance between classification algorithms, in addition to the conventional classification accuracy, a metric called logarithm of probabilistic scoring [95, 94, 62, 57, 186, 128, 39, 188],[187, sec.5.1],[40, sec.11.4.2] was implemented for comparisons of probabilistic prediction performance of machine learning learning algorithms. It is defined as $-\sum_{i=1}^{N} \log(p_i)$, where N is the total number of test data and p_i is the predicted probability of the true class associated with the corresponding data item [95, 94, 57, 62]. The metric can be interpreted as the optimal coding length for the test data given the resultant tree models. This metric can be used to approximate (within a constant) the Kullback-Leibler (KL) distance between the true (test) model and the inferred model. Its relation to log-likelihood via $-\log(\prod_{i=1}^{N} p_i) = -\sum_{i=1}^{N} \log(p_i)$, its relation to Kullback-Leibler distance and its corresponding general applicability to a wide range of probability distributions (recall section 5.5) [95, 94, 57, 62, 186, 128] strongly recommend this log(prob) bit costing as a statistically-based general alternative to metrics such as ROC and AUC [77, 78, 159, 153]. Moreover, LP score is unique, being invariant to re-framing of problems [49, footnote 175] [50, pp437-438] [51, sec. 3.2][52, sec. 4.1, especially p19].

Logarithm of probabilistic (bit) score - LP score, enables us to compare probabilistic prediction accuracy of inferences from an identical training data set by various decision tree and graph algorithms. The lower the value of the LP score, the more consistent the predicted probabilistic model is with the true model.

Given an array of occurrences of events of an M-state multinomial distribution $(c_1, c_2, \ldots, c_M)$, the probability of a certain event j can be estimated by (either)

$\hat{p_j} = \frac{c_j + 0.5}{(\sum c_i) + M/2}$ [209, p187(4);p194(28);p186(2)][218][217, p75][186] or

$\hat{p_j} = \frac{c_j + 1}{(\sum c_i) + M}$ [209, p187(3);p189(30)][186],

Table 3.1: An example of probabilistic trees

Index of leaf node	Number of positive instances	Number of negative instances	Prob. assigned to positive class
1	a_1	b_1	p_1
2	a_2	b_2	p_2
...	...	...	...
i	a_i	b_i	p_i
...	...	...	...
M	a_M	b_M	p_M

the latter being known as the Laplace estimate and also corresponding (with uniform prior) to both the posterior mean and the minimum expected Kullback-Leibler distance estimator [214]. In the experiments here, the first (+0.5) is used in MML multinomial message length calculations, $\hat{p_j}$ estimations and calculations of the log(prob) bit costing.

3.3.3 Comparing LP Scoring with Area Under Curve (AUC)

Consider Table 3.1 which represents a generalised example where the target variable is assumed to have binary values from positive and negative values. The inferred decision tree has M leaf nodes $L_i, i \in [1, M]$. At each leaf node, a pair $(p_i, 1 - p_i)$ is given as probabilistic score for the positive and the negative class. Without losing generality, the leaf node index is arranged so that $p_1 > p_2 > \ldots > p_M$. The size of test set is N, which consists of N^+ positive class instances and N^- negative class instances. Table 3.1 gives a summary of the resultant probabilistic tree.

Fawcett introduced an efficient algorithm [77] to generate a ROC graph from table 3.1. The algorithm works on a list of test instances decreasing by positive probabilistic score. For each positive instance the algorithm increments TP and for each negative instance it increments FP. The generated ROC graph for the decision tree is shown in Fig. 3.6.

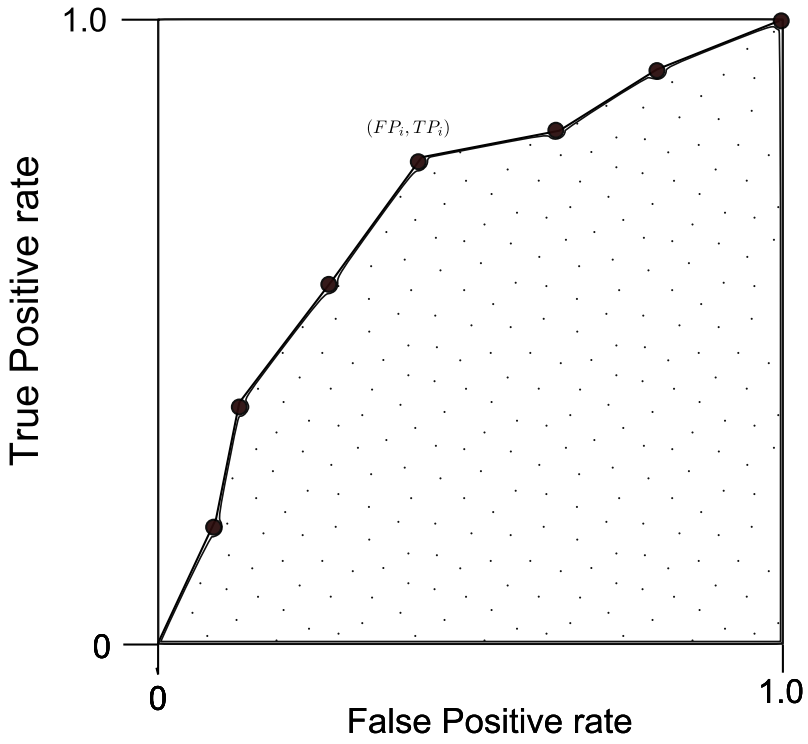


Figure 3.6: ROC and the corresponding AUC for a decision tree

The Log-Probabilistic score (LP score) for the decision tree can be calculated by summing up the LP score of instances in the test set. To illustrate the statistical property of the LP score, the LP score is divided by the size of the test set N, so average of LP score for an instance in the test set is given by

$$\begin{aligned}
&\frac{1}{N}\Big(-\sum_{i=1}^{M} a_i \log p_i - \sum_{i=1}^{M} b_i \log\left(1-p_i\right)\Big) \\
=&\sum_{i=1}^{M} \frac{a_i+b_i}{N}\Big(-\frac{a_i}{a_i+b_i}\log p_i - \frac{b_i}{a_i+b_i}\log\left(1-p_i\right)\Big) \\
=&\int_X p(x) \int_Y \ell(x|y,T)p(y)dydx \\
=&E\Big[\ell(x|T)\Big]
\end{aligned} \tag{3.2}$$

Table 3.2: A competing probabilistic tree

Index of leaf node	Number of positive instances	Number of negative instances	Prob. assigned to positive class
1	a_1	b_1	$p_1^{'}$
2	a_2	b_2	$p_2^{'}$
...	...	...	...
i	a_i	b_i	$p_i^{'}$
...	...	...	...
M	a_M	b_M	$p_M^{'}$

Equation 3.2 shows that the expected LP score approximates the expected code length for encoding a randomly chosen sample data from the population, assuming that the test samples are generated by the same underlying probability distribution $p(x)$ which generates the training data.

Suppose that there is another candidate tree, which has tree topology and internal nodes identical to the tree specified in Table 3.1. The class probabilities of each leaf node in the tree are given by Table 3.2, which produces a distinct set of probability $p_i^{'}$. If $p_i^{'}$ follows the same decreasing order $p_1^{'} > p_2^{'} > \ldots > p_M^{'}$, even when $p_i^{'} \neq p_i$, the two decision trees have identical ROC graphs shown in Fig 3.6, hence both trees have the same AUC score. On the other hand, it is obvious that

$$-\sum_{i=1}^{M} a_i \log p_i - \sum_{i=1}^{M} b_i \log (1 - p_i) \neq -\sum_{i=1}^{M} a_i \log p_i^{'} - \sum_{i=1}^{M} b_i \log (1 - p_i^{'})$$

The example shows that the LP score gives distinct probabilistic scores to these two trees while the AUC fails to differentiate the probabilistic performances of two trees. Another advantage of the LP score is that it generalises to the multi-class problem naturally. As discussed in section 3.3.2, there are several different ways to generalise the AUC into multi-class problems. Hence the LP score is used in this study to compare the probabilistic performances of classifiers.

3.4 Decision Tree Inference by MML Coding

The Minimum Message Length (MML) Principle [209, 218, 214, 187, 207] provides a guide for inferring the best model given a set of data. If a set of data is to be transmitted, it can either be transmitted directly (as it is); or alternatively a theory can be inferred from the data, then the set of data is transmitted with the help of the theory. Thus, the transmitted message is composed of the following two parts:

I. the description of the theory, or hypothesis, H.

II. the data, D, explained with help of the theory: D given H, or $D \mid H$.

From Bayes's theorem, we have

$$Pr(D)Pr(H \mid D) = Pr(H\&D) = Pr(H)Pr(D \mid H)$$

So

$$Pr(H \mid D) = \frac{Pr(H)Pr(D \mid H)}{P(D)} \tag{3.3}$$

where Pr(H&D) is the joint probability of D and H, Pr(H) is the prior probability of the hypothesis H, Pr(D) is the marginal probability of data D, Pr(D|H) is the statistical likelihood (function) of D given H and Pr(H|D) is the posterior probability of H given observed data D. From 3.3,

$$-\log Pr(H \mid D) = -\log Pr(H) - \log Pr(D \mid H) + \log Pr(D)$$

To maximize Pr(H|D) is equivalent to minimizing $-\log Pr(H \mid D)$.
Because $\log Pr(D)$ is a constant independent of the choice of H, it can be ignored and a minimum sought for $-\log P(H) - \log P(D \mid H)$.
Thus the hypothesis with the minimum two-part message length can be said to be the most probable and in turn, the best trade-off between simplicity and goodness of fit for the given data. For details of MML, see [209, 218, 214, 147, 70, 39, 40]. For a comparison between MML and the subsequent Minimum Description Length (MDL) principle[166], see, e.g., [214] (which also gives a survey) and other articles in that 1999 special issue of the *Computer Journal*, [40], [207, chap. 10], [51] and [52, sec 6.7].

MML and the subsequent Minimum Description Length (MDL) principle [166, 120] are widely used for model selection in various machine learning problems, and both can be thought of as operational forms of Ockham's razor [128]. In practice, MML and MDL work very well on inference of decision trees.

As discussed in section 3.2.4, decision tree learning algorithms should avoid over-fitting the training data set with over-complex decision trees. In MML decision tree inference schemes, there are two parts in the message. The first part of the message describes a candidate decision tree model (including the probability distributions in the leaves) whereas the second part of the message describes the data given the conveyed decision tree. The aim of a decision tree learning algorithm based on MML principle is to find the decision tree which has the minimum two-part message. Therefore the inferred tree has the optimal trade-off between fitting the data and having a low complexity.

From a Bayesian point of view, the coding scheme in effect defines a prior probability on every candidate tree T, $\{Pr(T) = 2^{-M_T}\}$, where M_T is the message length in bits for conveying the topological structure of the tree T. The prior distribution over the candidate trees should satisfy the requirement for a proper prior – the sum of the prior distribution should be less or equal to 1.

$$\sum Pr(T) = \sum 2^{-M_T} \leq 1$$

The second consideration for constructing the coding scheme for the structure is related to the decision tree inference problem itself. The aim of adopting a two-part code length as the selection criterion for inferring the optimal decision tree is to avoid over-fitting the training data. The implemented coding scheme assigns shorter code lengths (implying higher priors) to candidate trees with simple topological structures whereas longer code lengths (implying lower prior) are given to candidate trees with complex topological structures. Thus, encoding the topological structure of decision trees enables decision tree learning algorithm to rein in complexity of the inferred candidate trees.

Among work that has been put into the development of tree-based classification techniques in recent years, Quinlan and Rivest [165] proposed a method for inferring

decision trees using MDL. Wallace and Patrick subsequently [224] presented a refined coding scheme for decision trees using MML in which they identified and corrected some errors in Quinlan and Rivest's derivation of the code length [40, sec. 11.4.3], pertaining to the issue of probabilistic prediction.

The Wallace and Patrick [224] MML tree inference scheme made several important contributions towards coding-based tree inference algorithms, such as an efficient encoding for splits with multiple outcomes, encoding multinomial distributions and the idea of adaptive parameters in class prior probabilities. The Wallace and Patrick decision tree coding scheme is discussed in detail in the following subsections.

3.4.1 Efficient Codes for Topographic Structure of Multi-way Trees

As discussed in section 3.2, there are two types of nodes in decision trees: internal nodes under which have child nodes and leaf nodes that are at the end of branches. A binary code is sufficient to encode such tree structures. Quinlan and Rivest [165] implemented a simple code which encodes both internal nodes and leaf nodes with 1 bit. Such a simple code in effect returns a code length equivalent to the number of nodes in the tree. However, the simple code is inefficient when there are internal nodes with more than two child nodes in inferred trees [165, 224]. To simplify the discussion, assume the tree to be encoded is an M-ary decision tree, i.e., there are M child nodes of each internal node. Let the probability of a node being an internal node be p, then the relationship between p and the expected tree size $E(|T|)$ can be derived by

$$E(|T|) = \left(1 - p + p(1 + \sum^{M} E(|T_i|))\right) \tag{3.4}$$

where $E(|T_i|)$ is the expected size of the subtrees (if there are any) under the node. Assuming $E(|T_i|) = E(|T|)$, then

$$p = \frac{E(|T|) - 1}{M \cdot E(|T|)} \tag{3.5}$$

In general, the number of candidate trees is extremely large, even for very simple data sets. An example in [164] shows that for a training data set consisting of 14 items, there are more than $4\text{x}10^6$ trees that are consistent with the training set. As $E(|T|) \gg 1$, $p \approx \frac{1}{M}$. To look at the problem from another perspective, re-write equation 3.5,

$$E(|T|) = \frac{1}{1 - pM} \tag{3.6}$$

From equation 3.6, it can be derived that to set $p = \frac{1}{M}$ implies the expected tree size to be infinite. On the other hand, setting $p \in [0, \frac{1}{M})$ implies finite trees are expected. Following the above discussions, Wallace and Patrick pointed out [224] that the optimal coding scheme should encode an internal node with $-\log(\frac{1}{M})$ bits and a leaf node with $-\log\frac{M-1}{M}$ bits, with the code book generated by such a scheme having the prefix property $\sum 2^{-T_i} < 1$, where T_i is the code length to encode a tree structure.

3.4.2 Encoding Split Functions and Category Data

When the topology of a tree is conveyed to the receiver, the "contents" of nodes of the tree are also needed. For an internal node, the parameter(s) of the split function at the node must be specified (encoded and sent to the receiver). In this section, discussions are limited to coding schemes dealing with univariate functions. In section 3.5, a MML coding scheme to encode trees with multivariate linear functions at internal nodes will be discussed.

In the case of a split function with a discrete univariate variable, both sender and receiver are assumed to know the property of the input variables, therefore the only information needed to be specified is the variable itself. For data with M input variables, the cost to specify the variable is log(M). For a continuous variable, one or multiple thresholds must be specified for dividing the set into K subsets. However, as found in [197, 208], with a limited amount of data it is quite difficult to estimate the optimal K for the set. Most algorithms set K=2 and only one threshold is needed to be conveyed. For an internal node with N instances, there are only a maximum of N-1 distinct ways to partition the set into two subsets using one threshold. Applying a uniform prior on these thresholds gives a code length of $\log(N - 1)$.

For a leaf node, there are data (usually category data for a decision tree inference problem) that need to be conveyed. Here, an incremental coding scheme is used to encode category data at leaf nodes. The incremental code does not attempt to state a "theory", i.e., the multinomial distribution at the leaf node. Instead, it merely encodes the observed data efficiently. It works as follows. For a set of M-class category data, assume a uniform prior on all classes at the beginning, i.e., one observation for each class before the transmission starts. The probability for each class is updated every time a datum is encoded (observed). For example, after an instance belonging to class C_m is encoded, probability for class C_m is updated from $\frac{i_m+1}{j+M}$ to $\frac{i_m+1+1}{j+M+1}$, where i_m is the number of instances belonging to class C_m before this instance, while j is the index number of the datum being encoded. Under this scheme, the total code length to encode the category information of data set at a leaf node is given by

$$-\log\left(\frac{\prod_{i=1}^{M} I_i!}{M(M+1)(M+2)\cdots(M+N-1)}\right) = -\log\left(\frac{(\prod_{i=1}^{M} I_i!)(M-1)!}{(M+N-1)!}\right)$$

where I_m ($m \in [1, M]$) is the number of the data belonging to class C_m and N is the size of the data set.

In general, when a conjugate symmetric Beta prior over the class probabilities p_i is taken,

$$P_r(p_m) = \frac{\Gamma(M\alpha)}{\Gamma(\alpha)^M}\left(\prod_{i=1}^{M} p_i\right)^{\alpha-1}$$

where $\alpha \in (0, 1)$, the message length formula for the multinomial distribution becomes

$$-\log\left(\frac{\Gamma(C\alpha)}{\Gamma(C(\alpha+1))}\prod_{i=1}^{M}\frac{\Gamma(I_i+\alpha)}{\Gamma(\alpha)}\right)$$

Wallace and Patrick [224] also proposed an iterative method to estimate the optimal value of α in their decision tree inference scheme.

3.5 Oblique Decision Trees

This section presents a novel MML decision tree inference scheme. In the proposed scheme, multivariate functions replace univariate functions at internal nodes of inferred trees as the class of split functions. The algorithm uses a MML coding scheme as an objective (goodness-of-fit) function when selecting multivariate splits at internal nodes. The new objective function to evaluate multivariate splits not only costs the results of the split, but also takes the generalization ability of the split into consideration.

Many decision tree algorithms only permit splits at internal nodes by univariate discriminate functions, which have only one input variable. Decision tree algorithms based on univariate split functions have given excellent results and have been successfully solving a wide range of machine learning and data mining problems. Such decision trees are often known as univariate trees. However, one of the obvious limitations of univariate trees is that their internal nodes can only separate data with hyperplanes perpendicular to the co-ordinate axes. This limitation reduces the expressive power of univariate trees. If the size of training data set is small, it can often lead to comparatively poor rendering of many classes of concepts.

One solution to this issue is to allow splits at internal nodes with multivariate functions. Multivariate decision tree algorithms generate decision trees by employing discriminant functions with more than one attribute at internal nodes. With these discriminant functions, multivariate trees are able to partition the instance space with hyperplanes having arbitrary slopes – rather than only parallel to the co-ordinate axes. Such flexibility alleviates the problem of inefficient representation in univariate trees. Hence, the resultant multivariate trees tend to be more concise than their univariate trees counterparts.

On the other hand, multivariate discriminant functions can take multiple input variables. Since an univariate function can be regarded as a special case of a multivariate function, it is obvious that multivariate discriminate functions have more expressive power and flexibility on partitioning data. The difference between using univariate functions and multivariate decisions to divide a set of data is illustrated in Fig. 3.7. It demonstrates a case where, to separate the data purely, a univariate

tree would need at least three cuts while a multivariate tree would require only a single cut.

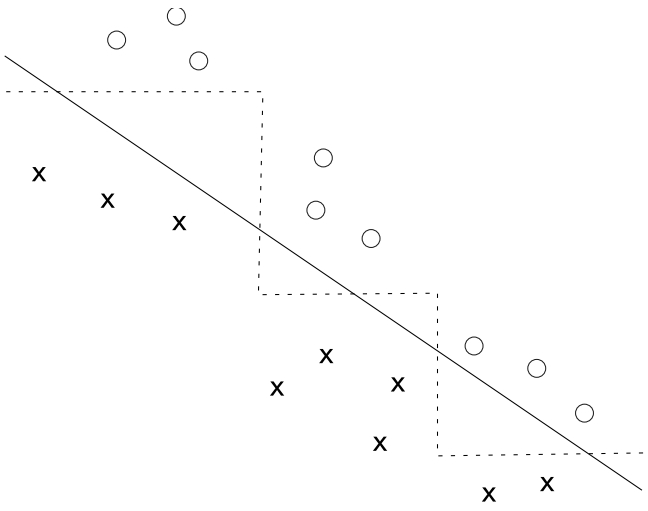


Figure 3.7: The difference between univariate and multivariate trees

Decision tree learning algorithms recursively partition the data into a finite number of homogeneous subsets (leaf nodes) which have separate inference models. For some problems, as shown in the Fig. 3.7, adopting multivariate discriminate functions to partition the data greatly increases effectiveness of the splits.

Other machine learning algorithms, such as support vector machine (SVM) [195, 173] and logistic regression [101] also employ linear multivariate discriminant functions to classify multivariate data. While multivariate tree uses multivariate linear functions in input space, SVM uses linear hyperplanes in a higher dimensional feature space defined by a kernel function to separate data. In logistic regression, linear functions serve as inputs to logistic function.

Due to the greater expressibility and flexibility of having multivariate functions in internal nodes, multivariate decision trees have the potential to render complex underlying concepts in a neater and more concise tree than univariate trees. However, the huge increase in the number of the candidate functions at internal nodes also poses new challenges in constructing multivariate tree inference schemes. Improving generalization ability is amongst the major challenges in inferring multivariate trees.

To keep the computational time reined in, most of the multivariate tree systems only allow the use of linear functions – such trees are often referred to as oblique decision trees. Here an oblique decision tree inference scheme is proposed by using the MML principle [209, 218, 214, 187].

The test results show that these oblique decision tree inference algorithms are able to find smaller trees while maintaining better or comparable accuracy compared to the standard univariate decision tree schemes C4.5 and C5 [164].

The following subsections are organized as follows. Related work on multivariate decision trees and the motivation for the new scheme for multivariate trees are discussed in 3.5.1. Subsection 3.5.2 to 3.5.4 describe MML inference of decision trees and the details of the new MML oblique decision tree inference scheme. Finally experimental results are given and analyzed in subsections 3.5.5 to 3.6.

3.5.1 Existing Multivariate Decision Tree Schemes - CART and OC1

Although most of the current decision tree learning algorithms concentrate on univariate trees, the benefits of multivariate decision trees have long been known and many multivariate decision tree schemes have appeared in the machine learning literature. Most of them are oblique decision trees, whose internal nodes test on (only) linear combinations of attributes. CART [33] was introduced by Breiman et al. and probably is the first oblique decision tree system.

With a multivariate input vector $(x_1, x_2, \ldots, x_d)$, a d-dimensional hyperplane $(w_1, w_2, \ldots, w_d)$ and a scalar w_{d+1}, then a split function at an internal node L is defined as $\sum_{i=1}^{d} w_i x_i \leq w_{d+1}$. CART implemented a deterministic search algorithm to find an optimal split function as follows.

Normalize data for all d attributes, set T=0 and given a fitness function G(L) and real number ϵ

1. T=T+1

2. Let i=0, $v = \sum_{i=1}^{d} w_i x_i$, $\gamma \in \{-0.25, 0, 0.25\}$ and $\delta \in R$

3. Search for the δ, γ that maximize the fitness function G(L) of the split $v - \delta(w_i + \gamma) \leq w_{d+1}$

4. Let δ^* and γ^* be the optimal values from step 3, update w_i and w_{d+1} so that $w_i = w_i - \delta^*$, $w_{d+1} = w_{d+1} - \delta^*\gamma^*$.

5. $i = i + 1$; if $i \leq d$, go to step 3

6. Search for the best w_{d+1} that maximizes G(L)

7. If $G(L)^T - G(L)^{T-1} > \epsilon$, go to step 1

The idea of the above heuristic is to perturb the hyperplane in one dimension at each iteration to search for the optimal hyperplane. However, as pointed out in [126], there are two limitations to the CART algorithm. Firstly it is deterministic so that when the algorithm is trapped in a local minimum, it is unable to escape. Secondly, G(L) is not guaranteed to converge, so the search process may run indefinitely unless a pre-set stopping condition is imposed. To resolve these problems, several algorithms take new randomized approaches for inferring oblique decision trees. Simulated Annealing of Decision Trees (SADT) [98] implemented simulated annealing to search for optimal oblique splits at each tree node. Murthy proposed a new refined search algorithm called OC1 [126], in which random perturbations of the hyperplane are performed to escape from local minima. The OC1 algorithm can be summarized as follows.

Given a leaf node L, a fitness function G(L) and a (run bound) integer J,

Set T=0

1. Choose a random hyperplane, H

2. Search for the optimal hyperplane, H, by using a deterministic search algorithm similar to CART

3. Repeat step 2 until value of fitness function G(L) does not improve

4. Randomly perturb the hyperplane, H, in one dimension

5. T=T+1; if $T \leq J$ then go to step 2

SADT and OC1 combine deterministic searches and random processes to avoid the local minima traps. The main drawback of these approaches is the substantial increase of time complexity.

Another important issue of multivariate decision tree schemes is to select a subset of attributes included in the test at each node. To decide the optimal subsets of features in tests at each node, the algorithms must find the trade-off between the complexity and the goodness of fit. Due to the considerable increase of candidate splits compared to cut-points in univariate trees, the main challenge for these oblique tree schemes is to differentiate the complexities of splits and quantify the trade-offs. While most of the schemes focus on the search issue, they rely on pruning to improve the generalization accuracy and take some ad hoc approaches to the complexity. Bennett and Blue [16] proposed to infer multivariate decision trees by using support vector machines (SVMs) [195] at internal nodes. Their algorithm[16] used the structural risk minimization (SRM) principle [195] to find the optimal splits. The scheme generates very simple decision trees – one with three non-linear multivariate decisions so that they are more like hybrids of SVMs than decision trees. However, the trade-off between the topological complexity of the tree and the complexity of the decisions remains unresolved in their scheme. Here this problem is addressed by using MML inference. The sum of the message length of a split and the encoded data given the split provides a trade-off between the complexity and the goodness of fit. This MML scheme is detailed in the following sections. This scheme is also built upon the author's previous work [188].

3.5.2 MML Inference of Multivariate Decision Trees

MML inference [209, 218, 214, 187] has been successfully implemented in [224] to infer univariate decision trees (refining [165]). Wallace and Patrick subsequently [224] presented a refined coding scheme for decision trees using MML in which they identified and corrected some errors in Quinlan and Rivest's derivation of the message length, including pertaining to the issue of probabilistic prediction. Wallace and Patrick also introduced a "Look Ahead" heuristic of arbitrarily many ply for

selecting the test attribute at a node. Here the Wallace and Patrick decision tree coding [224] is re-used as part of the coding scheme for the new oblique decision tree program.

In this chapter, the MML principle is used to infer multivariate decision trees. The multivariate decision tree scheme proposed here generalizes earlier MML decision tree work and should provide the advantages described in Section 3.5.1 and illustrated in Fig. 3.7.

3.5.3 Encoding an internal split with a linear discriminant function

The new MML decision tree coding scheme is able to encode an internal split using a linear discriminant function. Firstly, the data falling at an internal node is scaled and normalized so that every datum falls within a D-dimensional unit hyper-cube, where D is the number of input attributes. A linear decision function $d(w, x, b) = 0$ is written as $(\sum_{i=1}^{D} w_i x_i) + b = w \cdot x + b = 0$ where $w, x \in R^D$, $\cdot$ denotes the dot (or scalar) product, and the scalar b is often called the bias. The data is divided into two mutually exclusive sets by the following rules.

If $d(w, x_j, b) > 0, j \in [1, N]$, then x_j is assigned to set I (denoted '1' or '+').
If $d(w, x_j, b) < 0, j \in [1, N]$, then x_j is assigned to set II (denoted '2' or '$-$').

Encoding the hyperplane is equivalent to transmitting the value of vector w and the bias b to a receiver. Suppose the desired value of the vector w is w_c. If w_c is stated exactly (to infinite precision), it will cost infinitely many bits of information in the first part of the message. So instead, state a set of vectors $\Lambda(\theta), \theta \in (0, \frac{\pi}{2})$, which is defined as

$\Lambda(\theta) = \{w : \arccos(\frac{w \cdot w_c}{\|w\| \cdot \|w_c\|}) < \theta\}$

This is the set of vectors which form an angle less than θ with the optimal vector w_c as illustrated in Fig. 3.8. The probability that a randomly picked vector falls into the set is given by $\frac{V_\theta}{V_T}$, where V_θ is the volume of a partial sphere of radius θ and V_T is the total volume of the unit sphere. The value of $\frac{V_\theta}{V_T}$ is given [171] by $(\sin\theta)^{2(D-1)}$, so the information required to specify the set of the vectors is $-\log((\sin\theta)^{2(D-1)})$.

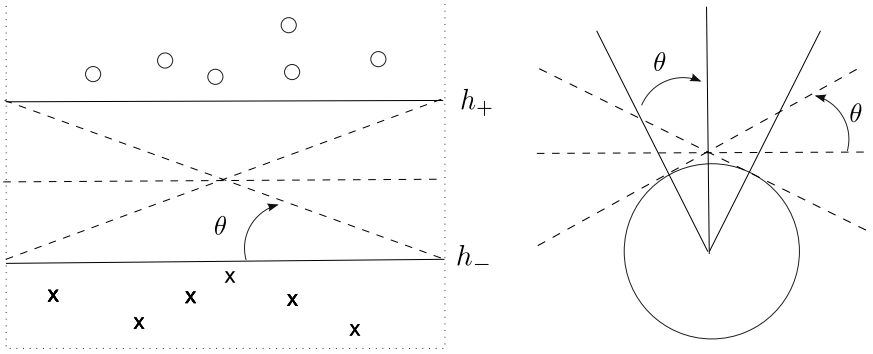


Figure 3.8: The set of hyperplanes defined by vector $w \in \Lambda(\theta)$ (Left of the Fig.) and a partial sphere of radius θ formed by $w \in \Lambda(\theta)$ (Right of the Fig.)

By specifying one data point on each side of the hyperplane h_c, two hyperplanes which are parallel to the decision surface d(w,x,b)=0 are also defined. These two hyperplanes are denoted as h_+ and h_-. These (h_+ and h_-) and the other boundaries of the unit cube form a hyper-rectangle as shown in Fig. 3.8.

The value of the θ is required so that the hyperplanes specified by vectors in the set $\Lambda(\theta)$ do not intersect with the hyperplane h_+ and h_- within the hypercube. Imagine a rectangle whose length of one side is the distance between h_+ and h_- and whose length of the other side is $\sqrt{D}$, which is the longest diagonal in a D-dimensional unit cube. Since {x: kwx+kb=0} $\equiv$ {x: wx+b=0} for any non-zero k, then choose w so that the margin between h_+ and h_- is equivalent to $\frac{2}{||w||}$. As shown in the Figure 3.9, given the margin $\frac{2}{||w||}$, if $\theta < \alpha$, where $\alpha = \arcsin(\frac{2}{\sqrt{D||w||^2+4}})$, it is easy to show that the hyperplane h_w defined by the vector w does not intersect with hyperplanes h_+ and h_- within the D-dimensional hyper-cube (from Fig. 3.8). As such, the message length needed to define a representative optimal hyperplane is: $M = -2(D-1)\log(\frac{2}{\sqrt{D||w||^2+4}}) + \log\binom{N}{2}$, where N is the number of data. See also discussions in [49, 0.2.2], re MML SVMs.

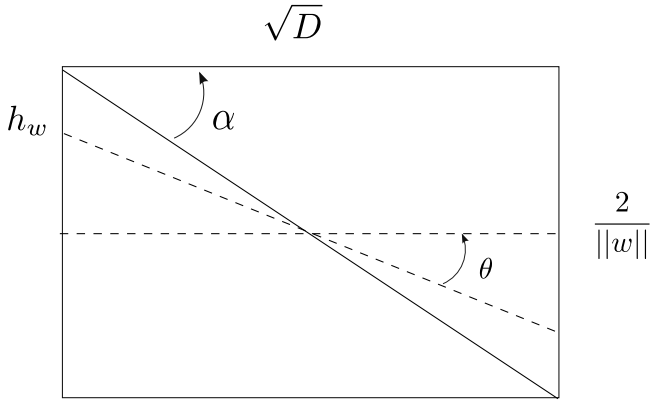


Figure 3.9: The upper bounds of θ

3.5.4 Search for the Optimal Hyperplane

One way to search the optimal hyperplane at an internal node is to employ one of convex optimisation methods [28], such as linear programming [43] and gradient descent [177]. However, non-convex optimisation methods have also achieved good results. Evolutionary algorithms are widely used in decision tree induction algorithms [10]. Bot and Langdon [20] proposed a genetic programming algorithm for evolving oblique decision trees. In this study, search heuristic implemented in OC1 [126] and SADT [98] are not used. Instead, a simple evolution strategy is implemented as the preliminary search heuristic. The approach is similar to the search heuristic appeared in [34], in which promising results were reported. The search process in this scheme can be summarized as follows. Assuming the linear discriminant function takes the form $\sum_{i=1}^{d} w_i x_i < w_{d+1}$, for each leaf node L, let M(unsplit) denote the message length of the node L while the node is unsplit, and let M(T) denote the message length of the subtree when node L is split by vector w^T at round T. The algorithm searches for the best vector w in the following steps:

Initialize T=0, input R, MaxP, M(unsplit)

1. Re-scale the coefficients of the vector w such that $\sum_{i=1}^{d} w_i^2 = 1$
2. With $v \sim N(0,1)$, randomly pick $j \in [1, d+1]$, $w_j^{T+1} = w_j^T + v$
3. if $M(T+1) < M(T)$, go to step 5
4. $w_j^{T+1} = w_j^T$
5. T=T+1; if $T < R$, go to step 1
6. Randomly select d (here, d is limited to 2 or 3) attributes
7. P=P+1; if $P < MaxP$, go to step 1
8. if $M(R) < M(unsplit)$, return w and M(R), otherwise return null and M(unsplit)

It is obvious (from steps 2 and 6) that the search process is non-deterministic, and thus the algorithms are able to generate many different trees. As such, these algorithms can be extended to take advantage of this by choosing the best one among these trees or averaging results from these trees.

3.5.5 Experiments

To evaluate the new oblique decision tree scheme, experiments are run on nine data sets selected from the UCI Machine Learning Repository [18]. The performance of the scheme is compared with those of C4.5 and C5 [164]. In addition to the traditional right/wrong accuracy, the probabilistic performance [187, sec 5.1] [62, 57, 128, 186] [49, footnote 175] [50, pp437-438] [51, sec. 3.2][52, sec. 4.1] of the learning algorithms are compared. In many domains, like oncological and other medical data, not only the class predictions but also the probability associated with each class is essential. In some domains, like finance, (long term) strategies heavily rely on accurate probabilistic predictions. Decision trees as probabilistic models are discussed in the following section.

3.5.6 Comparing and Scoring Probabilistic Predictions

Decision trees are often used as classifiers in many machine learning problems. In case where the target attribute is multinomial, each leaf node in a tree is given a class label corresponding to the class with the highest inferred probability for this node. However, the multinomial distribution given by a model in each leaf node can also be interpreted as a probabilistic (prediction) model. In this way, decision trees are not only classifiers, but they can also serve as probabilistic prediction models. Provost and Domingos [159] showed that with some modifications, tree induction programs can produce very high quality probability estimation trees (PETs). Perlich, Provost and Simonoff [153] also observed that for large data sets, tree induction often produces probability-based rankings that are superior to those generated by logistic regression. To compare probabilistic prediction performances, a new metric called the related (test data) code length (RCL) is proposed, which is defined as

$$RCL = -\frac{\sum_{i=1}^{n} \log(p_i)}{n \log(M)}$$

where n is the total number of the test data, M is the number of classes in the target attribute and p_i is the probability assigned to the real class associated with the test instance i by the model.

This metric can be interpreted as follows. For each instance of the test data set, a probability distribution for the target classes is given by the model. Then each instance is encoded with a code length corresponding to the probability assigned to the real class associated with the test instance by the model. If there are M classes of the target attribute, assuming a uniform prior, each instance in the test data set will be encoded with log(M) bits by the null theory, i.e., not attempting to infer a probability distribution and using an identical code length for each class. The related test data code length (RCL) is equivalent to the average code length of the test data encoded by a model divided by the average code length encoded by the null theory. Thus the smaller the RCL is, the better the performance of a model on probabilistic prediction. If RCL is larger than 1, a model performs worse than the null theory on probabilistic prediction. Also, as a multipliable variant of log-loss

[49, footnote 175][50, pp437-438][51, sec. 3.2][52, sec. 4.1, especially p19], RCL has the uniqueness property under re-framing.

3.5.7 Data Sets

Nine data sets from the UCI Machine Learning Repository [18] are used in this test. The purpose of the experiment is to have the proposed algorithms perform on the real world data, especially on the oncological and the medical data, which are described briefly below.

Balance: The Balance data set can be used to model psychological experimental results. The corrected value of targeted is determined by comparing values of (left-distance * left-weight) and (right-distance * right-weight). If they are equal, it is balanced.

Bupa: This data set contains blood test results of 345 male patients. The task is to predict whether a patient has a propensity for a liver disorder.

Breast Cancer: This breast cancer data was obtained from the University Medical Centre, Institute of Oncology, Ljubljana, Yugoslavia. It is one of three oncology data sets frequently used in the machine learning literature.

Wisconsin: The Wisconsin breast cancer data is another oncology data-set repeatedly used to benchmark the performance of learning algorithms.

Credit: The Credit data set is a collection of credit application data with anonymous user ID. The data set has 24 continuous attributes.

Lung Cancer: The set contains only 32 instances with 56 attributes, which is quite typical of medical data-set. Such types of data bring great challenges to machine learning algorithms.

Cleveland: The heart disease data set was compiled by Dr. Detrano and was collected at the Cleveland Clinic Foundation. The task is to predict whether the patient has heart disease.

Sonar: The Sonar data set consists of 208 items. Each item represents either a mine pattern or a rock pattern, it has 60 continuous attributes.

Wine: The wine data set was compiled by Forina's group. The data are the results of a chemical analysis of wines from different cultivates. The data set has 13

Table 3.3: Summary of Data Sets

Data-set Name	size	Discrete Attributes	Continuous Attributes	Number of Classes
Balance	625	4	0	3
Bupa	345	0	6	2
Breast Cancer	286	9	0	2
Wisconsin	699	9	0	2
Credit	1000	0	24	2
Lung Cancer	32	56	0	3
Cleveland	303	7	6	2
Sonar	208	0	60	2
Wine	178	0	13	3

attributes.

The summary of the data sets can be found in Table 3.3.

To further test the robustness and accuracy of the learning algorithms and to eliminate bias in the data set, for each data set, a tenfold cross validation was performed. The tenfold cross validation test was repeated ten times, each with a different random partition of the data. As such, for each machine learning algorithm and each of the nine data sets, 10 x 10 = 100 independent tests were done consisting of training and modelling from 90% of the data and testing goodness of fit on the remaining 10%.

3.5.8 Test Results

The test results are presented in Table 3.4, Table 3.5 and Table 3.6.

3.6 Discussion

The MML oblique tree scheme was compared to C4.5 and C5 in Tables 3.4, 3.5 and 3.6. The results from Table 3.6 clearly suggest that the MML oblique trees are much smaller than the univariate trees from C4.5 and C5. That can be explained

Table 3.4: "Right"/"Wrong" predictive accuracy

Name	C4.5	C5	MML Oblique Tree
Balance	77.8 ±4.3	77.8 ± 4.5	88.5 ± 4.0
Bupa	65.5 ± 7.4	65.5 ± 7.8	65.1 ± 8.1
Breast Cancer	71.2 ± 8.7	71.1 ± 8.4	72.8 ± 8.0
Wisconsin	94.6 ± 2.5	94.8 ± 2.5	96.0 ± 2.3
Credit	73.2 ± 4.3	73.3 ± 3.8	75.4 ± 4.7
Lung Cancer	40.0 ± 23.3	40.7 ± 24.8	46.8 ± 22.4
Cleveland	77.1 ± 7.6	77.2 ± 7.9	77.2 ± 7.8
Sonar	72.8 ± 9.2	73.9 ± 10.0	76.0 ± 9.2
Wine	93.6 ± 5.7	93.2 ± 5.8	93.2 ± 6.1

Table 3.5: Related test data code length (RCL)

Name	C4.5	C5	MML Oblique Tree
Balance	0.93±0.12	0.92 ±0.11	0.33 ± 0.08
Bupa	1.07±0.22	1.07 ±0.21	0.96 ± 0.15
Breast Cancer	0.88±0.17	0.88 ±0.17	0.84 ± 0.14
Wisconsin	0.26±0.10	0.25 ±0.12	0.21 ± 0.10
Credit	0.88±0.08	0.88 ±0.08	0.79 ± 0.09
Lung Cancer	1.83±0.50	1.86 ±0.65	0.94 ± 0.30
Cleveland	0.80±0.24	0.81 ±0.21	0.76 ± 0.22
Sonar	1.07±0.37	1.06 ±0.42	0.98 ± 0.33
Wine	0.42±0.30	0.44 ±0.29	0.28 ± 0.18

Table 3.6: Size of resultant trees - average number of leaf nodes

Name	C4.5	C5	MML Oblique Tree
Balance	81.6±9.7	41.7 ±4.6	10.4 ±0.9
Bupa	49.2±9.8	27.3 ±5.4	6.7 ± 2.6
Breast Cancer	24.2±8.3	13.1 ±4.2	3.0 ± 0.6
Wisconsin	23.7±5.3	12.3 ±2.8	5.5 ± 0.9
Credit	151.4±17.7	77.6 ±9.1	6.5 ± 2.4
Lung Cancer	12.2±2.3	6.6± 1.1	2.2 ± 0.4
Cleveland	36.7±7.2	20.0 ±4.2	7.3 ± 1.8
Sonar	28.2±3.1	14.9 ±1.6	11.6 ± 9.3
Wine	9.6±1.3	5.4 ±0.7	3.6 ± 0.5

in the following two points. Firstly, the oblique decision trees are able to partition the data more efficiently by using multivariate linear functions at splits. Secondly, the MML algorithm does not over grow the tree and does not rely on pruning to control the complexity. The MML oblique trees perform significantly better than C4.5 and C5 on all data-sets, as shown in Table 3.4 and Table 3.5. As reported in [186, 187], MML inference is resistant to over-fitting. Another reason is that univariate trees generated by C4.5 and C5 have more leaf nodes with less data, thus more skewed distributions are often inferred from leaf nodes. MML oblique trees also have higher "right"/"wrong" accuracy than C4.5 and C5 except on the Bupa and the wine data, which suggest that more work needs to be done on the search algorithms. As expected, none of the algorithms achieved good results from the lung cancer data. Learning from a small set of data with a great number of attributes remains a great challenge for machine learning algorithms.

3.7 Summary

The multivariate tree inference scheme on data sets from the UCI machine learning repository is tested and compared to the decision tree programs C4.5 and C5. The results show that on average and on most data-sets, MML oblique trees clearly perform better than both C4.5 and C5 on both "right"/"wrong" accuracy overall and probabilistic prediction, and furthermore, manage to do this with smaller trees i.e., fewer leaf nodes. By encoding the multivariate splits and extending the MML decision tree coding scheme, the scheme succeeds in finding the optimal trade-off between the complexity of the model and the goodness of fit.

The results are promising, however, more research could be done in the future. Firstly, the search heuristic might be improved as an efficient search algorithm is crucial for any multivariate tree scheme. Secondly, as pointed out in section 3.5.4, the performance of the system might be enhanced by using multiple tree averaging. The use of MML coding for internal nodes with SVMs or nonlinear splits might also be worthwhile research topics.

Chapter 4

MML Inference of Decision Graphs

A decision tree is a comprehensible representation that has been widely used in many machine learning domains. But in the area of supervised learning, decision trees have their limitations. Two notable problems are those of replication and fragmentation. One way of solving these problems is to introduce decision graph, a generalization of decision tree, which address the above problems by allowing for disjunctions, or joins. While various decision graph systems are available, all of these systems impose some forms of restriction on the proposed representations, often leading to either a new redundancy or the original redundancy not being removed. In this chapter, an unrestricted acyclic graph representation called the decision graph with multi-way joins is introduced. Compared to existing decision graph systems, it has improved representative power and is able to infer underlying graph models with less training data. An algorithm to infer these decision graphs with multi-way joins using the Minimum Message Length (MML) principle is also introduced.

4.1 Introduction

In spite of successes of decision tree systems in supervised classification learning, the search for a confirmed improvement of decision trees has remained a continuing topic in the machine learning literature. Two well-known problems from which the

decision tree representation suffers have provided incentives for such efforts. The first one is the replication problem which leads to the duplication of subtrees in order to representing disjunctive concepts. For example, Figure 4.1 shows a decision tree in which the term (C $\wedge$ D) requires two subtrees to be represented gives an inefficient representation of the proposition (A $\wedge$ B) $\vee$ (C $\wedge$ D). Figure 4.2 illustrates the proposition as an acyclic graph. Another effect of the replication problem is that many decision tree learning algorithms require an unnecessarily large amount of data to learn disjunctive functions.

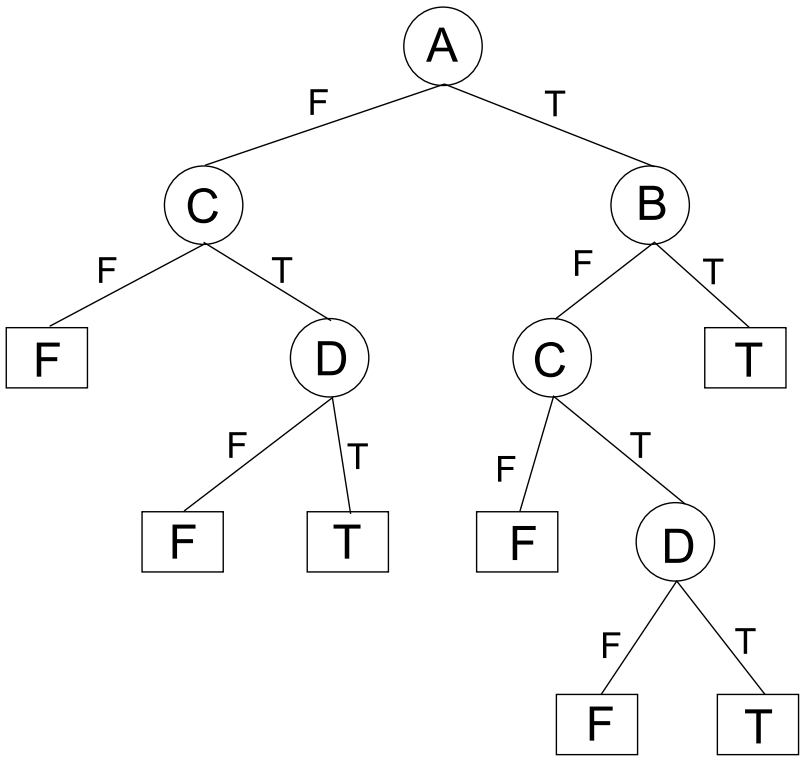


Figure 4.1: Decision tree representation for (A $\wedge$ B) $\vee$ (C $\wedge$ D)

The second problem is known as the fragmentation problem, which occurs when internal nodes of a decision tree split data into a large number of small subsets. When this happens, the subsequent subtrees under these internal nodes receive relatively little data. In such cases, the decision tree algorithm stops growing subtrees or prunes them back to avoid the over-fitting problem. As a consequence of such

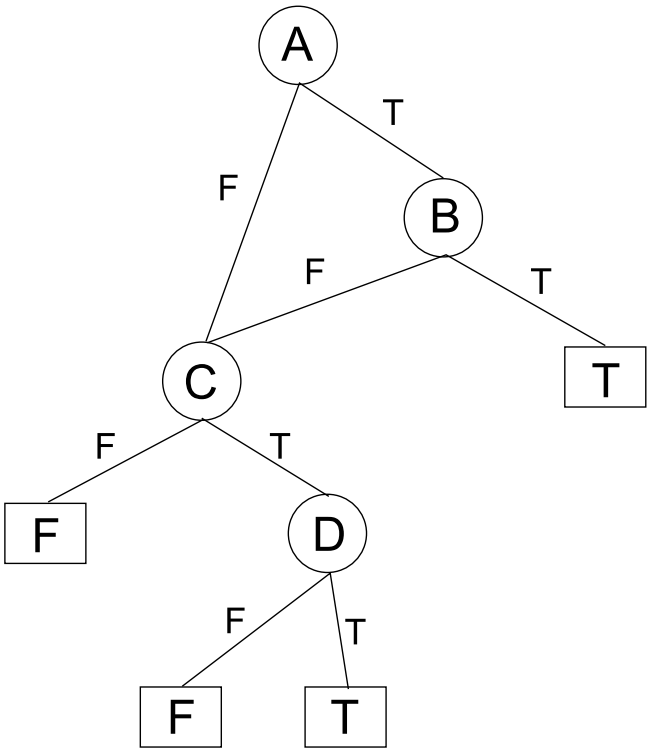


Figure 4.2: Decision graph representation for (A ∧ B) ∨ (C ∧ D)

premature termination of the inference processes, the decision tree learning algorithm fails to infer target concepts in the data. On the other hand, had the internal nodes received more data, the inference algorithm would have grown the tree to reveal the underlying concepts correctly. The fragmentation problem is illustrated in Figure 4.3. When decision nodes in a tree are split on high arity attributes (say $arity \geq 10$), data is fragmented into many partitions, each with relatively few data.

Both of these problems increase the size of decision trees and reduce the number of instances in the individual nodes. One way of resolving these problems, particularly when the number of instances in nodes are crucial for inferring the underlying concept, is to use decision graphs to provide an elegant, generalizing solution. Decision graphs [138, 134] can be viewed as generalizations of decision trees which still have decision nodes and leaves. The feature that distinguishes decision graphs from

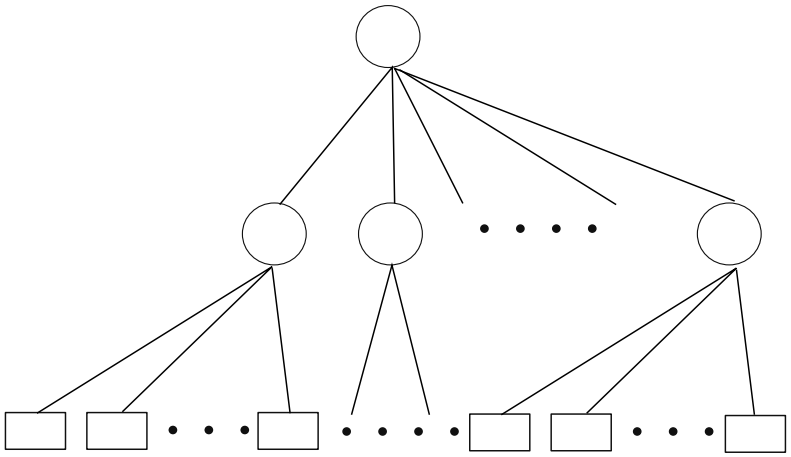


Figure 4.3: The fragmentation problem of decision trees

decision trees is that they may also contain joins (for disjunctions), which are represented by two nodes having a common child. This representation specifies that two subsets have some common properties, and hence can be considered as one subset. As shown on Figure 4.1, by merging duplicated subtrees with a join operator, the repeated subtrees - from the left hand side of Figure 4.2 - representing the term (C $\wedge$ D) are united into one subtree.

One attempt at generalizing decision trees was proposed by Mehta et al. [120]. As a refined decision tree system, it allowed multi-way joins but restricted these joins to nodes which have common parents. This improvement does, however, not help with decision tree replication problems such as that of Figure 4.1.

More recently, Mansour and McAllester [118] introduced a decision graph representation called the branch program. In their system, where decision trees were viewed as a form of boosting, the use of a boosting algorithm guaranteed that the training error declines as $2^{-\beta\sqrt{|T|}}$, where $|T|$ is the size of the decision tree and β is a constant determined by the weak learning hypothesis. Compared to decision trees, whose training error declines as $|T|^{-\beta}$, the branch program showed some promising theoretical results. However, the analysis of the generalization error of the branch program remained open, and thus no comparison with other systems could be made.

Together with some earlier works [138, 145, 107, 120], the branch program certainly indicated that decision graphs was a reasonable approach.

The HOODG system, introduced by Kohavi [107], used an Oblivious Read-Once Decision Graph (OODG). OODG was closely related to Order Binary Decision Diagrams (OBDDs). In the HOODG system, variables had to be strictly ordered and decision graphs were generated so that the graph would only test the variables in the specific order. But because the system insisted on a canonical representation, nodes with irrelevant attributes have to be included in the inferred graphs. Kohavi [107] reported significant improvement over decision tree algorithms such as C4.5 and ID3 [164] on artificial data sets generated from disjunctive functions, but the system achieved less success on real-world data sets. The system also did not adequately address noise in data.

Oliver and Wallace's system [138, 145, 147] used decision graphs that allowed joins with a pair of nodes. They also presented a MML inference coding scheme and a graph growing algorithm for the system. The decision graph system presented here builds on the Oliver and Wallace system but eliminates their limitation by allowing for multi-way joins. In this way, the scheme not only generalizes the Oliver-Wallace binary join decision graph but also generalizes the scheme of Mehta et al. [120], which only allowed joins of nodes which have common parents. A detailed comparison between the multi-way join scheme and the Oliver and Wallace binary join scheme, is given in section 4.3.2.

4.2 Decision Graphs with Multi-way Joins

A decision graph system which allows multi-way joins is proposed here. The system uses directed acyclic graphs as in both the Oliver and Wallace system [147, 145] and the Kohavi system [107]. In addition to decision nodes and leaf nodes, join nodes which are merged to have a common child are introduced. Join nodes are represented by a diamond shape in the figures. Although a join may be represented by directly attaching arcs from decision nodes to a child, join notes are explicitly included in the figures to clarify the coding scheme. The main idea behind the new decision graph representation is that a decision graph can be decomposed into a

sequence of decision trees. By doing this, some well-proved decision tree inductive techniques can be re-used in the system. The details of the algorithm are discussed in section 4.3.1. Figure 4.4 illustrates an example of how a decision graph can be decomposed into a sequence of decision trees.

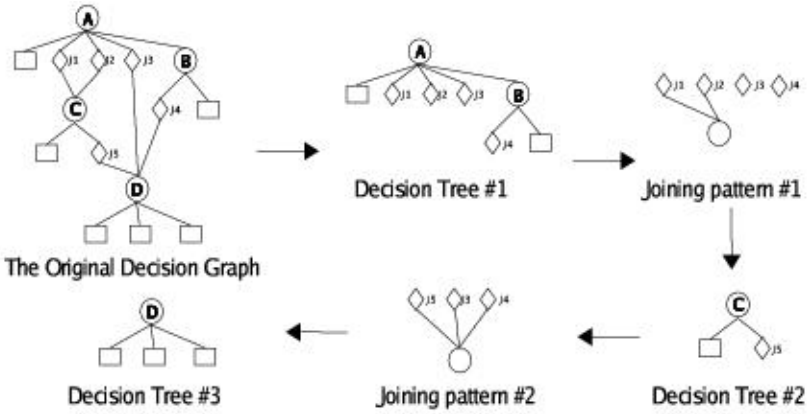


Figure 4.4: Decomposing a decision graph into a sequences of decision trees

4.3 MML Inference of Decision Graphs

4.3.1 Coding Decision Graphs with Multi-way Joins

Much effort has been put into the development of tree-based classification techniques in recent years. Quinlan and Rivest [165] proposed a method for inferring decision trees using the Minimum Description Length (MDL) principle [166]. Based on it, Wallace and Patrick [224] presented a refined coding scheme for decision trees using the MML principle in which they identified and corrected some errors in Quinlan and Rivest's derivation of the message. They also introduced a "Look Ahead" heuristic of arbitrarily many ply for selecting the test attribute at a node.

Oliver et al. presented an inference scheme [147, 145] to construct decision graphs using MML [209, 218, 214, 217]. The machine-learning technique of decision graphs was successfully applied to the inference of a theory of protein secondary structure from a particular dataset by Dowe et al. [66] (see Section 4.4.4). The

resulting decision graphs provided both an explanation and a prediction method for the problem. However, the Oliver-Wallace coding scheme [147, 145] only allows binary joins. When there are more than two nodes involved in a join, some intermediate nodes have to be created. Section 4.3.1 presents a refined coding scheme for decision graphs which allows multi-way joins to eliminate such inefficiency.

The coding scheme is refined for decision graphs so that a join operation is no longer limited to one pair of nodes. An arbitrary number of nodes can be involved in a join operation to form a common child node. The reader is referred to the earlier and similar coding scheme of Oliver et al., which only permitted binary joins [147]. To transmit a decision graph now allowing multi-way joins, the following steps are used.

1. From the root node of the decision graph, a prefix traversal of the decision tree is performed, treating any join nodes as leaves. Any join nodes that have been transmitted are added to an open list.

2. When step 1 is finished, if this results in any join nodes in the open list, then the combination pattern of the nodes in the open list is described. A combination pattern lists those groups in the open list which combine to have a child. Any groups of nodes in the open list that are involved in a join are deleted from the open list. Add their children to a new node list, in which nodes will become roots of new traversals.

3. If there is any node in the new node list, step 1 is repeated to transmit them.

By decomposing a decision graph into a sequence of decision trees (as in Figure 4.4), which are transmitted in step 1, the MML decision tree coding scheme proposed by Wallace and Patrick [224] can be re-used. Since join nodes are treated as leaves, an adaptive code is implemented to describe which leaves are actually join nodes and should be put into the open list.

In step 2 of the above process, a description of the combination pattern of join nodes from the open list needs to be communicated. Firstly, the nodes which were added to the open list in the current iterations of transmitting are defined as "new" nodes, and nodes which were added to the open list in previous iterations then become "old" nodes. This views the open list as containing N new nodes (from the most recent iteration) and Q old nodes. It should be noted that both N and Q

become common knowledge for both sender and receiver before transmission of the combination pattern.

The combination pattern of join nodes from the open list is communicated in four steps, thus four numbers (see points 1-4 below) must be transmitted in the following order.

1. The number of nodes, M, which are children of joins in this iteration.
There are at least two leaf nodes being involved in any join and because one of these leaf nodes must be a new node, so $1 \leq M \leq min(N, \frac{N+Q}{2})$. Assuming these values to be equally likely, the message length cost of transmitting this number, M, in the message would be $\log(min(N, \frac{N+Q}{2}))$.

2. The number of pending nodes, P, which are not involved in any join and the numbers of nodes $(J_1, J_2, \ldots, J_M)$ in each group of joining leaf nodes from the open list in this iteration.

The task is now equivalent to finding out the number of different solutions to the following equation.

$P + J_1 + J_2 + \ldots + J_M = N + Q$ (where $P \geq 0; J_i \geq 2; i = 1, 2, \ldots, M$)
If the number of valid answers is A and every solution is assumed a priori equally likely, the cost of transmitting those numbers in the message length would be $\log(A)$.

3. The number of "new" nodes, Y, among pending nodes and the number of new nodes, $(X_1, X_2, \ldots, X_M)$ in each group of joining leaf nodes from the open list in this iteration .

Similar to the above step and because there is at least one new node in each joining group, the task is now equivalent to finding out the number of different solutions to the following equation.

$Y + X_1 + X_2 + \ldots + X_M = N$ (where $0 \leq Y \leq P;\ 1 \leq X_i \leq J_i\ ; i = 1, 2, \ldots, M$)
If the number of valid solutions is B and every solution is assumed a priori equally likely, the cost of transmitting those numbers in the message length would be $\log(B)$.

4. The number of permutations of nodes from the open list in this iteration:

$C = \frac{N!}{Y!X_1!X_2!\ldots X_M!} \frac{Q!}{(P-Y)!(J_1-X_1)!(J_2-X_2)!\ldots(J_M-X_M)!}$
If every permutation is assumed a priori equally likely, the cost of transmitting the number in the message length would be $\log(C)$.

After finishing the transmission of the combination pattern, an adaptive code is used to encode the types (i.e., leaf or fork) of nodes in the new node list, followed by new rounds of graph traversal(s). Nodes resulting from join operations cannot be join nodes, because if a join operation involving such a node were to happen in a future iteration, the nodes which combined to form this node would have been labeled as "pending" until needed for such a join.

4.3.2 Comparison with Oliver and Wallace's Decision Graph Program

The difference between the above coding scheme and Oliver and Wallace's coding scheme for decision graphs [138, 145, 147] has been illustrated in Figure 4.5. Whenever a multi-way join is required, the Oliver-Wallace coding scheme [138, 145, 147] encodes it by proceeding with a series of consecutive binary joins. The result of each such binary join other than the last one is an intermediate node. These intermediate join nodes are redundant (or irrelevant), even though the Oliver-Wallace scheme insists upon encoding them and is thus inefficient. The new scheme is efficient in that it can encode multi-way joins without having to introduce any redundant intermediate nodes. As the example in Figure 4.5 shows, while the coding scheme with multi-way joins is able to join the four nodes in one step, Oliver and Wallace's scheme has to do it in two steps and two intermediate join nodes J5 and J6 are introduced. The message length caused by coding these intermediate nodes and various possible ways to form these joins make the scheme less inefficient. For a way of partly removing the inefficiency from the Oliver-Wallace coding scheme, see Section 4.3.3.

4.3.3 Unpublished Refinement of Oliver-Wallace Coding Scheme

The Oliver-Wallace coding scheme presented so far [138, 145, 147] is inefficient due to the fact that a series of consecutive binary joins is encoded including the order in which those consecutive binary joins occurred, even though that order is in fact irrelevant.

This coding scheme can be rectified and made efficient by accounting for this combinatorial inefficiency. For example, as Figure 4.5 shows, where four nodes join into one node by three consecutive binary joins; note that this could have been done in $C_2^4 C_2^3 C_2^2$ = 6x3x1 = 18 different ways, and that log(18) can be subtracted from the length of the corresponding (inefficient) message.

Although such a rectification has not been published, it is understood that Oliver might have since modified the Oliver and Wallace source code to at least partly correct for the above combinatorial inefficiency. However, his program is unavailable.

Such correction to the Oliver-Wallace coding scheme would - if correctly implemented - give it a comparable efficiency to the multi-way join coding scheme proposed here. The difference between the refined, corrected, Oliver-Wallace scheme and the new scheme here would essentially be a choice of Bayesian prior.

4.3.4 Growing a Decision Graph

A graph begins having one node, with the root being a leaf. The graph grows by performing the following procedures iteratively until no further improvement can be achieved.

1. For each leaf L, perform tentative splits on each available attribute in the leaf, and determine the attribute A that will lead to the shortest message length when L is split on A. Record, but do not perform, the alteration (Split L on A) along with its communication saving.

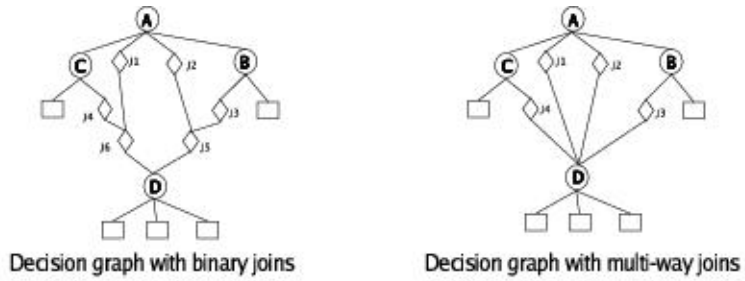


Figure 4.5: Different encoding of a function by (left) decision graphs with binary joins and (right) decision graphs with multi-way joins

2. For each leaf L, perform tentative joins with other leaves. Record, but do not perform, the alterations (join L_i and L_j; ...; join $L_i, L_j, \ldots, L_k$) along with its communication savings.

3. Sort the alterations from step 1 and step 2 by their communication savings. Choose the alteration (whether from step 1 or from step 2) that has greatest saving.

With help of information which is obtained from step 1, perform an heuristic search on potential joins rather than for all possible tentative joins exhaustively. For example, only leaves with communication savings when splitting on an identical attribute can form a join that has a possible message length saving. In the worst case, there will be $2N_a C_2^{|L|}$ tentative joins performed in each iteration, where N_a is the number of attributes available for splitting in leaves, $|L|$ is the number of leaves in the iteration, and $C_2^{|L|} = |L|(|L|-1)/2$.

4.4 Experiments

Six artificially generated noisy data sets and two real-world data sets were used in the tests. Half of the artificial data sets were generated from disjunctive underlying functions. Most of them were downloaded from the UCI machine learning repository [18] and had been widely tested in other decision tree or decision graph systems [147, 107, 164]. All the data sets choosen have only discrete attributes without missing values. The test results were compared with the well-known classification programs C4.5 and C5 [164]; the Oliver-Wallace binary join decision graph program [138, 145, 147] was unavailable except for a past result [66] (see section 4.4.4). In addition to the conventional classification accuracy, a metric called probabilistic costing [95, 94, 57, 62] was implemented for comparison. See section 4.8.1 for full details.

4.4.1 Testing with Artificially Generated Data

The experimental results are presented in Table 4.1, "accuracy" describes the classification accuracy and "pr costing" describes Good's probabilistic costing, or logarithmic 'bit costing' [95, 94, 57, 62]. For the data sets on which ten 10-fold

cross-validations were performed, the classification accuracy and probabilistic costing were presented in the form of mean $\pm$ standard deviation, $\mu \pm \sigma$ (where $\sigma^2 = \sum_{i=1}^{N}(x_i - \mu)^2/N$).

4.4.2 Inferring Probabilities for a Multinomial Distribution

Given an array of occurrences of events of an m-state multinomial distribution $(c_1, c_2, \ldots, c_m)$, the probability of a certain event j can be estimated by (either)

$\hat{p_j} = \frac{c_j+0.5}{(\sum c_i)+m/2}$ [p186 (2), p187 (4), p194 (28)][209][p75][217] or

$\hat{p_j} = \frac{c_j+1}{(\sum c_i)+m}$ [p187 (3), p189 (30)][209],

the latter being known as the Laplace estimate and also corresponding (with uniform prior) to both the posterior mean and the minimum expected Kullback-Leibler distance estimator. In the experiments here (see tables 4.1, 4.2 and 4.3), both formulas were used to give the probability estimates in this decision graph program while only the latter (+1) was used to give the probability estimates in C5 and C4.5. The first (+0.5) $\hat{p_j}$ estimate was used in the MML multinomial message length calculations, but both $\hat{p_j}$ estimates were used in calculating the decision graph log-prob bit cost.

XD6 data set: The XD6 data set [165, 224, 147] is an artificial set with 10 binary attributes. It was generated so that a division into two categories according to the boolean function of attributes 1 to 9

(A1 $\wedge$ A2 $\wedge$ A3) $\vee$ (A4 $\wedge$ A5 $\wedge$ A6) $\vee$ (A7 $\wedge$ A8 $\wedge$ A9)

with 10% noise added to the target attribute. 10 data sets, each containing 500 data items, were randomly generated. Then 10 individual 10-fold cross-validations on these data sets were performed. Each 10-fold cross-validation consists of 10 tests. In each test, training was performed on nine-tenths of the data and tested on the remaining one-tenth. This amounted to 10x10=100 tests.

1st monk's data set: The 1st monk's data set is in the UCI machine learning repository [138, 18], and constructed from the noiseless function

(Jacket_Color = Red) V (Head_Shape = body_Shape)

10 independent tests were performed, with each set of training data consisting of 124 data randomly selected from full data set of size 432. The UCI convention was followed on this data set [18] of using the entire data set as test data.

Table 4.1: Test Results - Both "Right"/"Wrong" and Log(Prob) - on Artificial Data-Sets

		D-Graph with M-W joins(+1)	D-Graph with M-W joins(+.5)	C5	C4.5
XD6	accuracy	89.2±1.8%	89.2±1.8%	85.2±2.5%	84.9±2.7%
	pr costing	25.2±2.4bits	27.5±2.7%	30.2±3.3bits	33.6±4.2bits
1st Monk	accuracy	100±0.0%	100±0.0%	75.2±4.3%	75.0±3.9%
	pr costing	9.8±0.0bits	5.0±0.0bits	278.4±12.1bits	285.9±11.3bits
LED-7	accuracy	72.3±3.2%	72.3±3.2%	71.2±3.5%	71.1±3.3%
	pr costing	74.5±7.4bits	72.6±6.2bits	89.7±7.9bits	89.6±7.8bits
LED-24	accuracy	68.5±4.4%	68.5±4.4%	67.8±4.7%	67.8±4.5%
	pr costing	96.4±6.2bits	95.1±6.7%	102.8±7.3bits	103.2±8.0bits
Car	accuracy	91.5±1.7%	91.5±1.7%	91.6±1.9%	91.5±2.4%
	pr costing	44.8±5.6bits	40.5±5.3bits	70.9±6.8bits	72.4±11.9bits
Scale	accuracy	79.2±3.2%	79.2±3.2%	61.9±2.8%	61.4±3.8%
	pr costing	54.6±5.9bits	55.8±6.7bits	83.4±4.7bits	83.3±6.7bits

LED-7 data set and LED-24 data set: The LED-7 data set [18] is an artificially generated data set in the UCI machine learning repository with 7 binary attributes and 10 output classes. Each of the attributes corresponds to one different segment in a Light Emitting Diode that displays the numbers 0 to 9. 10% noise was added to each of the seven input attributes. The only difference between LED-7 and LED-24 is that 17 irrelevant attributes with randomly generated values were deliberately added in the LED-24 data set. Again, 10 data sets, each of them containing 500 data items, were randomly generated and then an individual 10-fold cross-validation was performed on the each of the data sets - giving a total 10x10=100 tests.

Car Evaluation data set: The car evaluation data set from the UCI repository [18] was generated from an underlying decision tree model. There are 1728 instances with four output classes in the set. Each data item has 6 nominal ordered attributes, which are treated as unordered discrete attributes in tests here because data files from UCI have been set in this format. 10 independent 10-fold cross-validations were again performed on the data set.

Balance scale data set: The balance scale data set from the UCI repository [18] was generated to model psychological experimental results. There are 625 instances with 3 output classes in the set. 10 independent 10-fold cross-validations were again performed on the data set.

4.4.3 Discussion of Above Test Results on Artificial Data

As table 1 shows, the decision graph program achieved better or significantly better classification prediction accuracy in most of these test sets (5 out of 6), especially the data sets with a disjunctive underlying function, such as the XD6 and Scale data sets. On the fifth data set (Car), performance here was only 0.1% worse. The multi-way join decision graph program also has substantially lower probabilistic bit costing than both C4.5 and C5 on all the test sets. This might best be explained by the fact that decision graphs are more expressive than decision trees and are often able to use the data more efficiently. This should give a shorter Kullback-Leibler distance from the underlying model [214].

4.4.4 Testing with Real World Data

UCI protein secondary structure database The protein secondary structure database was one of the UCI molecular biology databases [18]. The data contains amino acid chains with a secondary structure specified at each point. Microbiologists can determine the amino acid chain of a protein, but finding the secondary structures - which are "alpha-helix", "beta-sheet" and "random-coil" - is quite difficult. Decision graphs were constructed that predicted the secondary structure at a point in a protein by (following [66] and) using a window of size 7 (centred at the point of interest) of the amino acid chain attributes. Each of these 7 attributes has arity 21. The multi-way join decision graph program, C4.5 and C5 were tested with the default training and test data set downloaded from the UCI repository. As shown in table 4.2, decision graphs with multi-way joins performed better than C4.5 and C5. In particular, the multi-way join decision graph program achieved at least 2.0% higher classification accuracy and at least 215 bits less in probabilistic costing compared to both C4.5 and C5.

Table 4.2: Test Results of UCI Protein Data Set (From Section 4.4.4)

	D-Graph with M-W joins(+1)	D-Graph with M-W joins(+.5)	C5	C4.5
accuracy	57.6%	57.6%	55.4%	55.6%
pr costing	4715.2bits	4718.6bits	4935.1bits	4935.5bits

Another protein data set This protein data set was generated from a protein database [66]. It is used because the Oliver-Wallace program [147, 145, 138] is no longer accessible and it is the only data set for which results from that program are available. C4.5, C5 and the multi-way join coding scheme were tested by performing an 8-fold cross-validation on the data set because this was done in the original paper [66]. Test results are also compared here with results achieved [66] by the Oliver-Wallace decision graph with binary joins. As shown in Table 4.3, decision graphs with multi-way joins performed better 6 times out of 8 than each of C4.5, C5 and the Oliver Wallace binary join decision graph program [147, 145, 138]. On average, MML decision graph scheme achieves 1.6%, 1.5% and 1.2% higher 'right'/'wrong' prediction accuracy respectively than C4.5, C5 and the Oliver-Wallace binary join decision graph program. The multi-way join scheme also has a lower probabilistic bit cost score than both C4.5 and C5 on all 8 test sets [66] in Table 3.

4.5 Internal Repeated Structures and Linked Decision Forests

The decision graph system proposed in section 4.3.1 allows multi-way joins. For a similar scheme, see [129, Appendix]. Directed acyclic graphs were used in the MML decision graph with multi-way joins as in both the Oliver and Wallace system [147, 138, 145] and the Kohavi system [107]. The main idea behind the coding of the decision graphs with multi-way joins is to decompose a decision graph into a sequence of decision trees and joining patterns [186]. In this way, encoding a decision graph is equivalent to encoding a sequence of decision trees and joining patterns in order. An efficient coding scheme for decision graphs can be achieved by

Table 4.3: Prediction accuracies on another protein data set [66] (from Section 4.4.4)

Test Set		D-G with Multi-Way Joins	D-G with binary Joins	C5	C4.5
1	accuracy	53.0%	54.2%	52.4%	52.4%
	pr costing	2296.6bits	2296.7bits	2380.7bits	2380.7bits
2	accuracy	54.6%	53.3%	53.1%	53.1%
	pr costing	1907.7bits	1907.7bits	1975.2bits	1975.2bits
3	accuracy	55.9%	51.7%	54.6%	54.6%
	pr costing	2185.4bits	2186.3bits	2241.7bits	2241.7bits
4	accuracy	58.2%	56.4%	55.8%	55.8%
	pr costing	2066.2bits	2065.9bits	2252.2bits	2252.2bits
5	accuracy	50.2%	46.8%	43.9%	43.9%
	pr costing	2227.9bits	2228.1bits	2439.2bits	2439.2bits
6	accuracy	50.7%	49.0%	50.8%	51.0%
	pr costing	2314.8bits	2316.0bits	2362.3bits	2351.3bits
7	accuracy	53.5%	52.8%	51.4%	50.5%
	pr costing	2218.2bits	2220.3bits	2238.1bits	2295.0bits
8	accuracy	52.9%	54.6%	54.6%	54.6%
	pr costing	2246.3bits	2247.1bits	2281.0bits	2281.0bits
Avg	accuracy	53.6%	52.4%	52.1%	52.0%
	pr costing	2182.9bits	2183.5bits	2271.3bits	2277.0bits

re-using some of the well-proved Wallace-Patrick decision tree coding scheme [224] and devising an efficient coding of the joining patterns [186].

As discussed in the previous sections, decision graphs are able to represent some duplicated sub-concepts efficiently by uniting these subtrees into one tree. However, in many tree learning problems, these subtrees are not entirely identical but rather share repeated internal structures. For example, the tree in Figure 4.6 contains three subtrees with internal repeated structures (involving C and D).

The repeated internal structure problem was first brought forward and studied by Uther and Veloso [194]. Their solution to this problem was to introduce a new representation called the decision linked forest [194], in which the decision tree is turned into a sequence of attribute trees and a root tree. The attribute trees were formed by abstracting the topological structures of the repeated internal structures in the original trees. The attribute trees were then treated as new attributes in the root tree. Their new coding scheme [194] only encodes the repeated sub-concepts once by forming attribute trees as new attributes available to decision trees in the linked decision forest. Their scheme [194] can be explained by Figure 4.7, which

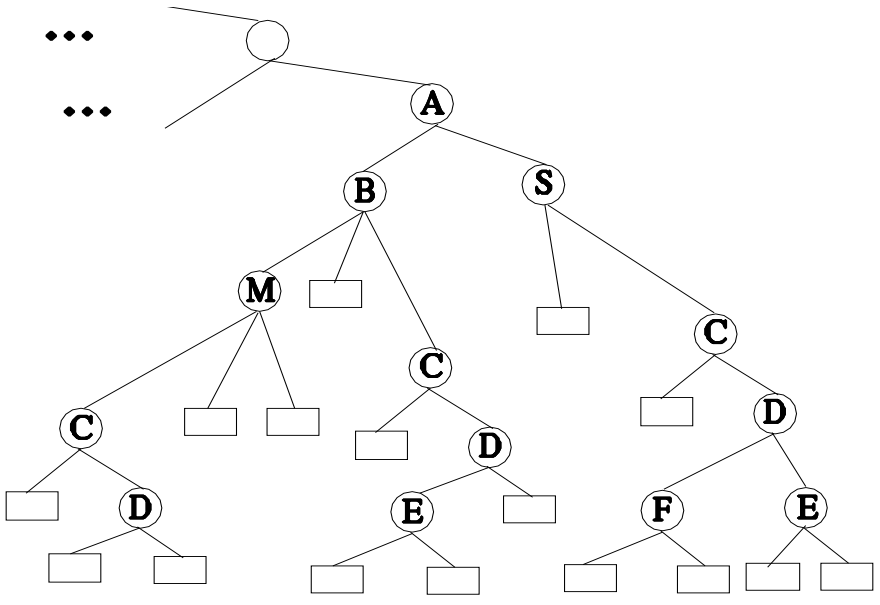


Figure 4.6: A decision tree with internal repeated structures (involving C and D)

shows how linked decision forests provide a more efficient solution to resolve the internal repeated structure problems in Figure 4.6.

4.6 Decision Graphs with Dynamic Attributes

Uther and Veloso's novel use of linked decision forests [194] eliminated inefficient coding of the internal repeated structures in a decision tree. However, the root tree of a linked decision forest, where the inference process occurs, is still a decision tree, so the fragmentation problem of decision trees (which is solved by decision graphs) remains unresolved in the linked decision forest. Uther and Veloso [194] claimed that none of the existing decision graph programs was able to resolve the problem of encoding internal repeated structures. This issue has been addressed here by introducing dynamic attributes in the multi-way joint decision graph, which essentially generalises both decision graphs with multi-way joins [186] and Uther-Veloso linked decision forests [194] - as is explained in detail below.

Whenever there is an M-way join operation in a decision graph, a new attribute is created and is made available for every node in the subtrees under the node resulting from the M-way join operation. From the node, back traces are performed along the M joining routes until they reach the root of the decision graph. Then the root and the M routes define a new attribute with arity M. The purpose of this attribute

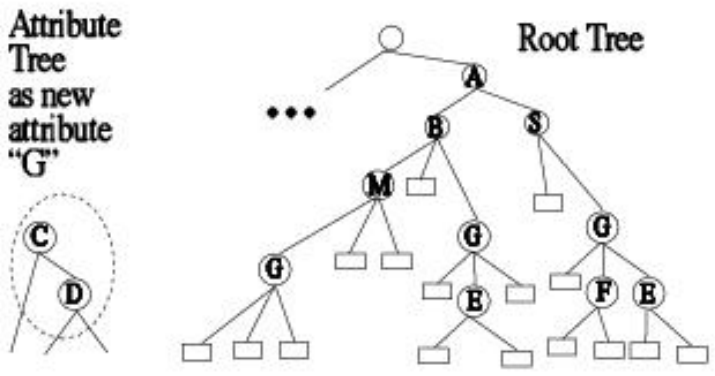


Figure 4.7: A linked decision forest [194] with an attribute tree

is to separate the data into M categories corresponding to the way by which the data arrived at the M-way join. When there are several subtrees in a decision graph with internal repeated structures, they are joined to form one subtree consisting of the internal structure. Then, the leaf nodes where there are differences among corresponding leaves in the original subtrees are split on the new attribute. As such, the decision graphs are able to join subtrees with internal repeated structure (immediately before each one would split on this structure) so that the repeated structure is only encoded once. Once created, the new attribute becomes common knowledge to both sender and receiver and thus there is no cost incurred for its description. (Ideally, the new dynamic attribute is not immediately split on.) This scheme provides a more efficient and elegant solution to this problem than linked decision forests and certainly decision graphs described in much of the literature. A solution to resolve the internal repeated structure problems in Figure 4.6 for these new decision graphs with dynamic attributes is shown in Figure 4.8.

4.7 Growing a Decision Graph

While growing a decision tree, the order in which the leaf nodes are expanded is irrelevant to the resultant tree since splitting a leaf node would have no effect on

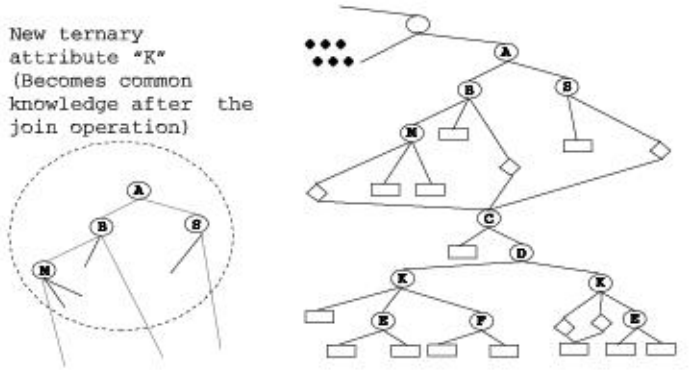


Figure 4.8: A decision graph with a dynamic attribute

the following actions taken on other leaves. However, the order in which the leaf nodes are expanded or joined is often crucial while growing a decision graph. Such a significant difference between decision tree inference and decision graph inference makes the algorithm used in the former inadequate for the latter. In this section, the MML decision graph growing algorithm implemented for Oliver and Wallace's binary join decision graph program [138, 145, 147] is investigated, and a drawback in their search heuristic which makes their program unable to infer the optimal graph in some circumstances is explained. Section 4.7.2 presents the new algorithm for inferring decision graphs with multi-way joins in detail.

4.7.1 Oliver and Wallace's MML Decision Graph Generation Algorithm

Oliver and Wallace's algorithm extends a decision graph by iteratively performing the following procedures until no further improvement can be achieved.

1. For each leaf, L, determine the attribute A on which it should be split. Record, but do not perform, the alteration (Split L on A) along with its saving in message length.

2. For each pair of leaves, L1 and L2, perform a tentative join. Record, but do not perform, the alteration (Join L1 and L2) along with its saving in message length.

3. Choose the alteration (whether from step 1 - a Split, or from step 2 - a Join) that has the greatest saving. If this alteration creates a saving in message length, then perform that alteration on the graph.

Oliveira et al. [137] reported that this algorithm tended to perform premature joins on complex systems and similar observations were obtained in tests here. The example in Figure 4.9 shows how and why this could happen.

Suppose an algorithm decides to grow the decision tree shown on the left of Figure 4.9. For leaf L1, consider splitting on attribute C to save S_1 bits in message length whereas split on attribute E would yield S_1' bits saving in message length. If $S_1' > S_1$, then according to the algorithm above, the alteration that splits L1 on attribute E is recorded. For leaf L2, the same is done for the alteration that

splits L2 on attribute C with S_2 bits saving in message length. When performing a tentative join, the same is done again for the alteration that joins L1 and L2 with S_j bits saving in message length. When estimating the saving from a tentative join, a lookahead search whose aim is to look for the subtree with the minimum message length is conducted on the node resulting from the join. Since expanding the node resulting from joining leaf L1 and leaf L2 would be viewed as merging expanded L1 with expanded leaf L2, so the saving is $S_j = S_1 + S_2 - S_g + S_t$, where S_g is the cost in message length to transmit the join, and S_t is the cost to transmit the topological structure of one of the subtrees. In the case when $S_j > S_1'$, the resultant graph would be the graph shown in the middle of Figure 4.9 instead of the optimal one shown on the right of Figure 4.9. The Oliver and Wallace algorithm has a bias toward joins because it compares the sum of the savings from expanding the two leaf nodes with the saving from expanding just one leaf node. This shows why Oliver and Wallace's decision graph growing algorithm produces premature joins in some circumstances.

4.7.2 The New MML Decision Graph Growing Algorithm

Were the above algorithm adopted for the dynamic attribute decision graph inference scheme, the multi-way joins could only increase the bias toward premature joins. The following algorithm eliminates the premature joins. To grow a decision graph, begin with a graph having one node, with the root being a leaf. Grow the graph by performing the following procedures iteratively until no further improvement can be achieved.

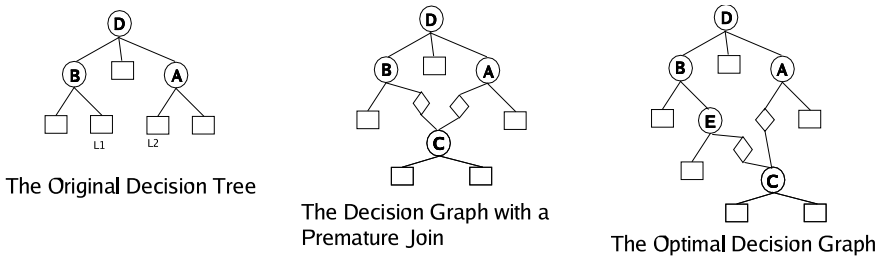


Figure 4.9: An example illustrating how the premature joins are generated

1. For each leaf L, perform tentative splits on each available attribute in the leaf, and determine the attribute A that will lead to the shortest message length when L is split on A. Record, but do not perform, the alteration (Split L on A) along with its rate of communication saving - the communication saving divided by the number of data items in the leaf.

2. For each leaf L, perform tentative joins with other leaves. Record, but do not perform, the alterations (join L_i and L_j; ...; join $L_i, L_j, \ldots, L_k$; etc.) along with its rate of communication savings - the communication saving divided by the number of data items in the join.

3. Sort the alterations from step 1 and step 2 by their communication savings rates. Choose the alteration (whether from step 1 or from step 2) that has greatest rate of saving.

When splitting on any continuous-valued attributes, a simple single cut-point search algorithm is employed, in which the information gained from the cut is the objective function. Then the cost in message length to state the cut-point is log(the number of values of the attribute in this node - 1). In each iteration, the data (i.e., split a leaf or join leaves) is manipulated so that the greatest rate of saving in message length can be achieved. Thus, it is guaranteed that the later iterations will generate a decision graph better or not worse than the possible optimal graph expected in the current iteration. Of course, the algorithm with this search heuristic is only locally optimal.

4.8 Experiments

One artificially generated data set and eight real-world data sets were used in tests. The only artificial data set is the XD6 data set [165, 224, 147, 186], which consists of 10 (9 input, 1 output) binary attributes. It was generated according to the boolean function of attributes 1 to 9:

(A1 $\wedge$ A2 $\wedge$ A3) $\vee$ (A4 $\wedge$ A5 $\wedge$ A6) $\vee$ (A7 $\wedge$ A8 $\wedge$ A9)

with 10% noise added to the target attribute. The other eight real-world data sets from the UCI machine learning repository [18] have been widely tested in other decision tree or decision graph systems [147, 107, 164]. In order to rigorously examine

Table 4.4: Summary of Data Sets

Data-set Name	size	Discr. Attrib.	Cont. Attrib.	No. of Classes	Run Time (C4.5, C5)	Run Time dGraph[186]	Run Time dG (dyn atts)
abalone	4177	1	7	29	1.35s	1062s	1132s
car	1728	6	0	4	0.03s	2.07s	2.42s
cmc	1473	8	2	2	0.06s	11.0s	13.7s
credit	690	9	6	2	0.04s	29.9s	38.9s
led	500	7	0	10	0.01s	4.50s	5.90s
scale	625	4	0	3	0.01s	1.10s	1.70s
tic-tac-toe	958	9	0	3	0.01s	3152s	4031s
vote	435	16	0	2	0.00s	0.01s	0.01s
XD6	500	9	0	2	0.01s	2.55s	3.22s

the proposed algorithms, 10 10-fold cross-validations were performed on each of the nine data sets. This amounted to 10x10=100 tests for one single data set. Each pair of training/test data from these tests was fed into four different decision tree and graph algorithms: the well-known decision tree classification programs C4.5, C5 [164], the decision graph with multi-way joins [186] and the new decision graphs with multi-way joins and dynamic attributes.

The experimental results are presented in tables 4.4, 4.5, 4.6 and 4.7. In table 4.4, the average run time records the execution time of one test by the algorithms on a PIII 1G Linux Redhat7.3 PC. In table 4.5, "Accuracy" describes the rate of "right"/"wrong" classification accuracy. In table 4.6, "pr costing" describes Good's (binomial) probabilistic costing [94, 95], or logarithmic 'bit costing' [57, 62, 186, 95, 94, 128]. Table 4.7 compares the size of resultant decision trees and graphs by recording the number of leaf nodes in them. For the data sets on which 10 10-fold cross-validations were performed, the rate of classification errors and probabilistic costings are presented as mean $\pm$ standard deviation, $\mu \pm \sigma$.

4.8.1 Comparing and Scoring Probabilistic Predictions

Decision trees and graphs are often used as classifiers in many machine learning problems. In the case where the target attribute is multinomial, each leaf node in

a tree or graph is given a class label corresponding to the class with the highest inferred probability for this node. However, the multinomial distribution in each leaf node can also be interpreted as a probabilistic prediction model. In this way, the decision trees and decision graphs are not only classifiers, but can also provide a probabilistic prediction model. Provost and Domingos [159] showed that with some modifications, tree induction programs can produce very high quality probability estimation trees (PETs). Perlich, Provost and Simonoff [153] also observed that for large data sets, tree induction often produces probability-based rankings that are superior to those generated by logistic regression. Thus, in addition to the conventional classification accuracy, a metric called probabilistic costing [95, 94, 57, 62, 186, 128] was implemented in the tests for comparisons of probabilistic predictions with C4.5 and C5. It is defined as $-\sum_{i=1}^{n} \log(p_i)$, where n is the total number of test data and p_i is the predicted probability of the true class associated with the corresponding data item [95, 94, 57, 62]. The reader can interpret the metric as the optimal coding length for the test data given the resultant tree and graph models. This metric can be used to approximate (within a constant) the Kullback-Leibler distance between the true (test) model and the inferred model. Its relation to log-likelihood via $-\log(\prod_{i=1}^{n} p_i) = -\sum_{i=1}^{n} \log(p_i)$, its relation to Kullback-Leibler distance and its corresponding general applicability to a wide range of probability distributions (recall section 5.5) [95, 94, 57, 62, 186, 128] strongly recommend this log(prob) bit costing as a statistically-based general alternative to metrics such as ROC and AUC (Area Under Curve) [159, 153]. Furthermore, as per sec. 4.4, log(prob) scoring is unique in being (to within a constant) the only scoring system which remains invariant to the re-parametrising of the problem [49, footnote 175][51, sec. 3.2][52, sec 4.1].

Logarithm of probability (bit) scoring, Pr_cost, enables us to compare probabilistic prediction accuracy of inferences from an identical training data set by various decision tree and graph algorithms. The lower the value of the Pr_cost, the more consistent the predicted probabilistic model is with the true model.

Given an array of occurrences of events of an m-state multinomial distribution $(c_1, c_2, \ldots, c_m)$, the probability of a certain event j can be estimated by (either)

Table 4.5: Test Results ("right"/"wrong" Accuracy) %

Data-set Name	C4.5	C5	dGraph with M-way joins [186]	dGraph with dynamic atts
abalone	78.9 ± 1.9	79.0 ±1.8	74.3 ±2.1	74.3 ± 2.1
car	7.8 ± 2.2	7.8±2.1	8.5 ±2.8	6.7 ± 2.8
cmc	48.2 ±3.6	48.5 ± 3.6	48.4 ±3.6	48.2 ±3.6
credit	14.4 ±3.6	14.5 ± 3.6	14.2±4.3	14.2 ±4.3
led	30.0 ±5.2	30.0 ± 5.2	30.0±5.8	30.0 ±5.8
scale	35.6 ±4.9	35.4 ± 3.3	22.0±5.3	22.0 ±5.3
tic-tac-toe	14.4 ±3.4	14.0 ± 3.5	11.9±4.8	10.7 ±4.9
vote	5.0 ± 3.1	5.0 ± 3.1	4.4±3.3	4.4 ± 3.3
XD6	14.1 ±4.9	14.2 ± 5.0	9.2±4.0	9.2 ±4.0

$\hat{p_j} = \frac{c_j+0.5}{(\sum c_i)+m/2}$ [209, p187(4);p194(28);p186(2)] [218] [217, p75][186] or

$\hat{p_j} = \frac{c_j+1}{(\sum c_i)+m}$ [209, p187(3);p189(30)][186], the latter being known as the Laplace estimate and also corresponding (with uniform prior) to both the posterior mean and the minimum expected Kullback-Leibler distance estimator [214]. In these experiments, the first (+0.5) was used in MML multinomial message length calculations, $\hat{p_j}$ estimations and calculations of the log(prob) bit costing.

4.8.2 Discussions of Above Test Results

Tables 4.5 and 4.6 clearly show the decision graph with dynamic attributes to always be either outright first or (sometimes) equal first. When testing on the data sets with disjunctions (like abalone, scale, tic-tac-toe and XD6), the decision graph with dynamic attributes has a much lower error rate. On other data sets, it returns results not worse than those from C4.5 and C5. Results from the decision graph with dynamic attributes are either identical or marginally better than those from the decision graphs with multi-way joins [186]. In the cases that the decision graph with dynamic attributes performs better, generated dynamic attributes are found in the resultant graphs. This proves the importance of the dynamic attributes and also shows that decision graphs with dynamic attributes are a superset of decision

Table 4.6: Test Results ($-\log(Prob)$ Costing) bits

Data-set Name	C4.5 (+0.5)	C5 (+0.5)	dGraph with M-way joins [186] (+0.5)	dGraph with dyn atts (+0.5)
abalone	1810.3 ± 26.3	1814.5±25.9	1269.6±32.0	1269.8±33.4
car	60.0 ± 9.9	61.2±9.8	49.8 ±10.5	40.7 ±12.0
cmc	221.4±13.3	222.1 ±13.4	202.4±9.9	202.2 ±9.9
credit	38.3 ±8.1	38.4 ± 8.0	35.2±7.5	35.2 ±7.5
led	79.3 ±9.8	79.3±9.7	76.2±10.0	76.2 ±10.0
scale	82.9 ±6.8	80.1±6.6	55.0±10.0	55.0 ±10.0
tic-tac-toe	46.4 ±8.4	45.4 ±7.1	44.7±13.5	41.1 ±14.7
vote	9.8 ± 6.2	9.8 ±6.1	8.6±5.5	8.6 ± 5.5
XD6	28.5 ±7.8	28.4 ±7.1	22.4±6.7	22.4 ±6.7

graphs with multi-way joins [186]. In table 4.6, the Pr_cost from both kinds of decision graphs are clearly lower than those from both C4.5 and C5. This suggests that the decision graphs inferred by MML are resistant to over-fitting. As such, the decision graphs not only produce excellent "right"/"wrong" predictions, but also provide inferred probabilistic models that are clearly more consistent with the test data (cf. also tables 4.1 to 4.3).

In table 4.7, the resultant multi-way join decision graphs [186] are similar in size to the decision graphs with dynamic attributes. The sizes of the resultant graphs tend to be substantially smaller than the sizes of both the C4.5 and C5 trees. From table 4.4, both kinds of MML decision graphs take longer to infer than C4.5 and C5 trees. Tables 4.5, 4.6 and 4.7 suggest it is well worth the wait. Nonetheless, trimming the decision graph searches might be beneficial.

4.9 Summary

This chapter introduced a refined coding scheme for decision graphs which allows multi-way joins. The use of the MML principle and the new coding scheme to infer (multi-way join) decision graphs was discussed. Experimental results demonstrated

Table 4.7: Size of Resultant Tree or Graph (Number of leaf nodes)

Data-set Name	C4.5	C5	dGraph with M-Way joins [186]	dGraph with dynamic atts
abalone	2103	1062	315	313
car	170	122	36	38
cmc	230	132	10	10
credit	34	15	7	7
led	42	22	22	22
scale	38	34	21	21
tic-tac-toe	132	88	37	35
vote	16	8	5	5
XD6	52	25	5	5

that the refined coding scheme compared favourably with other decision tree inference schemes, namely C4.5, C5 and the Oliver-Wallace binary join decision graph. This pronounced favourable comparison holds both for 'right'/'wrong' prediction accuracy and I.J. Good's logarithm of probability bit costing, and both for artificially generated and real-world data.

Further refinements of the decision graph with multi-way joins representation by introducing dynamic attributes in the decision graphs were presented. Using the MML principle, an improved coding scheme for inferring the new decision graphs has been devised to address some of the inefficiencies in previous decision tree and decision graph coding schemes. The experimental results demonstrated that the refined coding scheme compares favourably with other decision tree inference schemes, namely both C4.5 and C5. This favourable comparison holds true both for 'right'/'wrong' prediction accuracy and especially for I.J. Good's logarithm of probability bit costing (recall section 4.8.1 and table 4.6), as well as for both artificially generated and real-world data.

Future work could both speed up the decision graph searches and compare with more programs, such as, e.g., Yin and Han's CPAR [229].

Chapter 5

MML Decision Forests

Ensemble learning schemes have shown impressive increases in prediction accuracy over single model schemes. This chapter introduces a new decision forest learning scheme, whose base learners are Minimum Message Length (MML) oblique decision trees. Unlike some other tree inference algorithms, MML oblique decision tree learning does not over-grow the inferred trees. The resultant trees thus tend to be shallow and do not require pruning. MML decision trees are known to be resistant to over-fitting and excellent for probabilistic predictions. A novel weighted averaging scheme is also proposed which takes advantage of high probabilistic prediction accuracy produced by MML oblique decision trees. The experimental results show that the new weighted averaging offers solid improvement over other averaging schemes, such as majority vote. This MML decision forests scheme also returns favourable results compared to other ensemble learning algorithms on data sets with binary classes. This chapter also investigates various aspects of building classification models from microarray data with tree-based classification algorithms by using Partial Least-Squares (PLS) regression as a feature selection method. Experimental results show that the PLS regression method is an appropriate feature selection method and MML decision forests are capable of delivering high performance classification models for microarray data.

5.1 Introduction

Ensemble learning is one of the major advances in supervised learning research in recent years [47]. The outputs of an ensemble classifier are determined by a committee, in which a group of classifiers cast (possibly weighted) votes on final predictions [113]. Generally, ensemble learning schemes are able to outperform single classifiers in predictive accuracy. The intuitive explanation for the success of ensemble learning is that mistakes made by individual classifiers are corrected by complementary results submitted by other classifiers in the committee.

The most widely adopted approaches are called Perturb and Combine (P&C) methods. P&C methods infer each classifier from a set of distinct training sets, which are generated by perturbing the unaltered original training set. An important prerequisite for the P&C methods is that the base learner must be unstable in that the inferred models are sensitive to even minor variation of the training set. Bagging (bootstrap aggregating) [29] and AdaBoost (adaptive boosting) [89] are two popular P&C methods implemented in many ensemble learning schemes. Both methods are able to generate diverse committees by feeding base learners with a set of distinct training sets drawn from the original training set.

Another way to create variations in the training sets is to alter the label of the target attribute of instances in the training set. Breiman proposed a scheme [31] which grows a set of decision trees by injecting random noise into the output label of the original training set. DECORATE (Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples) [121] is another ensemble learning scheme which has a similar motivation. But, instead of randomly altering the output labels, the algorithm inserts the artificially constructed instances (and adds to the training set) with the aim of deliberately increasing diversity among the inferred committee members.

Decision tree inducers are unstable in that resultant trees are sensitive to minor perturbations in the training data set. Largely for this reason, decision trees are widely applied as the base learners in ensemble learning schemes. Some ensemble algorithms have implemented modified decision tree inference algorithms in order to generate diverse decision forests. Ho proposed a scheme called the random subspace

method [99] for constructing decision forests. When the algorithm constructs a split at each internal node of the inferred trees, candidate features are restricted to a randomly selected subset of the original input features. Dietterich introduced another ensemble learning scheme [48] that does not rely on the instability of the decision tree inducer. Instead of picking the split with the best score on the objective function, the algorithm randomly chooses a split among a pre-defined number of candidate splits with the highest score (on the objective function). Ferri et al. [80] have an interesting scheme in which the subset of the data that was deemed most difficult to classify is put aside and then delegated to another run of the classifier.

Research on combining some of the above methods to further improve the performance of ensemble learning has also shown promising results. In the random forests scheme developed in [32], the bagging method and random feature selection schemes were implemented to inject randomness into the decision tree growing processes.

Given that different ensemble schemes have different effects on the inferred committee members, it is natural to explore the benefits of building hybrids of various ensemble learning schemes to complement predictions submitted by classifiers from various schemes. Both MultiBoosting [225] and Multistrategy [226] were introduced by Webb. While MultiBoosting can be considered as bagging committees formed by AdaBoost, Multistrategy further extended this idea by constructing a hierarchy of various ensemble learning schemes. Ensemble using other classification methods as base learners were also be proposed [168]. Bonissone et al. [19] presented an ensemble scheme based on fuzzy decision trees called FRF ensemble. Neural networks ensembles are also proposed in [169, 93].

While there are no clear winners emerging from the above ensemble schemes, all of them reported superior "right"/"wrong" predictive accuracy compared to single classifier learning schemes. In this chapter, a new ensemble algorithm called decision forests with oblique decision trees is proposed. The proposed ensemble learning scheme is different from other random forests in several ways. While most of the ensemble learning algorithms grow deep and unpruned decision trees, the base learner in our ensemble learning is MML oblique decision trees, which were introduced in [188]. Menze et al. [122] showed that oblique random forests perform better than standard random forests on a large range of data. This chapter also shows how

to include simple probabilistic Support Vector Machines in the internal and/or leaf nodes of decision trees. A MML coding scheme is applied to select optimal candidate trees (with overall lower MML coding) with high probabilistic prediction accuracy and smaller tree size (lower height with fewer leaf nodes). Compared to the schemes with univariate trees (which cut on only one attribute at a time), using MML (multivariate) oblique trees offers potential to greatly increase the diversity of the inferred forest. A new weighted averaging scheme is also proposed. The proposed averaging scheme is based on Bayesian weighted tree averaging but uses a modified, smoothed prior on decision trees (see section 5.3.3). In order to take advantage of the above weighted averaging scheme, a new algorithm to rapidly generate a large number of distinct oblique decision trees is introduced.

5.2 Details of Some Related Ensemble Schemes

5.2.1 Bagging

One of the well known Perturb and Combine (P&C) methods is bagging. Bagging [29] relies on perturbing the training set. When unstable learning algorithms are applied as base inducers, a diverse ensemble can be generated by feeding the base learner with training sets re-sampled from the original training set. Letting the original training data set S have N data instances, bagging uses the following procedure to generate a group of T distinct training sets:

1. Assign probability p=1/N to each of the instances in the set S.
2. Sample with replacement from set S and put the sampled instances into a re-sampled set S^i.
3. Repeat step 2 N times until a re-sampled set S^i with N instances is formed.
4. Repeat steps 2 and 3 T times until a group of T distinct training sets $(S^1, S^2, \ldots, S^T)$ is generated, with i ranging from 1 to T.

When the above procedure is finished, the algorithm then infers T classifiers from the group of T distinct training sets, from step 4.

5.2.2 AdaBoost

Another type of P&C method is the AdaBoost algorithm, which is also referred to as an arcing (adaptive re-sampling and combining) algorithm by Breiman in [30]. The fundamental difference between bagging and AdaBoost is that while bagging is non-deterministic, AdaBoost is deterministic and iterative.

AdaBoost iteratively alters the probability over instances in the training set while performing the re-sampling. It works very well when the data is noise free and the number of training data is large. But when noise is present in the training sets, or the number of training data is limited, AdaBoost does not perform as well as bagging and (see below) random forests.

5.2.3 Random Forests

Random forests use CART [33] as the base learner and combine several methods to generate a diverse ensemble. Each decision tree in a random forest is trained on a distinct and random data set re-sampled from the original training set, using the same procedure as bagging. While selecting a split at each internal node during the tree growing process, a random set of features is formed by either choosing a subset of input variables or constructing a small group of variables formed by linear combinations of input variables. Random forests [32] have achieved "right"/"wrong" predictive accuracy comparable to that of AdaBoost and much better results on noisy data sets. Breiman also claimed and showed that AdaBoost is a form of random forest (algorithm) [32].

5.3 Ensemble Learning with MML Random Forests

It has been shown that the performance of an ensemble classifier depends on the strength of individual classifiers and correlations among them [32]. MML oblique trees are shown to return excellent accuracies, especially on probabilistic predictions. The algorithm beats both C4.5 [164] and C5 on both "right"/"wrong" and especially probabilistic predictions with smaller trees (i.e., fewer leaf nodes) [188]. In order to implement a Bayesian averaging scheme, performance of individual classifiers on

probabilistic prediction is crucial. Therefore MML oblique trees are chosen as the base learners in algorithms here. Due to the introduction of hyperplanes at internal nodes, the space of candidate trees is also hugely enlarged. This helps to increase diversity among the trees, especially for the data sets with fewer input attributes.

5.3.1 Searching For Optimal Splits at Internal Nodes

The algorithm proposed here is tailored for searching for optimal two-dimensional hyperplanes (linear combinations of two input attributes). It works as follows:
Firstly, a random two dimensional hyperplane is generated. Then the hyperplane is rotated by 10 degrees each time, so that a set of 18 orientations is generated. For each hyperplane in the 18 orientations, a maximum of 32 cut-points were tested and the one with the minimum total code length is recorded. The process is repeated on all candidate combinations of two input attributes. The total code length is given as below. For further details of the MML coding of oblique decision trees, please see [188].
Total code length $= Part1 + Part2$,

$$Part1 = -2(D-1)\log(\frac{2}{\sqrt{D||w||^2+4}}) + \log\binom{N}{2},$$

$$Part2 = \sum_{l=1}^{L} Msg_l,$$

$$Msg_l = \frac{M-1}{2}(\log\frac{N_l}{12}+1) - \log(M-1)! - \sum_{m=1}^{M}(n_m+\tfrac{1}{2})\log(\frac{n_m+\frac{1}{2}}{N_l+\frac{M}{2}})$$

where D is the dimension of the hyperplane, N is the number of data at the internal node to be split, L is the number of child nodes resulting from the split (in this case L is 2), Msg_l is the code length for encoding data in each resultant lth leaf node, M is the number of classes in the data, n_m is the number of instances in class m in the leaf node and N_l is the number of data in leaf node l. For the remainder of this chapter, $D = 2$.

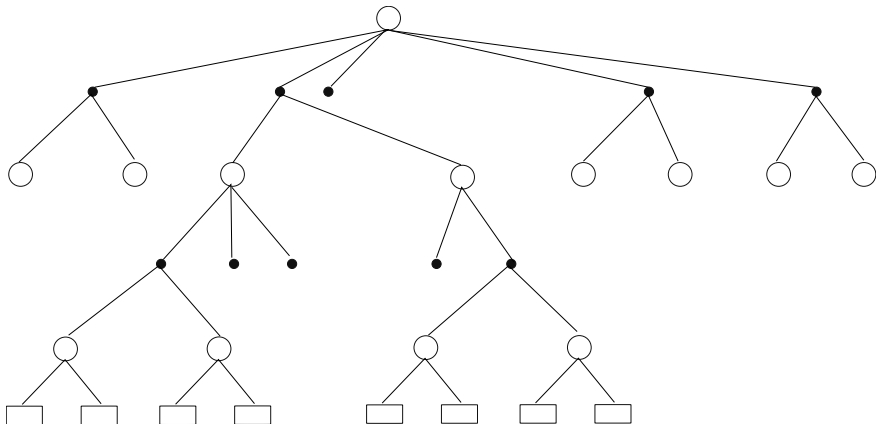


Figure 5.1: A decision tree is generated by picking a random path in the search tree.

5.3.2 An Efficient Algorithm to Generate a Large Number of MML Oblique Trees

The idea behind this new rapid forest generation algorithm is that, at each node, a specified number of viable splits (splits which improve the message length) are recorded. For each of these candidate splits, tentative splits are performed. The above procedure is recursively run on each resulting child node.

The forest growing process is divided into the following two parts (A and B):
A: In the first part, a search tree is constructed using the following steps:
Start the tree with a single leaf node as the root node,

1. Generate a set of possible combinations of two input attributes.

2. Search for the best split by hyperplanes (i.e., those yielding the shortest two-part message length) constructed from each of the given combinations of two attributes for this node.

3. Record each of the candidate splits from step 2 that achieve better message length than the unsplit leaf node.

4. For each of the splits recorded in step 3, perform a tentative split.

5. Recursively apply this procedure on each of the child nodes generated in step 4, until the height of the search tree is H.

B: In the second part, a group of decision trees is generated by randomly selecting a path in the search tree, which is shown in figure 5.1. The candidate splits are represented by the solid dots in the picture, while the resultant subtrees of these splits are represented by the circles (internal nodes) and rectangles (leaf nodes).

In order to approximate the number of searches and the number of distinct trees that are able to be generated from a search tree, it is assumed that there are M viable candidate binary splits at each internal node to be searched. For a search tree with height H, it is easy to see that the number of searches is $S(H, M) = 2M \times S(H-1, M)$ and $S(2, M) = M$, thus $S(H, M) = 2^{H-2}M^{H-1}, H > 1$. The number of distinct trees that can be generated from the search tree can be estimated by using the fact that $T(2, M) = M$ and $T(H, M) \approx T(H-1, M)^M$, thus $T(H, M) \approx M^{(2^{H-1}-1)}$. To keep the computational time reasonable, the algorithm puts some upper limits on M and H. In experiments reported here, M ranges from 25 to 50 while H ranges from 4 to 5.

5.3.3 Weighted Averaging of Trees

Oliver and Hand proposed a Bayesian weighted tree averaging scheme [146] in which the weights are set to be proportional to the posterior probability of each tree in the forest. Given a tree, T_i, with I internal nodes, L leaves and C classes, they give the posterior probability of the tree T_i as

$$P(T_i|D) \propto \prod_{i=1}^{I} \frac{1}{ap(i)} \prod_{i=1}^{L} (1 - \frac{1}{ap(i)}) \prod_{i=1}^{L} \frac{\prod_{j=1}^{C} M_j!}{(\sum_{j=1}^{C} M_j)!} \quad \ldots (2)$$

where $ap(i)$ is the arity of the parent of node i, M_j is the number of the instances belonging to class j (j$\in \{1, 2, \ldots, C\}$) in each leaf node. After initially implementing a similar scheme as an averaging method here, experimental results returned by this averaging method were worse than using simple arithmetic vote averaging. Investigation showed that even in forests with 1000 trees, there are always 2 to 3 trees dominating the majority (.8 to .9) of the weights. While such results may seem to be contradictory to Bayesianism, there are several possible explanations for this. One possible reason is that the prior of decision trees given by

$P(T_i|D) \propto \prod_{i=1}^{I} \frac{1}{ap(i)} \prod_{i=1}^{L}(1 - \frac{1}{ap(i)})$ is not right for the oblique decision trees in the forest. Another possible explanation lies in the fact that there are high correlations between the predictions submitted by the trees with the top posterior probabilities, despite their distinct tree structures.

Attempting to approximate the correct weights for decision trees generated from the real posterior probabilities, a new decision tree averaging scheme for the decision forests algorithm is proposed. Because there are an indefinite number of trees in the posterior space and the real distribution of the posterior probabilities is unknown, three assumptions have been made. Firstly, assume that the real posterior distribution is a unit normal distribution. Secondly, assume that the posterior $P(T_i|D)$ of the inferred trees can be sampled from the range (-R, R), given that $\int_{-R}^{R} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx \approx 1$, R is set to 3.5 in our tests. Lastly, assume that the order of the real posterior probabilities of inferred trees is identical of that of the posterior probabilities obtained from (2). Then the weight of a decision tree, $w(T_i)$, can be calculated as follows:

1. Generate a set of weights $\{w_1, w_2, \ldots, w_S\}$, so that $w_i = \int_{x_i}^{x_i+\Delta x} \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} dx$, where $x_i = -R + (i-1)\Delta x$, $\Delta x = \frac{2R}{S}$
2. Normalize and sort $\{w_1, w_2, \ldots, w_S\}$ so that $w_1 \leq w_2 \ldots \leq w_S$ and $\sum w_i = 1$,
3. Sort the set of trees $\{T_1, T_2, \ldots, T_S\}$ so that
$P(T_1|D) \leq P(T_2|D) \ldots \leq P(T_S|D)$
4. Set the weights for the trees T_i, $w(T_i) = w_i$.

5.4 Experimental Results

5.4.1 Data Sets

Experiments were run on 13 data sets from the UCI data repository [18], some of which were also tested in Breiman's random forest paper [32]. For each data set, 10 independent 10-fold cross-validation tests were performed. A summary of the data sets is shown in table 5.1.

Table 5.1: Summary of the data sets.

Data set	Size	Attributes	Classes
Breast	286	9	2
Breast-Wins	699	9	2
Bupa	345	6	2
Cleveland	303	13	2
Ecoli	336	7	8
German	1000	24	2
Glass	214	9	6
Image	2310	19	7
Ionosphere	351	34	2
Pima	768	8	2
Sat-Images	6435	36	6
Sonar	208	60	2
Vowel	990	10	11

5.4.2 Comparing and Scoring Probabilistic Predictions

In addition to the traditional "right"/"wrong" accuracy, the probabilistic performance [187, sec 5.1] [62, 57, 128, 186] of the ensemble algorithms is often of importance. Recall from section 3.3.2, that probabilistic costing is defined as $P_{cost} = -\sum_{i=1}^{N} \log(p_i)$ and the probability p_j of an instance belonging to class j in a leaf node was estimated by $\hat{p_j} = \frac{N_j+1}{(\sum N_j)+M}$, where N_j is the number of instances belonging to the class j in the leaf node, M is the number of the classes.

5.4.3 Comparisons with Other Ensemble Algorithms

Results for the learning scheme presented here are compared with two other prominent ensemble learning algorithms – AdaBoost and Random Forest – for which 10 independent 10-fold cross-validations tests (averaging over 10x10=100 runs) on the 13 data sets were also run. The random forests algorithm is also available in weka3.4.6 [227]. At each test, a random forest with 1000 decision trees, whose internal node contains a linear combination of 2 input attributes attributes, was grown. C5.0 is a commercial version of the C4.5 [164] program and was created by Ross

Table 5.2: "Right/Wrong" classification accuracies and probabilistic costing results for forests with MML oblique trees, AdaBoost and random forests.

Data set	MML Forest "R/W"	C5 AdaBoost "R/W"	Random Forest "R/W"	MML Forest P_{cost}	C5 AdaBoost P_{cost}	Random Forest P_{cost}
Breast*	**73.4**	72.9	71.0	**23.4**	23.6	N/A
Breast-Wins*	**97.4**	96.3	96.7	**10.2**	18.0	N/A
Bupa*	67.9	68.5	**73.0**	30.9	**30.4**	N/A
Cleveland*	**83.4**	80.6	82.5	**17.7**	19.0	N/A
Ecoli	85.9	84.0	**86.2**	**26.4**	45.6	N/A
German*	74.1	75.4	**77.1**	74.7	**74.5**	N/A
Glass	67.1	74.9	**80.8**	**28.9**	30.3	N/A
Image	92.0	97.9	**98.1**	**156.3**	176.5	N/A
Ionosphere*	**93.8**	93.6	92.8	**12.5**	**12.5**	N/A
Pima*	**76.2**	75.3	75.4	**53.8**	56.2	N/A
Sat-Images	83.3	90.7	**91.5**	**411.4**	574.3	N/A
Sonar*	81.9	80.8	**84.0**	**12.5**	13.3	N/A
Vowel	64.3	89.9	**97.0**	217.7	**176.5**	N/A

Quinlan. To obtain the results for AdaBoost, tests were run using the built-in AdaBoost function of C5.0 with a maximum runs of 1000. In most cases, AdaBoost finished before 1000 runs due to diminutive or no gain in the subsequent runs. The results for MML forests are returned by forests with 1000 trees. MML oblique trees are averaged by weighted averaging of class probabilities, see section 5.3.3.

The results for the "right"/"wrong" classification accuracies are shown in table 5.2. For the 8 data sets with binary classes (asterisked, recalling table 5.1), in terms of "right"/"wrong" accuracy, the MML forests perform best on 5 out of 8 sets, second best on 1 and worst on 2; and on logarithm of probability (P_{cost}) bit costing (recall sec. 5.4.2), the MML forests win 5, tie on 1 and lose the other 2. The results on the 5 multiple-class data sets are interesting. In "right"/"wrong" scoring, random forests win 4 out of 5 and come second in 1 out of 5. But in logarithm of probability scoring, MML forests win 4 out of 5 cases. The most probable explanation is that the base learner, MML oblique decision trees [188], does not perform well in "right"/"wrong" scoring on data sets with multiple classes. Nevertheless, one way to improve the

performance on data sets with multiple classes might be to convert a multiple class learning problem into a set of binary class learning problems.

To compare the performance on probabilistic predictions of ensemble learning algorithms, the P_{cost} for C5.0 AdaBoost, the probabilistic prediction for each test instance is calculated by arithmetic averaging of the probabilistic predictions submitted by each iteration. Unfortunately, probabilistic costing for random forests from weka [227] cannot be obtained. Table 5.2 shows that MML forests have achieved better (lower) or equal probabilistic costing in 9 out of 13 data sets compared to C5.0 AdaBoost. The superior performance on probabilistic prediction of the MML forests can be attributed to the fact that both the base learner algorithm and the averaging scheme are well-suited to probabilistic predictions.

5.5 Models for Microarray Data

DNA microarrays measure a large quantity (often in the thousands or even tens of thousands) of gene expressions of several samples simultaneously. The collected data from DNA microarrays, which contains either expressed or non-expressed values of measured genes, are often called microarray data sets. Microarrays are useful for genome annotation [135], and research microarrarys have also been used by direct-to-consumer genomic service to enable consumers to gain access to their DNA [154]. Advancing statistical methods and machine learning techniques have played important roles in analysing microarray data sets. Results from such analyses have been fruitful and have provided powerful tools for studying the mechanism of gene interaction and regulation for oncological and other studies.

Among bioinformatics research concerned with microarray data, two areas have been extensively studied. One is to design algorithms to select a small subset of genes most relevant to the target concept among a large number of genes for further scrutinising. Another popular research topic is to construct effective predictors which are capable of producing highly accurate predictions based on diagnostic or prognostic data. Both support vector machine and random forest have been used to classify gene expression microarrays and both showed impressive results [179].

However, due to the nature of the collection of microarray data, a microarray data set usually has a very limited number of samples. In a typical gene expression profile, the number of gene expressions (input variables) is substantially larger than the size of samples. Most standard statistical methods and machine learning algorithms are unable to cope with microarray data because these methods and algorithms require the number of instances in a data set to be larger than the number of input variables. Therefore, many machine learning articles have proposed modified statistical methods and machine learning algorithms tailored to microarray analyses. As such, many proposed classification algorithms for microarray data have adopted various hybrid schemes. In these algorithms, the classification process usually has two steps, which are now outlined.

In the first step, the original gene expression data is fed into a dimensionality reduction algorithm, which reduces the number of input variables by either filtering out a larger amount of irrelevant input variables or building a small number of linear or nonlinear combinations from the original set of input variables. The former approach is often known as *variable selection* while the latter is often known as *feature selection*. In the second step, classification models are trained on the data set with a reduced number of input attributes (created in the previous step) using an ordinary supervised classification algorithm.

In principle, many dimensionality reduction algorithms for supervised learning can be applied to the classification of gene expression data. Various two-step schemes have been presented and all of them reported improved classification accuracy. However, in practice, end results are dependent on the combination of the dimensionality reduction algorithm and the classification algorithm. There is no conclusion from previous studies so far which confirms superiority of any particular scheme for microarray data classification.

In this chapter, improved predictive accuracy is attempted by building a hybrid classification scheme for microarray data sets. In the first step, two different dimensionality reduction schemes are implemented: (i) Partial Least-Squares (PLS) regression [90, 100] as the dimensionality reduction algorithm, and (ii) an alternative and novel hybrid feature selection scheme which consecutively applies the

discretization method from [79] on the original data sets followed by the PLS regression algorithm. Then in the second step, the two sets of filtered data with new features resulting from the two feature selection schemes described in the first step are separately fed into tree-based classification algorithms. These two schemes (in Tables 5.5 and 5.6 respectively) are then used to compare the results from four tree-based classification algorithms – C4.5 [164], AdaBoost [88, 89], Random Forests [32] and MML decision forests [189].

5.6 Dimensionality Reduction for Microarray Data

As discussed earlier, various dimensionality reduction algorithms have been proposed for the task of dimensionality reduction of microarray data. Fayyad and Irani's discretisation method [79] discretises continuous-valued attributes by recursively applying an entropy minimisation heuristic. Tan and Gilbert [182] applied this method to filter out irrelevant genes for classifications. They also compared the single decision tree-based classification algorithm C4.5 with ensemble classification algorithms Bagging and AdaBoost, they concluded that ensemble methods often perform better than a single classification algorithm, especially in classifying gene expression data. Similar claims can also be found in [21, 46].

Evolutionary algorithms have also been applied in some classification algorithms for microarray data sets. Jirapech-Umpai [104] implemented an evolutionary algorithm proposed by Deutsch [45] for multi-class classification. The study intended to investigate the problem of searching the optimal parameters for the evolutionary algorithm which will generate the optimal number of predictive genes among the initial gene pool. The performances of the algorithm are measured by testing on the leukemia and the NCI60 data sets. They concluded that good results can be achieved by tuning the parameters within the evolutionary algorithms.

Díaz-Uriarte and Alvarez de Andrés used random forests [32] as the dimensionality reduction algorithm as well as the algorithm for classification of microarray data. Their scheme trains random forests iteratively. At each iteration, the number of input variables is reduced by discarding those variables with the smallest variable

importance. They showed that their new gene selection procedure selects small sets of genes while maintaining high predictive accuracy.

Statistical methods in multivariate analysis such as Partial Least-Squares (PLS) regression [90, 100] and Principal Component Analysis (PCA) have also been adopted for feature selection in microarray data. Nguyen and Rocke [133] conducted a numerical simulation study on the PLS and the PCA methods for microarray-based classification. They concluded that when being applied as the dimensionality reduction method for classification algorithms, PLS out-performs PCA with microarray data.

Although feature selection methods do not explicitly select a subset of genes most relevant to the target concept, attempts have been made to interpret the results of feature selection methods. Roden et al. [167] presented a method for identifying subsets of biologically relevant genes by using a combination of principal component analysis and information-theoretic metrics. The connection between PLS dimensionality reduction and gene selection was examined by Boulesteix [21]. The study found that the order of the absolute values of the coefficients for the first PLS component was identical to the order produced by the classical BSS/WSS ratio gene selection scheme.

5.7 Related Algorithms

5.7.1 Principal Component Analysis Regression and Partial Least Squares Regression

Principal Component Analysis (PCA) reduces the dimension of the original data space by projecting original data points to a new coordinate system of the same dimensionality, and then restricting this. The principle components (PC) are orthogonal and calculated by running the nonlinear iterative partial least squares (NIPALS) algorithm, which in turn maximizes the variance on each coordinate sequentially. So the i^{th} PC is given by

$$w_i = \arg\max_{w^T w=1} var\{w^T x\},$$

subject to $t_i^T t_j = 0$, where $i \neq j, t_k = w_k^T x$. The idea behind PCA is to discover and retain those characteristics which contribute most to its variance. As such, the dimension of the data set can be reduced by keeping the lower order (small i) PCs while omitting the higher order (large i) PCs.

Partial least squares (PLS) regression aims to reduce the data dimensionality with a similar motivation, but differs from PCA by adopting a different objective function to obtain PLS components. Whereas PCA maximises the variance of each coordinate and whereas both PCA and latent factor analysis will not take into account the values of the target (dependent) attribute, the PLS regression model attempts to find a small number of linear combinations of the original independent variables which maximise the covariance between the dependent variable and the PLS components. (PLS uses the entire data set: input and target attributes.) So the i^{th} PLS component is given by

$$w_i = \arg\max_{w^T w=1} cov\{w^T x, y\},$$

subject to $t_i^T t_j = 0$, where $i \neq j, t_k = w_k^T x$.

The PLS method can be illustrated by examining the following relations. Assuming X is an $N \times M$ matrix representing a data set of N instances with p independent variables, then if the number of PLS components is K, then the matrix X can be written as the summation of K matrices generated by outer products between vector t_i (which is often known as the score vector) and p_i^T (which is often called the load vector). The optimal number of PLS components, K, is usually determined by applying cross-validation methods on training data. The details of choosing the optimal K for this study can be found in sec. 5.8.2.

$$X = TP^T + E = \sum_{i=1}^{K} t_i p_i^T + E$$

In effect, the relation in the PLS model projects the data vectors X from the original p-dimensional space into a (much lower than p) K-dimensional space. In the

same way, when PLS components are used in the regression, the relation between dependent variable y and PLS component t_i can be written as

$$Y = TBQ + F$$

where T is PLS components matrix, B is the coefficients vector so that TB is orthogonal, Q is the regression coefficients matrix, F is the residual matrix and $\|F\|$ is to be minimised.

Partial least squares regression can be regarded as an extension of the multiple linear regression model. It has the advantage of being more robust, and therefore it provides a good alternative to the traditional multiple linear regression and principal component methods. The original PLS method was proposed by Wold [228] in the late 1960s and initially applied in the field of econometrics. Since then the method had been adopted in other research disciplines and been widely applied in many scientific analyses. SIMPLS is an algorithm for partial least squares regression proposed by de Jong [44]. Compared to conventional nonlinear iterative partial least squares (NIPALS)-PLS, SIMPLS runs faster and is easier to interpret. In SIMPLS, the PLS components are calculated directly as linear combinations of the original variables, which avoids the construction of deflated data matrices. An implementation of SIMPLS by Mevik [123] as an add-on package for the R statistical environment was used in this study.

5.7.2 MML Oblique Trees and Decision Forests

A MML oblique tree [188] is a multivariate decision tree classification algorithm. At internal nodes of MML oblique trees, the data is divided into two mutually exclusive sets by employing a linear discriminant function of input variables. A MML coding scheme encodes such a split, with the margin between the data and the separating hyperplane taken into account. The motivation behind such a scheme is to find a linear discriminant function with the optimal trade-off between fitting the data and simplicity. Decision forests with MML oblique trees [189] is an ensemble classification algorithm which at least matches and sometimes surpasses the "right"/"wrong"

performance of Breiman's random forests [32]. The optimal candidate trees in decision forests (with overall lower MML coding) with high probabilistic prediction accuracy (low log-loss score) and smaller tree size (lower height with fewer leaf nodes) in MML decision forests are selected by the MML oblique trees algorithm. Compared to schemes with univariate trees (which cut on only one attribute at a time), using MML (multivariate) oblique trees offers potential to greatly increase the diversity of the inferred forest. A new weighted tree averaging scheme is also proposed. The scheme is based on Bayesian weighted tree averaging but uses a modified, smoothed prior on decision trees.

5.7.3 C4.5, AdaBoost and Random Forests

C4.5 [164] is a decision tree inference algorithm introduced by Quinlan. Similar to most decision tree learning algorithms, C4.5 adopts the divide-and-conquer approach to construct decision trees and the procedure is recursive in nature. The C4.5 classification tree algorithm runs fast and it is simple to implement. Therefore, C4.5 trees are often used as base learners in ensemble learning schemes like AdaBoost and random forests.

AdaBoost [89] iteratively re-samples the training set with adapted probabilities (or assigns adapted weights) over instances of the training set. In the end, the scheme gives a weighted average of the results returned by running the classification algorithms on the re-sampling sets. It works very well when the data is noise free and the number of training data is large. But when noise is present in the training sets, or the number of training data is limited, AdaBoost tends not to perform as well as Bagging and random forests.

Random forests [32] uses CART [33] as the base learner and employs several methods to generate a diverse ensemble. Each decision tree in a random forest is trained on a distinct and random data set re-sampled from the original training set, using the same procedure as bagging. While selecting a split at each internal node during the tree growing process, a random set of features is formed by either choosing a subset of input variables or constructing a small group of variables formed by linear combinations of input variables. Random forests [32] have achieved "right"/"wrong" predictive accuracy comparable to that of AdaBoost and much better results on

Table 5.3: Summary of Datasets

Dataset	Number of attributes	Number of PLS components	Number of Instances (Training+Test)	(Binary) Class distribution
Leukaemia	7129	9	72	47:25
Breast cancer	24481	10	97	46:51
Central nervous system	7129	8	60	21:39
Colon tumour	7129	8	62	40:22
Lung cancer	12533	14	181	31:150
Prostate cancer	12600	12	136	77:59
Prostate cancer outcome	12600	5	21	8:13

noisy data sets. Breiman also claimed and showed that AdaBoost is a form of random forest (algorithm) [32].

5.8 Experiments

5.8.1 Data sets

In this study, seven (mainly oncological) microarray data sets are selected - Leukaemia, Breast cancer, Central nervous system (CNS), Colon tumour, Lung cancer, Prostate cancer and Prostate cancer outcome. All seven microarray data sets have binary output attributes and can be freely downloaded from the Gene Expression Datasets Collection. They have properties that are common in microarray data sets and have also been extensively tested in many previous studies, which make comparisons with other approaches more convenient. Table 5.3 shows the summary of the data sets, which can also be found in [182] and on the web site http://sdmc.lit.org.sg/GEDatasets (the web site was accessible in Feb. 2007, but can not be reached now). The datasets now can be located at http://levis.tongji.edu.cn/gzli/data/mirkentridge.html.

5.8.2 Methodology

The dimensionality reduction scheme for this experiment is implemented as follows. Each column of the training set is normalised, so that each column has a mean of zero and variance of one. The values of the binary target attribute are set to either 0 or 1. Specifying the number of components for the Partial Least Square Regression, then a PLS model for a training data set is built by feeding the original training set into the SIMPLS algorithm. The output scores of the PLS algorithm are regarded as the values of input variables and form the training set for the classification algorithms. Similarly, the test sets for the classification algorithm were obtained by feeding each instance of the original test data into the PLS model built from the training set.

Determining the optimal number of PLS components

There is only one free parameter in the PLS algorithms - the number of components, m. There are extensive discussions on how to determine the optimal number of components. However, the goal for performing PLS on the training set in this study is not for regression, rather, the PLS method is applied as a procedure for data pre-processing for the decision tree-based classification algorithms. In this scheme, m is the number of input variables of the data sets to train decision tree and various ensemble learning algorithms.

One major advantage of leave-one-out cross-validation is that it retains the maximum number of data as training sets. As the number of samples in a typical microarray data set is small, leave-one-out cross-validations is used here to find the optimal number of components m which will result in classification models with highest "right"/"wrong" predictive accuracies. For each pair of data set and learning algorithm, the PLS method was repeated with various numbers of PLS components m which ranged from 2 to $4\sqrt{N}$. To reduce the computational cost, the number of PLS components is increased by 2 instead of 1 in each iteration. Then the numbers of PLS components leading to classification models with highest predictive accuracies are regarded as the optimal numbers, as shown in table 5.8.2.

Table 5.4: The optimal number of PLS components

Dataset	Single C4.5	Random Forest	C5.0 AdaBoost	MML Oblique Forest
Leukaemia	2	8	8	2
Breast cancer	14	10	10	10
Central nervous system	4	6	4	8
Colon tumour	2	4	2	4
Lung cancer	2	2	2	4
Prostate cancer	10	24	12	32
Prostate cancer outcome	18	18	6	6

Ten-fold cross-validation

For each original data set, 100 pairs of training and test data sets are generated by repeating the 10-fold cross-validation method ten times. Then these 100 pairs of data sets are pre-processed by using procedures described at the beginning of this section. For each of 100 pairs of training and test sets which resulted from the above process, classification models were built and tested by using the four classification algorithms (C4.5 [164], AdaBoost [88, 89], Random Forests [32] and MML decision forests [189]) described at the end of the introduction.

Leave-one-out cross-validation

By selecting one instance from a data set as a test set and using the rest of the data as a training set, N pairs of training and test sets were obtained for a data set with N instances by selecting each data instance only once as a test set. Then for each of the N pairs of training and test set, the experiments were conducted as the procedures described in subsection 5.8.2 immediately above.

5.8.3 Results and Discussions

Table 5.5 shows the classification performances of the four classification algorithms on seven microarray data sets, with the lowest classification errors for each data set highlighted, MML oblique forest achieves the lowest classification error in 5 out

Table 5.5: Predictive error (%) of classification algorithms, using PLS dimensionality reduction scheme ((i) from section 5.5)

Dataset	Single C4.5	Random Forest	C5.0 AdaBoost	MML Oblique Forest
Leukaemia	5.7	3.8	4.3	**3.3**
Breast cancer	34.8	**28.8**	32.1	30.8
Central nervous system	38.8	35.5	36.8	**34.1**
Colon tumour	19.1	15.3	17.3	**11.2**
Lung cancer	2.0	3.8	1.8	**0.6**
Prostate cancer	17.0	9.4	11.9	**8.7**
Prostate cancer outcome	35.0	**30.7**	48.5	46.8

of 7 data sets while random forest performs best in the other 2 sets. The MML oblique forests, which ensemble oblique trees with optimal probabilistic prediction performance (see e.g., [188, sec. 3.1][189, sec. 4.2]), return excellent predictive accuracy on noisy data such as microarray data. On the other hand, C5 AdaBoost did not perform well on such noisy data sets. In general, all three decision tree-based ensemble classification algorithms achieve higher predictive accuracies than the single model based decision tree algorithm C4.5. For all seven data sets, the best performing ensemble learning algorithms have classification errors 12.7% to 70% (e.g., $\frac{0.6-2.0}{2.0} = -0.7$) lower than those of C4.5. This clearly indicates that ensemble algorithms are better candidates for building classification models for microarray data sets. Such a conclusion can also be found in [182, 46].

When applying the PLS method directly on the whole gene set from the original data, tests here returned improved classification accuracies on three (Leukaemia, Lung Cancer and Prostate Cancer) data sets. However, the other tests returned lower "right"/"wrong" classification accuracies on other four data sets than those reported in Tan and Gilbert's paper [182]. In [182], a subset of the original set of genes was selected by using Fayyad and Irani's discretization method [79]. For each of the three data sets with improved results in the study herein, at least 14% of the original gene set were retained. Whereas, for each of the other four data sets with worse results, less than 5% of the original gene set was selected and used to build classification models. The observation suggests a two-stage dimensionality

Table 5.6: Predictive error (%) of classification algorithms, using a hybrid dimensionality reduction scheme ((ii) from section 5.5)

Dataset	Single C4.5	Random Forest	C5.0 AdaBoost	MML Oblique Forest
Leukaemia	3.3	**1.9**	3.6	**1.9**
Breast cancer	21.9	18.4	20.4	**17.9**
Central nervous system	25.8	**21.7**	27.6	23.1
Colon tumour	12.3	15.7	15.8	**11.6**
Lung cancer	1.8	**0.1**	1.8	0.2
Prostate cancer	10.9	7.1	8.6	**6.6**
Prostate cancer outcome	32	**20.3**	30.3	28.1

reduction scheme. In the first stage, irrelevant genes are filtered out by using Fayyad and Irani's discretization method [79]. In the second stage, the dimension of the data is further reduced by applying the PLS method on the data with reduced numbers of genes. Each of seven data sets was processed using the above scheme, then the experiments were re-run. These experimental results show that, in going from the PLS scheme in table 5.5 to the hybrid scheme in table 5.6, there are significant increases across the board in classification accuracy. In some data sets like the lung cancer data set, the predictive accuracies were extremely high (something like 99.9%).

5.9 Summary

An ensemble classifier using shallow oblique decision trees, the new ensemble learning algorithm presented here, achieves favourable results on data sets with binary classes. The novel random decision tree generating scheme is capable of constructing a decision forest with a large number of distinct high performance decision trees. The proposed weighted averaging scheme exploits the potential of using Bayesian weighted averaging to improve the predictive accuracy of ensemble classifiers, especially on probabilistic predictions and any predictions involving binary output classes. It is reasonable to believe that replacing the preliminary model with well developed and more elaborate models (such as mixtures of Normal distributions or

other distributions) to approximate the posterior probabilities of the inferred trees can only further enhance the results from this kind of weighted averaging.

An extensive survey in the area of building classification models from microarray data with various tree-based classification algorithms was conducted. Experimental results show that in most cases, tree-based ensemble learning algorithms delivered classification accuracies equivalent to or better than those on the same data sets reported by other studies. Combined with the Partial Least-Squares (PLS) regression method, which has proved to be an appropriate feature selection method, tree-based ensemble learning algorithms, especially MML oblique decision forests, are capable of building classification models with high predictive accuracies from microarray data. The study shows that the hybrid feature selection scheme improves classification accuracies.

Chapter 6

Conclusion

This thesis presents a series of statistical inference schemes based on the MML principle. MML inference not only retains merits of general Bayesian inference methods, but also possesses many desirable properties such as model invariance, which the former lacks. MML estimation attempts to derive the most accurate assertion from a set of finite data by converting statistical inference problems into coding processes, which encode a model in the first part of the message and encode the data given the model in the second part of the message. Then a search algorithm is applied to look for the assertion resulting in the shortest two-part message length. However, to encode a group of data in the shortest code length is not the object of devising a MML coding scheme. Rather, construction of an efficient code book should always follow the task itself. In practice, MML and the subsequent Minimum Description Length (MDL) principle [166, 120] (see also [207, sec. 10.2] and [214, sec. 6.7] for comparisons and a survey thereof) are widely used for model selection in various machine learning problems, and both can be thought of as operational forms of Ockham's razor [128]. In the proceeding chapters, MML coding schemes are applied to the problem of inference of decision trees, graphs and forests. The aim of these schemes is to find the optimal trade-off between the complexity of these structure models and goodness-of-fit for a given set of data.

In Chapter 3, a novel oblique decision tree inference scheme is presented. The resulting decision tree inference scheme allows multivariate discriminant functions at internal nodes that are able to partition the instance space with hyperplanes having

arbitrary slopes (rather than only parallel to the co-ordinate axes). Such flexibility alleviates the problem of inefficient representation in univariate trees. By encoding the multivariate splits and extending the MML decision tree coding scheme, the algorithm succeeds in finding the optimal trade-off between the complexity of the model and the goodness of fit.

Data sets from the UCI machine learning repository are used to test the multivariate tree inference scheme. The results are compared to the decision tree programs C4.5 and C5. The results show that on average and on most data-sets, MML oblique trees clearly perform better than both C4.5 and C5 on both "right"/"wrong" accuracy overall and probabilistic prediction, and furthermore, manage to do this with smaller trees, i.e., fewer leaf nodes.

In Chapter 4 a refined coding scheme for decision graphs which allows multi-way joins is introduced. The novel idea is to decompose a decision graph into a sequence of decision trees. By doing this, a MML decision tree coding scheme and inference algorithm can be used in the decision graph system. The replication and the fragmentation problems of decision tree are addressed by introducing more efficient representations of decision graphs. In the second part of the chapter, further refinements of the decision graph with multi-way joins representation are achieved by introducing dynamic attributes in the decision graphs. Thus an improved coding scheme for inferring the new decision graphs is devised to address some of the inefficiencies in previous decision tree and decision graph coding schemes. The experimental results demonstrate that the refined coding scheme compares favourably with other decision tree inference schemes, namely both C4.5 and C5.

Chapter 5 introduces an ensemble classifier using shallow oblique decision trees. The new ensemble learning algorithm achieves favourable results on data sets with binary classes. The proposed novel random decision tree generating scheme is capable of constructing a decision forest with a large number of distinct highly performing decision trees. The proposed weighted averaging scheme exploits the potential of using Bayesian weighted averaging to improve the predictive accuracy of ensemble classifiers, especially on probabilistic predictions and any predictions involving binary output classes. It is reasonable to believe that replacing the preliminary model

with well developed and more elaborate models (such as mixtures of Normal distributions or other distributions) to approximate the posterior probabilities of the inferred trees can only further enhance the results from this kind of weighted averaging. In the second part of the chapter, the MML oblique decision forests are used to classify microarray data sets. Experimental results show that in most cases, tree-based ensemble learning algorithms delivered classification accuracies equivalent to or better than those on the same data sets reported by other studies. Combined with the Partial Least-Squares (PLS) regression method, which is proved to be an appropriate feature selection method, tree-based ensemble learning algorithms, especially MML oblique decision forests, are capable of building classification models with high predictive accuracies from microarray data. The study shows that the hybrid feature selection scheme combining with MML oblique decision forests improves classification accuracies. Recently, deep learning [15, 172] gained popularity in machine learning community. The wide adoptions of deep neural networks are partly due to its excellent accuracy and flexibility, and partly due to recent advances in computational hardware, such as GPGPU, which enable efficient implementations of deep learning algorithms. Seith [174] shows that oblique decision tree model has similarities with deep learning architectures, which can also be exploited for systematic design of layered neural networks.

References

[1] Y. Agusta. *Minimum Message Length Mixture Modelling for Uncorrelated and Correlated Continunous Data Applied to Mutual Funds Classification.* PhD thesis, School of Computer Science and Software Engineering, Monash University, Australia, 2004.

[2] Y. Agusta and D. L. Dowe. Clustering of Gaussian and t distributions using minimum message length. In *Proc. International Conference on Knowledge Based Computer Systems (KBCS 2002)*, pages 289–299. Vikas Publishing House, 2002.

[3] Y. Agusta and D. L. Dowe. MML clustering of continuous-valued data using Gaussian and t distributions. In B. McKay and J. Slaney, editors, *Lecture Notes in Artificial Intelligence (LNAI), Proc. Australian Joint Conference on Artificial Intelligence*, volume 2557, pages 143–154, Berlin, Germany, December 2002. Springer-Verlag.

[4] L. Allison and C. S. Wallace. An information measure for the string to string correction problem with applications. *17th Australian Comp. Sci. Conf.*, pages 659–668, January 1994.

[5] L. Allison and C. S. Wallace. The posterior probability distribution of alignments and its application to parameter estimation of evolutionary trees and to optimization of multiple alignments. *J. Mol. Evol.*, 39(4):418–430, October 1994. Early version: TR 93/188, Dept. Computer Science, Monash University, July 1993.

[6] L. Allison, C. S. Wallace, and C. N. Yee. Induction inference over macro–molecules. Technical Report 90/148, Monash University, Clayton, Victoria, Australia, 3168, 1990.

[7] L. Allison, C. S. Wallace, and C. N. Yee. When is a string like a string? In *International Symposium on Artificial Intelligence and Mathematics*, January 1990.

[8] L. Allison, C. S. Wallace, and C. N. Yee. Finite-state models in the alignment of macro-molecules. *J. Mol. Evol.*, 35(1):77–89, July 1992.

[9] L. Allison, C. S. Wallace, and C. N. Yee. Minimum message length encoding, evolutionary trees and multiple alignment. *25th Hawaii Int. Conf. on Sys. Sci.*, 1:663–674, January 1992. Another version is given in TR 91/155, Dept. Computer Science, Monash University, Clayton, Vic, Australia, 1991.

[10] R. C. Barros, M. P. Basgalupp, A. De Carvalho, and A. A. Freitas. A survey of evolutionary algorithms for decision-tree induction. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42(3):291–312, 2012.

[11] R. Baxter. *Minimum Messeage Length Inference: Theory and Applications*. PhD thesis, Dept. Computer Science, Monash University, Australia, 1996.

[12] 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.

[13] R. A. Baxter and D. L. Dowe. Model selection in linear regression using the MML criterion. Technical Report 96/276, 13pp, Dept. Computer Science, Monash University, Australia 3168, September 1996.

[14] R. A. Baxter, D. L. Dowe, and J. J. Oliver. Bayesian point estimation via the Kullback-Leibler distance. In *Proc. Kullback Memorial Conference*, Washington, D.C., U.S.A., May 1996.

[15] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. Also published as a book. Now Publishers, 2009.

[16] K. P. Bennett and J. A. Blue. A support vector machine approach to decision trees. In *IJCNN*, pages 2396–2401, Anchorage, Alaska, 1998.

[17] A. C. Bickerstaffe and D. L. Dowe. MML-based Compressive Models for Musical Melody. In *Proc. 4th International ICSC Symposium on Engineering of Intelligent Systems (EIS 2004)*, Island of Madiera, Portugal, 2004. ICSC Academic Press, Canada, ISBN: 3-906454-35-5, 6pp.

[18] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/∼MLearn/MLRepository.html.

[19] P. Bonissone, J. M. Cadenas, M. C. Garrido, and R. A. Díaz-Valladares. A fuzzy random forest. *International Journal of Approximate Reasoning*, 51(7):729–747, 2010.

[20] M. Bot and W. B. Langdon. Application of genetic programming to induction of linear classification trees. In *Genetic Programming*, pages 247–258. Springer, 2000.

[21] A. L. Boulesteix. PLS dimension reduction for classification with microarray data. *Statistical Applications in Genetics and Molecular Biology*, 3(1), Nov. 2004.

[22] D. M. Boulton. *The Information Criterion for Intrinsic Classification.* PhD thesis, Dept. Computer Science, Monash University, Australia, 1975.

[23] D. M. Boulton and C. S. Wallace. The information content of a multistate distribution. *J. Theor. Biol.*, 23:269–278, 1969.

[24] D. M. Boulton and C. S. Wallace. A program for numerical classification. *Computer Jrnl.*, 13(1):63–69, February 1970.

[25] D. M. Boulton and C. S. Wallace. A comparison between information measure classification. In *Proc. of the Australian & New Zealand Association for the Advancement of Science (ANZAAS) Congress*, August 1973.

[26] D. M. Boulton and C. S. Wallace. An information measure for hierarchic classification. *Computer J.*, 16(3):254–261, 1973.

[27] D. M. Boulton and C. S. Wallace. An information measure for single link classification. *The Computer Journal*, 18(3):236–238, 1975.

[28] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[29] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[30] L. Breiman. Arcing classifiers. *The Annals of Statistics*, 26(3):801–824, Jun. 1998.

[31] L. Breiman. Randomizing outputs to increase prediction accuracy. *Machine Learing*, 40:229–242, 2000.

[32] L. Breiman. Random forests. *Machine Learning*, 45(1):5, 2001.

[33] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification And Regression Trees*. Wadsworth & Brooks, 1984.

[34] E. Cantu-Paz and C. Kamath. Using evolutionary algorithms to induce oblique decision trees. In *Proc.Genetic and Evolutionary Computation Conference*, pages 1053–1060, Las Vegas, Nevada, USA, 2000. Morgan Kaufmann.

[35] D. M. Clarke, G. C. Smith, D. L. Dowe, and D. P. McKenzie. An empirically-derived taxonomy of common distress syndromes in the medically ill. *J. Psychosomatic Research*, 54:323–330, 2003.

[36] M. J. Collie, D. L. Dowe, and L. J. Fitzgibbon. Stock market simulation and inference technique. In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*, Rio de Janeiro, Brazil, Nov 2005.

[37] M. J. Collie, D. L. Dowe, and L. J. Fitzgibbon. Trading rule search with autoregressive inference agents. Technical report CS 2005/174, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia, 2005.

[38] J. W. Comley. *Machine Learning With Generalised Tree-Based Bayesian Networks, Decision Trees And MML.* PhD thesis, School of Computer Science and Software Engineering, Monash University, Australia, 2005.

[39] J. W. Comley and D. L. Dowe. Generalised Bayesian networks and asymmetric languages. In *Proc. Hawaii International Conference on Statistics and Related Fields*, 5-8 June 2003.

[40] J. W. Comley and D. L. Dowe. Chapter 11, Minimum Message Length and generalized Bayesian networks with asymmetric languages. In P. Grünwald, M. A. Pitt, and I. J. Myung, editors, *Advances in Minimum Description Length: Theory and Applications*, pages 265–294. M.I.T. Press, Apr 2005. Final camera-ready copy submitted Oct. 2003.

[41] Honghua Dai, K. B. Korb, C. S. Wallace, and Xindong Wu. A study of causal discovery with weak links and small samples. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97*, pages 1304–1309, 1997.

[42] M. Dale, P. Dale, and P. J. Tan. Supervised clustering using decision trees and decision graphs: An ecological comparison. *ecological modelling*, 204(1):70–78, 2007.

[43] G. B. Dantzig. *Linear programming and extensions.* Princeton university press, 1998.

[44] S. de Jong. SIMPLS: an alternative approach to partial least squares regression. *Chemometrics and Intelligent Laboratory Systems*, 2(4):251–263, 1993.

[45] J. M. Deutsch. Evolutionary algorithms for finding optimal gene sets in microarray prediction. *Bioinformatics*, 19:45–52, 2003.

[46] R. Díaz-Uriarte and S. Alvarez de Andrés. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7:3, 2006.

[47] T. G. Dietterich. Machine-learning research: Four current directions. *The AI Magazine*, 18(4):97–136, 1998.

[48] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.

[49] D. L. Dowe. Foreword re C. S. Wallace. *Computer Journal*, 51(5):523–560, 2008. Christopher Stewart WALLACE (1933-2004) memorial special issue.

[50] D. L. Dowe. Minimum message length and statistically consistent invariant (objective?) Bayesian probabilistic inference – from (medical) "evidence". *Social Epistemology*, 22(4):433–460, 2008.

[51] D. L. Dowe. MML, hybrid Bayesian network graphical models, statistical consistency, invariance and uniqueness. *Handbook of the Philosophy of Science, Vol.7 Philosophy of Statistics*, pages 901–982, 2011.

[52] D. L. Dowe. Introduction to Ray Solomonoff 85th Memorial Conference. In D. L. Dowe, editor, *Algorithmic Probability and Friends. Bayesian Prediction and Artificial Intelligence*, volume 7070 of *Lecture Notes in Computer Science*, pages 1–36. Springer Berlin Heidelberg, 2013.

[53] D. L. Dowe, L. Allison, T. I. Dix, L. Hunter, C. S. Wallace, and T. Edgoose. Circular clustering of protein dihedral angles by minimum message length. In *Pacific Symposium on Biocomputing '96*, pages 242–255. World Scientific, January 1996.

[54] D. L. Dowe, L. Allison, and G. Pringle. The hunter and the hunted: Modelling the relationship between web pages and search engines. In Xindong Wu, R. Kotagiri, and K. B. Korb, editors, *Proceedings of the 2nd Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data*

Mining (PAKDD-98), volume 1394 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 380–382, Berlin, April 15–17 1998. Springer.

[55] D. L. Dowe, R. A. Baxter, J. J. Oliver, and C. S. Wallace. Point estimation using the Kullback-Leibler loss function and MML. In Xindong Wu, R. Kotagiri, and K. B. Korb, editors, *Proceedings of the 2nd Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining (PAKDD-98)*, volume 1394 of *LNAI*, pages 87–95, Berlin, April 15–17 1998. Springer.

[56] D. L. Dowe, M. Doran, G. E. Farr, A. J. Hurst, D. R. Powell, and T. Seemann. Kullback-Leibler distance, probability and football prediction. In W. Robb, editor, *Proceedings of the Fourteenth Biennial Australian Statistical Conference (ASC-14)*, page 80, July 1998.

[57] D. L. Dowe, G. E. Farr, A. J. Hurst, and K. L. Lentin. Information-theoretic football tipping. pages 233–241. Bond University, Qld, Australia, 1996. See also Technical Report TR 96/297, Dept. Computer Science, Monash University, Australia 3168, Dec 1996.

[58] D. L. Dowe, S. Gardner, and G. R. Oppy. Bayes not Bust! Why simplicity is no problem for Bayesians. *British Journal for the Philosophy of Science*, 58(4):709–754, 2007.

[59] D. L. Dowe and A. R. Hajek. A computational extension to the Turing test. In *Proceedings of the 4th Conference of the Australasian Cognitive Science Society*, Newcastle, NSW, Australia, Sep. 1997. See also technical report 97/322, Dept. Computer Science, Monash University, Australia, Oct, 1997.

[60] D. L. Dowe and J. Hernández-Orallo. How universal can an intelligence test be? *Adaptive Behaviour*, 22(1):51–69, 2014.

[61] D. L. Dowe and K. B. Korb. Conceptual difficulties with the efficient market hypothesis: Towards a naturalized economics. In *Proc. Information, Statistics and Induction in Science (ISIS)*, pages 212–223, 1996. See also technical

Report TR 94/215, Dept. Computer Science, Monash University, Australia 3168, 1994.

[62] D. L. Dowe and N. Krusel. Decision tree models of bushfire activity. *AI Applications*, 8(3):71–72, 1994. See also technical report 93/190 Dept. of Computer Science, Monash University, Clayton, Vic. 3800, Australia.

[63] D. L. Dowe, K. L. Lentin, J. J. Oliver, and A. J. Hurst. An information-theoretic and a Gaussian footy-tipping competition. *FCIT Faculty Newsletter, Monash University, Australia*, pages 2–6, June 1996.

[64] D. L. Dowe, J. J. Oliver, L. Allison, T. I. Dix, and C. S. Wallace. Learning rules for protein secondary structure prediction. In C. McDonald, J. Rohl, and R. Owens, editors, *Proc. 1992 Department Research Conference*. Dept. Computer Science, University of Western Australia, July 1992.

[65] D. L. Dowe, J. J. Oliver, R. A. Baxter, and C. S. Wallace. Bayesian estimation of the von Mises concentration parameter. In *Proc. 15th Int. Workshop on Maximum Entropy and Bayesian Methods, Santa Fe*, July 1995. See also Tech Report CS 95/236, Dept. of Computer Science, Monash University, Clayton, Australia.

[66] D. L. Dowe, J. J. Oliver, T. I. Dix, L. Allison, and C. S. Wallace. A decision graph explanation of protein secondary structure prediction. *26th Hawaii Int. Conf. Sys. Sci.*, 1:669–678, January 1993.

[67] D. L. Dowe, J. J. Oliver, and C. S. Wallace. MML estimation of the parameters of the spherical Fisher distribution. In *Algorithmic Learning Theory, 7th International Workshop, ALT '96, Sydney, Australia, October 1996, Proceedings*, volume 1160 of *Lecture Notes in Artificial Intelligence*, pages 213–227. Springer, 1996.

[68] D. L. Dowe and K. Prank. Information theoretic approaches to biology - session introduction. In *Pacific Symposium on Biocomputing*, pages 252–253, 1999.

[69] D. L. Dowe and C. S. Wallace. Resolving the Neyman-Scott problem by minimum message length (abstract). In *Proc. Sydney Int. Stat. Congress*, pages 197–198, 1996.

[70] D. L. Dowe and C. S. Wallace. Kolmogorov complexity, minimum message length and inverse learning. In *14th Australian Statistical Conference (ASC-14)*, page 144, Gold Coast, Qld, Australia, 6-10 July 1998.

[71] J. G. Dowty. Smml estimators for 1-dimensional continuous data. *The Computer Journal*, 58(1):126–133, 2015.

[72] T. Edgoose, L. Allison, and D. L. Dowe. An MML classification of protein structure that knows about angles and sequence. In *Pacific Symposium on Biocomputing '98*, pages 585–596. World Scientic, January 1998.

[73] R. T. Edwards and D. L. Dowe. Single factor analysis in MML mixture modelling. In Xindong Wu, R. Kotagiri, and K. B. Korb, editors, *Proceedings of the 2nd Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining (PAKDD-98)*, volume 1394 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 96–109, Berlin, April 15–17 1998. Springer.

[74] J. P. Egan. Signal detection theory and ROC analysis. *Series in Cognition and Perception, Academic Press*, 1975.

[75] G. E. Farr and C. S. Wallace. Algorithmic and combinatorial problems in strict minimum message length inference. *Res. in Combinatorial Algorithms, Queensland University of Technology, Brisbane, Australia*, pages 50–58, 1997.

[76] G. E. Farr and C. S. Wallace. The complexity of strict minimum message length inference. *Computer J.*, 45(3):285–292, 2002.

[77] T. Fawcett. ROC graphs: Notes and practical considerations for researchers. Technical report, HP Laboratories, Palo, Alto, California, USA, 2004.

[78] T. Fawcett. An introduction to ROC analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.

[79] U. M. Fayyad and K. B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *IJCAI*, pages 1022–1029, 1993.

[80] C. Ferri, P. Flach, and J. Hernandez-Orallo. Delegating classifiers. In *Proc. 21st International Conference on Machine Learning*, pages 106–110, Banff, Canada, 2004.

[81] C. Ferri, P. A. Flach, and J. Hernández-Orallo. Learning decision trees using the area under the ROC curve. In *ICML*, pages 139–146, 2002.

[82] C. Ferri, P. A. Flach, and J. Hernández-Orallo. Volume under the ROC surface for multi-class problems. *Machine Learning: ECML 2003*, pages 108–120, 2003.

[83] L. J. Fitzgibbon. *Message From Monte Carlo: A Framework for Minimum Message Length Inference using Markov Chain Monte Carlo Methods.* PhD thesis, School of Computer Science and Software Engineering, Monash University, Australia, 2005.

[84] L. J. Fitzgibbon, L. Allison, and D. 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, 2000.

[85] L. J. Fitzgibbon, D. L. Dowe, and L. Allison. Change-point estimation using new minimum message length approximations. In *Lecture Notes in Artificial Intelligence (LNAI) 2417 (Springer), the Pacific Rim International Conferences on Artificial Intelligence (PRICAI)*, pages 244–254, 2002.

[86] L. J. Fitzgibbon, D. L. Dowe, and L. Allison. Univariate polynomial inference by Monte Carlo message length approximation. In *19th International Conference on Machine Learning (ICML)*, pages 147–154, 2002.

[87] L. J. Fitzgibbon, D. L. Dowe, and L. Allison. Bayesian posterior comprehension via message from Monte Carlo. *2nd Hawaii Int. Conf. on Statistics and Related Fields (HICS)*, June 2003.

[88] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.

[89] Y. Freund and R. E. Schapire. Experiments with a new boosting algorithm. In *International Conference on Machine Learning (ICML)*, pages 148–156, 1996.

[90] P. Geladi and B. Kowalski. Partial least-squares regression: a tutorial. *Analytical Chimica Acta*, 185:1–17, 1986.

[91] M. P. Georgeff and C. S. Wallace. A general selection criterion for inductive inference. In *Sixth European Conf. on Artificial Intelligence (ECAI-84): Advances in Artificial Intelligence*, pages 473–482. Elsevier Science, 1984.

[92] M. P. Georgeff and C. S. Wallace. Minimum information estimation of structure. In T. O'Shea, editor, *Advances in Artificial Intelligence*, pages 219–228. Elsevier, 1985.

[93] I. A. Gheyas and L. S. Smith. A novel neural network ensemble architecture for time series forecasting. *Neurocomputing*, 74(18):3855–3864, 2011.

[94] I. J. Good. Rational Decisions. *Journal of the Royal Statistical Society. Series B*, 14:107–114, 1952.

[95] I. J. Good. Corroboration, Explanation, Evolving Probability, Simplicity and a Sharpened Razor. *British Journal of Philosophy of Science*, 19:123–143, 1968.

[96] D. J. Hand and R. J. Till. A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, 45(2):171–186, Nov 2001.

[97] J. A. Hanley and B. J. McNeil. The meaning and use of the area under a receiver operating characteristic (ROC) curve. *Radiology*, 143(1):29–36, Apr. 1982.

[98] D. G. Heath, S. Kasif, and S. Salzberg. Induction of oblique decision trees. In *International Joint Conference on AI (IJCAI)*, pages 1002–1007, 1993.

[99] Tin Kam Ho. The random subspace method for constructing decision forests. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):832–844, August 1998.

[100] A. Höskuldsson. PLS regression methods. *Journal of Chemometrics*, 2(3):211–228, 1988.

[101] D. Hosmer and S. Lemeshow. Introduction to the logistic regression model. *Applied Logistic Regression, Second Edition*, pages 1–30, 2000.

[102] J. H. Conway and N. J. A. Sloane. Voronoi regions of certain lattices. *SIAM J. Algebraic Discrete Methods*, 5(3):294–305, 1984.

[103] J. H. Conway and N. J. A. Sloane. Low-dimensional lattices I: Quadratic forms of small determinant. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 418(1854):17–41, Jul 1988.

[104] Thanyaluk Jirapech-umpai. *Classifying Gene Data Expression using an Evolutionary Algorithm*. Master thesis, University of Edinburgh, 2004.

[105] D. W. Kissane, S. Bloch, W. I. Burns, J. D. Patrick, C. S. Wallace, and D. P. McKenzie. Perceptions of family functioning and cancer. *Psycho-oncology*, 3:259–269, 1994.

[106] D. W. Kissane, S. Bloch, P. Onghena, D. P. McKenzie, R. D. Snyder, and D. L. Dowe. The Melbourne family grief study, II: Psychosocial morbidity and grief in bereaved families. *American Journal of Psychiatry*, 153:659–666, 1996.

[107] R. Kohavi. Bottom-up induction of oblivious read-once decision graphs: Strengths and limitations. In *National Conference on Artificial Intelligence*, pages 613–618, 1994.

[108] A. N. Kolmogorov. Three approaches to the quantitive definition of information. *Problems in Information Transmission*, 1:1–7, 1965.

[109] K. B. Korb and C. S. Wallace. In search of the philosopher's stone: Remarks on Humphreys and Freedman's critique of causal discovery. *British Jrnl. for the Philosophy of Science*, pages 543–553, 1999. TR 97/315, Mar 1997, Dept. Computer Science, Monash University, Australia 3168.

[110] L. Kornienko, D. W. Albrecht, and D. L. Dowe. A preliminary MML linear classifier using principal components for multiple classes. In *Proc. 18th Australian Joint Conference on Artificial Intelligence (AI'2005)*, page 176, Sydney, Australia, Dec 2005. See also Technical report CS 2005/179, School of Computer Science and Software Engineering, Monash University, Melbourne, Australia, 2005.

[111] L. Kornienko, D. L. Dowe, and D. W. Albrecht. Message length formulation of support vector machines for binary classification - A preliminary scheme. In *Lecture Notes in Artificial Intelligence (LNAI) 2557 (Springer), Proc. 15th Australian Joint Conf. on AI*, pages 119–130, 2002.

[112] N. Krusel and D. L. Dowe. Predicting bushfire activity in the Mallee region of north west Victoria using a decision tree model. In *Proc. Inst. Australian Geographers (IAG) Conf.*, Sep 1993.

[113] Suk Wah Kwok and C. Carter. Multiple decision trees. *CoRR*, abs/1304.2363, 2013.

[114] N. Lachiche and P. A. Flach. Improving accuracy and cost of two-class and multi-class probabilistic classifiers using ROC curves. In *ICML*, pages 416–423, 2003.

[115] L. Lymburner, P. Tan, N. Mueller, R. Thackway, A. Lewis, M. Thankappan, L. Randall, A. Islam, and V. Senarath. 250 dynamic land cover dataset version 1. http://www.ga.gov.au/scientific-topics/earth-obs/landcover/executive-summary, 2010. Accessed: 2016-08-11.

[116] M. T. Maheswaran, J. G. Sanjayan, D. L. Dowe, and P. J. Tan. MML mixture models of heterogeneous Poisson processes with uniform outliers for bridge

deterioration. In *Lecture Notes in Artificial Intelligence (LNAI) (Springer), Proc. 19th Australian Joint Conf. on AI*, pages 322–331, Hobart, Australia, Dec. 2006.

[117] E. Makalic, L. Allison, and D. L. Dowe. MML inference of single-layer neural networks. In *Proc. of the 3rd IASTED Int. Conf. on Artificial Intelligence and Applications,*, pages 636–642, September 2003. See also TR 2003/142, CSSE, Monash University, Australia Oct. 2003.

[118] Y. Mansour and D. McAllester. Boosting using branching programs. In *Proc. 13th Annual Conference on Comput. Learning Theory (CoLT)*, pages 220–224. Morgan Kaufmann, San Francisco, 2000.

[119] D. P. McKenzie, P. D. McGorry, C. S. Wallace, L. H. Low, D. L. Copolov, and B. S. Singh. Constructing a minimal diagnostic decision tree. *Methods in Information in Medicine*, 32:161–166, 1993.

[120] M. Mehta, J. Rissanen, and R. Agrawal. MDL-based Decision Tree Pruning. In *The First International Conference on Knowledge Discovery & Data Mining*, pages 216–221. AAAI Press, 1995.

[121] P. Melville and R. J. Mooney. Creating diversity in ensembles using artificial data. *Journal of Information Fusion (Special Issue on Diversity in Multiple Classifier Systems)*, 6(1):99–111, 2004.

[122] B. Menze, B. Kelm, D. Splitthoff, U. Koethe, and F. Hamprecht. On oblique random forests. In *Machine Learning and Knowledge Discovery in Databases*, pages 453–469. Springer, 2011.

[123] Björn-Helge Mevik and R. Wehrens. The PLS package: principal component and partial least squares regression in R. *Journal of Statistical Software*, 18(2):1–24, 2007.

[124] S. B. Molloy, D. W. Albrecht, D. L. Dowe, and K. M. Ting. Model-Based Clustering of Sequential Data. In *Proceedings of the 5th Annual Hawaii International Conference on Statistics, Mathematics and Related Fields*, January 2006.

[125] N. Mueller, A. Lewis, D. Roberts, S. Ring, R. Melrose, J. Sixsmith, L. Lymburner, A. McIntyre, P. Tan, S. Curnow, and A. Ip. Water observations from space: Mapping surface water from 25 years of landsat imagery across australia. *Remote Sensing of Environment*, 174:341 – 352, 2016.

[126] S. K. Murthy. *On Growing Better Decision Trees from Data*. PhD thesis, The John Hopkins University, 1997.

[127] S. K. Murthy. Automatic construction of decision trees from data: A multidisciplinary survey. *Data Min. Knowl. Discov*, 2(4):345–389, 1998.

[128] S. L. Needham and D. L. Dowe. Message length as an effective Ockham's razor in decision tree induction. In *Proc. 8th International Workshop on Artificial Intelligence and Statistics*, pages 253–260, Key West, Florida, U.S.A., Jan. 2001.

[129] J. R. Neil. *MML discovery of Causal Models*. PhD thesis, Monash University, Clayton 3800, Australia, Computer Science and Software Engineering, 2001.

[130] J. R. Neil, C. S. Wallace, and K. B. Korb. Bayesian networks with non-interacting causes. Technical Report 1999/28, School of Computer Science & Software Engineering, Monash University, Australia 3168, September 1999.

[131] J. R. Neil, C. S. Wallace, and K. B. Korb. Learning Bayesian networks with restricted causal interactions. In K. B. Laskey and H. Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 486–493, S.F., Cal., U.S.A., July 30–August 1 1999. Morgan Kaufmann Publishers.

[132] J. Neyman and E. L. Scott. Consistent estimates based on partially consistent observations. *Econometrika*, 16:1–32, 1948.

[133] D. V. Nguyen, D. M. David, and M. Rocke. On partial least squares dimension reduction for microarray-based classification: a simulation study. *Computational Statistics & Data Analysis*, 46(3):407–425, June 2004.

[134] T. D. Nielsen and F. V. Jensen. *Bayesian networks and decision graphs.* Springer Science & Business Media, 2009.

[135] J. Nijkamp, M. van den Broek, E. Datema, S. de Kok, L. Bosman, M. Luttik, P. Daran-Lapujade, W. Vongsangnak, J. Nielsen, W. Heijne, P. Klaassen, C. Paddon, D. Platt, P. Kotter, R. van Ham, M. Reinders, J. Pronk, D. de Ridder, and Jean-Marc Daran. De novo sequencing, assembly and analysis of the genome of the laboratory strain saccharomyces cerevisiae cen.pk113-7d, a model for modern industrial biotechnology. *Microbial Cell Factories*, 11(1):36, 2012.

[136] S. Nowozin. Improved information gain estimates for decision tree induction. *arXiv preprint arXiv:1206.4620*, 2012.

[137] A. L. Oliveira and A. L. Sangiovanni-Vincentelli. Using the minimum description length principle to infer reduced ordered decision graphs. *Machine Learning*, 25(1):23–50, 1996.

[138] J. J. Oliver. Decision Graphs - An Extension of Decision Trees. In *Proc. 4th International Workshop on Artif. Intelligence and Statistics*, pages 343–350, 1993.

[139] J. J. Oliver, R. A. Baxter, and C. S. Wallace. Unsupervised learning using MML. In *Proc. 13th International Conference on Machine Learning*, pages 364–372. Morgan Kaufmann, 1996.

[140] J. J. Oliver, R. A. Baxter, and C. S. Wallace. Minimum message length segmentation. In Xindong Wu, R. Kotagiri, and K. B. Korb, editors, *Proceedings of the 2nd Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining (PAKDD-98)*, volume 1394 of *LNAI*, pages 222–233, Berlin, April 15–17 1998. Springer.

[141] J. J. Oliver and D. L. Dowe. On pruning and averaging decision trees. In *Proc. 12th Int. Conf. Machine Learning*, pages 430–437. Morgan Kaufmann, 1995.

[142] J. J. Oliver and D. L. Dowe. Using unsupervised learning to assist supervised learning. In *Proc. 8th Australian Joint Conf. on AI*, pages 275–282, November 1995. See also TR 95/235, Dept. Comp. Sci., Monash University, Australia 3168, Sep 1995.

[143] J. J. Oliver and D. L. Dowe. Averaging over decision trees. *J. Classification*, 13, 1996.

[144] J. J. Oliver and D. L. Dowe. Minimum message length mixture modelling of Spherical von Mises-Fisher distributions. In *Proc. Sydney International Statistical Congress (SISC-96)*, page 198, Sydney, Australia, Jul 1996. Also Institute of Mathematical Sciences (IMS) Bulletin, Vol. 25 (No. 4), July/August 1996, pp410-411.

[145] J. J. Oliver, D. L. Dowe, and C. S. Wallace. Inferring Decision Graphs Using the Minimum Message Length Principle. In *Proceedings of the 5th Joint Conference on Artificial Intelligence*, pages 361–367. World Scientific, Singapore, 1992.

[146] J. J. Oliver and D. J. Hand. On pruning and averaging decision trees. In A. Prieditis and S. Russell, editors, *Machine Learning: Proceedings of the Twelfth International Conference*, pages 430–437. Morgan Kaufmann, 1995.

[147] J. J. Oliver and C. S. Wallace. Inferring decision graphs. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), workshop 8*, January 1991.

[148] J. N. Ooi and D. L. Dowe. Inferring phylogenetic graphs of natural languages using minimum message length. In *CAEPIA 2005 (11th Conference of the Spanish Association for Artificial Intelligence)*, volume 1, pages 143–152, Nov. 2005.

[149] G. R. Oppy and D. L. Dowe. The Turing test. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy (Summer 2003 Edition)*, 2003.

[150] J. R. Otukei and T. Blaschke. Land cover change assessment using decision trees, support vector machines and maximum likelihood classification algorithms. *International Journal of Applied Earth Observation and Geoinformation*, 12, Supplement 1(0):S27 – S31, 2010.

[151] E. Papp, D. L. Dowe, and S. J. D. Cox. Spectral classification of radiometric data using an information theory approach. In *Proc. Advanced Remote Sensing Conf.*, pages 223–232, UNSW, Sydney, Australia, July 1993.

[152] J. Patrick and C. S. Wallace. Stone circle geometries: an information theory approach. In D. Heggie, editor, *Archaeoastronomy in the New World*. CUP, 1982.

[153] C. Perlich, F. Provost, and J. S. Simonoff. Tree induction versus logistic regression: a learning-curve analysis. *Journal of Machine Learning Research*, 4:211–255, 2003.

[154] D. Platt. When consumers get their genomes. *Personalized Medicine*, 6(6):669–679, 2009.

[155] D. R. Powell, L. Allison, T. I. Dix, and D. L. Dowe. Alignment of low information sequences. In *Australian Computer Science Theory Symposium, (CATS '98)*, pages 215–230. Springer Verlag, February 1998.

[156] D. R. Powell, D. L. Dowe, L. Allison, and T. I. Dix. Discovering simple DNA sequences by compression. In *Pacific Symposium on Biocomputing '98*, pages 597–608. World Scientific, January 1998.

[157] G. Pringle, L. Allison, and D. L. Dowe. What is a tall poppy among web pages? *Computer Networks*, 30(1-7):369–377, 1998.

[158] M. Prior, R. Eisenmajer, S. Leekam, L. Wing, J. Gould, B. Ong, and D. L. Dowe. Are there subgroups within the autistic spectrum? A cluster analysis of a group of children with autistic spectrum disorders. *J. Child Psychol. Psychiat.*, 39(6):893–902, 1998.

[159] F. Provost and P. Domingos. Tree induction for probability-based ranking. *Machine Learning*, 52:199–215, Sept. 2003.

[160] F. J. Provost, T. Fawcett, and R. Kohavi. The case against accuracy estimation for comparing induction algorithms. In *ICML*, pages 445–453, 1998.

[161] J. R. Quinlan. Discovering rules from large collections of examples: a case study. In D. Michie, editor, *Expert Systems in the Microelectronic Age*, Edinburgh, Scotland, 1979. Edinburgh University Press.

[162] J. R. Quinlan. Learning efficient classification procedures and their application to chess end games. In R. Michalski, J. Carbonnel, and T. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*, Palo Alto, CA, 1983. Tioga.

[163] J. R. Quinlan. Simplifying decision trees. *International Journal of Man-Machine Studies*, 27:221–234, 1987.

[164] J. R. Quinlan. *C4.5 : Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, U.S.A., 1992. The latest version of C5 is available from http://www.rulequest.com.

[165] J. R. Quinlan and R. Rivest. Inferring Decision Trees Using the Minimum Description Length Principle. *Information and Computation*, 80:227–248, 1989.

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

[167] J. C. Roden, B. W. King, D. Trout, A. Mortazavi, B. J. Wold, and C. E. Hart. Mining gene expression data by interpreting principal components. *BMC Bioinformatics*, 7:194, 2006.

[168] L. Rokach. Ensemble-based classifiers. *Artificial Intelligence Review*, 33(1-2):1–39, 2010.

[169] B. E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3-4):373–384, 1996.

[170] G. W. Rumantir and C. S. Wallace. Sampling of highly correlated data for polynomial regression and model discovery. In *The 5th International Symposium on Intelligent Data Analysis (IDA)*, pages 370–377, 2001.

[171] R. Schack, G. M. D. Ariano, and C. M. Caves. Hypersensitivity to perturbation in the quantum kicked top. *Physical Review E.*, 50:972–987, 1994.

[172] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015.

[173] B. Schölkopf and A. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, December 2002.

[174] I. Sethi. Entropy nets: from decision trees to neural networks. *Proceedings of the IEEE*, 78(10):1605–1613, Oct 1990.

[175] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423 and 623–656, July and October, 1948.

[176] C. E. Shannon and W. W. Weaver. The mathematical theory of communication, 1949.

[177] J. Snyman. *Practical mathematical optimization: an introduction to basic optimization theory and classical and new gradient-based algorithms*, volume 97. Springer Science & Business Media, 2005.

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

[179] A. Statnikov, L. Wang, and C. F. Aliferis. A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification. *BMC bioinformatics*, 9(1):319, 2008.

[180] C. Strobl, J. Malley, and G. Tutz. An introduction to recursive partitioning: rationale, application, and characteristics of classification and regression trees, bagging, and random forests. *Psychological methods*, 14(4):323, 2009.

[181] J. A. Swets. Measuring the accuracy of diagnostic systems. *Science*, 240(4857):1285–1293, 1988.

[182] Aik Choon Tan and D. Gilbert. Ensemble machine learning on gene expression data for cancer classification. *Applied Bioinformatics*, 2:S75–S83, 2003.

[183] P. Tan, L. Lymburner, and N. Mueller. A novel image based end-member extraction technique to map green, non-photosynthetic and bare soil fractions using landsat data. In *MODSIM2013, 20th International Congress on Modelling and Simulation*, pages 401–407. Modelling and Simulation Society of Australia and New Zealand, 2013.

[184] P. Tan, S. Sagar, N. Mueller, L. Lymburner, M. Thankappan, and A. Lewis. A surface cover change detection method based on the australian geoscience data cube. In McPhee M.J. Weber, T. and R.S Anderssen, editors, *MODSIM2015, 21st International Congress on Modelling and Simulation.*, pages 173–179. Modelling and Simulation Society of Australia and New Zealand, 2015.

[185] P. Tan, L. Lymburnerand M. Thankappan, and A. Lewis. Mapping cropping practices using modis time series: Harnessing the data explosion. *Journal of the Indian Society of Remote Sensing*, 39(3):365–372, 2011.

[186] P. J. Tan and D. L. Dowe. MML inference of decision graphs with multi-way joins. In *Lecture Notes in Artificial Intelligence (LNAI) 2557 (Springer), Proc. 15th Australian Joint Conf. on AI*, pages 131–142, Canberra, Australia, 2-6 Dec. 2002.

[187] P. J. Tan and D. L. Dowe. MML inference of decision graphs with multi-way joins and dynamic attributes. In *Lecture Notes in Artificial Intelligence (LNAI) 2903 (Springer), Proc. 16th Australian Joint Conf. on AI*, pages 269–281, Perth, Australia, Dec. 2003.

[188] P. J. Tan and D. L. Dowe. MML inference of oblique decision trees. In *Lecture Notes in Artificial Intelligence (LNAI) 3339 (Springer), Proc. 17th Australian Joint Conf. on AI*, pages 1082–1088, Cairns, Australia, Dec. 2004.

[189] P. J. Tan and D. L. Dowe. Decision forests with oblique decision trees. In *Lecture Notes in Artificial Intelligence (LNAI) 4293 (Springer), Proc. 5th Mexican International Conf. on Artificial Intelligence*, pages 593–603, Apizaco, Mexico, Nov. 2006.

[190] P. J. Tan, D. L. Dowe, and T. I. Dix. Building classification models from microarray data with tree-based classification algorithms. In *AI 2007: Advances in Artificial Intelligence, 20th Australian Joint Conference on Artificial Intelligence, Gold Coast, Australia, December 2-6, 2007, Proceedings*, pages 589–598, 2007.

[191] A. Torsello and D. L. Dowe. Learning a generative model for structural representations. In Wayne Wobcke and Mengjie Zhang, editors, *AI 2008: Advances in Artificial Intelligence*, volume 5360 of *Lecture Notes in Computer Science*, pages 573–583. Springer Berlin Heidelberg, 2008.

[192] A. Torsello and D. L. Dowe. Supervised learning of a generative model for edge-weighted graphs. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, Dec 2008.

[193] C. R. Twardy, S. Gardner, and D. L. Dowe. Empirical data sets are algorithmically compressible: Reply to McAllister. *Studies in the History and Philosophy of Science (Part A)*, 36(2):391–402, Jun 2005.

[194] W. T. B. Uther and M. M. Veloso. The Lumberjack Algorithm for Learning Linked Decision Tree. In *Proc. 6th Pacific Rim International Conf. on Artificial Intelligence (PRICAI'2000), LNAI 1886 (Springer)*, pages 156–166, 2000.

[195] V. N. Vapnik. *Statistical Learning Theory*, chapter 6.2, pages 224–229. Johnson Wiley & Sons, Inc., 1998.

[196] V. N. Vapnik. *The Nature of Statistical Learning Theory*, chapter 4.6, pages 106–110. Springer, 2000.

[197] M. Viswanathan, C. S. Wallace, D. L. Dowe, and K. B. Korb. Finding cut-points in noisy binary sequences - A revised empirical evaluation. In *Australian Joint Conference on Artificial Intelligence*, volume 1747 of *Lecture Notes in Artificial Intelligence*, pages 405–416. Springer Verlag, 1999.

[198] C. S. Wallace. Inference and estimation by compact coding. Technical Report 84/46, Dept. of Computer Science, Monash University, Clayton 3168, Australia, August 1984.

[199] C. S. Wallace. An improved program for classification. In *Proc. of the 9th Australian Computer Science Conference (ACSC-9)*, pages 357–366, February 1986. Published as Proc. of ACSC-9, volume 8, number 1.

[200] C. S. Wallace. Classification by minimum-message-length encoding. In S. G. Akl et al, editor, *Advances in Computing and Information - ICCI '90*, volume 468 of *Lecture Notes in Computer Science (LNCS)*, pages 72–81. Springer-Verlag, May 1990.

[201] C. S. Wallace. A Model of Inductive Inference. Seminar, November 1992. Also on video, Dept. of Computer Science, Monash University, Clayton 3168, Australia, Wed. 25 Nov. 1992.

[202] C. S. Wallace. False oracles and SMML estimators. In D. L. Dowe, K. B. Korb, and J. J. Oliver, editors, *Proc. Int. Conf. on Information, Statistics and Induction in Science*, pages 304–316. World Scientific, 1996. ISBN 981-02-2824-4.

[203] C. S. Wallace. MML inference of predictive trees, graphs and nets. In A. Gammerman, editor, *Computational Learning and Probabilistic Reasoning*, pages 43–66. Wiley, 1996.

[204] C. S. Wallace. Multiple factor analysis by MML estimation. In *Proceedings of the Fourteenth Biennial Australian Statistical Conference (ASC-14)*, page 144, Queensland, Australia, July 1998.

[205] C. S. Wallace. On the selection of the order of a polynomial model. In W. Robb, editor, *Proc. of the 14th Biennial Australian Statistical Conf.*, page 145, Queensland, Australia, July 1998.

[206] C. S. Wallace. A brief history of MML (seminar), November 2003. Text available at http://www.allisons.org/ll/MML/20031120e/.

[207] C. S. Wallace. *Statistical and Inductive Inference by Minimum Message Length*. Springer, 2005.

[208] C. S. Wallace. *Statistical and Inductive Inference by Minimum Message Length*. Information Science and Statistics. SpringerVerlag, May 2005. ISBN 0-387-23795-X.

[209] C. S. Wallace and D. M. Boulton. An information measure for classification. *The Computer Journal*, 11(2):185–194, 1968.

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

[211] C. S. Wallace and D. L. Dowe. Estimation of the von Mises concentration parameter using minimum message length. In *Proc. 12th. Aust. Stat. Soc. Conf.*, 1994. See also technical report 93/193, Dept. of Computer Science, Monash University, Clayton 3168, Australia, 1993.

[212] C. S. Wallace and D. L. Dowe. Intrinsic classification by MML - the Snob program. In *Proc. 7th Australian Joint Conf. on Artificial Intelligence*, pages 37–44. World Scientific, November 1994.

[213] C. S. Wallace and D. L. Dowe. MML mixture modelling of multi-state, Poisson, von Mises circular and Gaussian distributions. In *Sixth International Workshop on Artificial Intelligence and Statistics, Society for AI and Statistics*, pages 529–536, San Francisco, USA, 1997.

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

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

[216] C. S. Wallace and D. L. Dowe. Rejoinder. *Computer J.*, 42(4):345–347, 1999.

[217] 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, January 2000.

[218] C. S. Wallace and P. R. Freeman. Estimation and inference by compact coding. *Journal of the Royal Statistical Society series B*, 49(3):240–265, 1987.

[219] C. S. Wallace and P. R. Freeman. Single-factor analysis by minimum message length estimation. *Journal of the Royal Statistical Society. Series B*, 54(1):195–209, 1992.

[220] C. S. Wallace and M. P. Georgeff. A general objective for inductive inference. Technical Report 83/32, Department of Computer Science, Monash University, Clayton, Australia, March 1983. Re-issued in June 1984 (perhaps more concisely) as TR No. 84/44.

[221] C. S. Wallace and K. B. Korb. A Bayesian learning agent. In C. S. Wallace, editor, *Research conference: Faculty of Computing and Information Technology*, page 19. Monash University Melbourne, 1994.

[222] C. S. Wallace and K. B. Korb. Learning linear causal models by MML sampling. In A. Gammerman, editor, *Causal Models and Intelligent Data Management.*, pages 89–111. Springer Verlag, 1999. TR 97/310, Dept. Computer Science, Monash University, Australia, June 1997.

[223] C. S. Wallace, K. B. Korb, and Honghua Dai. Causal discovery via MML. In *The Nineteenth International Conference on Machine Learning (ICML-2002)*, pages 516–524, 1996.

[224] C. S. Wallace and J. D. Patrick. Coding decision trees. *Machine Learning*, 11:7–22, 1993.

[225] G. I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40:159–196, 2000.

[226] G. I. Webb and Zijian Zheng. Multistrategy ensemble learning: Reducing error by combining ensemble learning techniques. *IEEE Trans. Knowl. Data Eng*, 16(8):980–991, 2004.

[227] I. H. Witten and E. Frank. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco, 2nd edition, 2005.

[228] H. Wold. Estimation of principal components and related models by iterative least squares. *Multivariate Analysis*, pages 391–420, 1966.

[229] Xiaoxin Yin and Jiawei Han. CPAR: Classification based on predictive association rules. In *SIAM International Conference on Data Mining*. San Francisco, CA, USA, May 2003.

[230] J. D. Zakis, I. Cosic, and D. L. Dowe. Classification of protein spectra derived for the resonant recognition model using the minimum message length principle. *Australian Comp. Sci. Conf. (ACSC-17)*, pages 209–216, January 1994.