TREE MODELS: A BAYESIAN PERSPECTIVE

By

Blaise F Egan

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF PHILOSOPHY
AT
QUEEN MARY, LONDON UNIVERSITY
MILE END ROAD, LONDON
NOVEMBER 2006

QUEEN MARY, LONDON UNIVERSITY
DEPARTMENT OF
MATHEMATICAL SCIENCES

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "**Tree Models: A Bayesian Perspective**" by **Blaise F Egan** in partial fulfillment of the requirements for the degree of **Master of Philosophy**.

Dated: November 2006

External Examiner: _____
David Hand

Research Supervisor: _____
Lawrence Pettit

Examining Committee: _____
James Smith

_____

QUEEN MARY, LONDON UNIVERSITY

Date: **November 2006**

Author: **Blaise F Egan**

Title: **Tree Models: A Bayesian Perspective**

Department: **Mathematical Sciences**

Degree: **M.Phil.** Convocation: **May** Year: **2007**

Signature of Author

To Glenys

Table of Contents

Abstract

Classical tree models represent an attempt to create nonparametric models which have good predictive powers as well a simple structure readily comprehensible by non-experts. Bayesian tree models have been created by a team consisting of Chipman, George and McCulloch and second team consisting of Denison, Mallick and Smith. Both approaches employ Green's Reversible Jump Markov Chain Monte Carlo technique to carry out a more effective search than the 'greedy' methods used classically.

The aim of this work is to evaluate both types of Bayesian tree models from a Bayesian perspective and compare them.

Acknowledgements

I would like to thank Lawrence Pettit, my supervisor, for his many suggestions, wise advice and continual support. I would also like to thank Peter Cochrane, formerly head of Advanced Applications and Technology division at British Telecommunications PLC, for his enlightened training policies. In addition, my managers over the years (Ken Totton, Rob Booth, Frances Hedley, Paul Garner and Nigel Barnes) who supported me and found the budgets to fund the fees, books and the study time needed to complete this course. Last, but not least, I must thank my life partner Glenys, whose love has encouraged me throughout the duration of this long course of study.

Chelmsford, Essex
August 12th 2006

Blaise F Egan

Chapter 1

A Short History of Trees

1.1 Introduction

Mathematical models have been at the core of the applied sciences since the 17^{th} century, when Newton showed that the movements of the celestial bodies could be predicted accurately many years in advance by means of deterministic models. In the 20^{th} century, statistical models were developed for situations where the relationships involved are of a statistical, not deterministic, character. These take the form

$$y = f(x_1, \ldots, x_k) + \text{ error}$$

That is, we predict the *response variable* y by the judicious use of the *predictor variables* x_1,...,x_k. (The xs and ys may be vectors). Our predictions are not expected to be perfect and there will be some random prediction error, but the prediction error will follow a known form of probability distribution. In the early part of the 20^{th} century the *normal linear regression*, of the form

$$y = \alpha + \sum_{i=1}^{k} \beta_i x_i + \varepsilon$$

$$\varepsilon \sim N(0, \sigma^2)$$

was pre-eminent. The model parameters α and the β s would be considered to be unknown real constants to be estimated from the data or, in a Bayesian formulation, probability distributions conditioned on the data. Model complexity was primarily limited by computational considerations and, as a result, early models often failed to capture all of the structure in the data. This is called *underfitting.* Rapid and sustained increases in the power of computer hardware in the latter half of the 20^{th} century led to easing of computational restrictions on model form and much was effort devoted to finding less restrictive model forms. The normal linear regression model was extended in many ways: transformations of the response variable or one or more of the predictor variables could be included and Box and Cox (1964) [12] identified a class of useful transformations. The linear expression on the right hand side of Equation 2 could be replaced by an expression with nonlinear terms (Bates and Watts, 1988)[9]. For example, S-shaped functions such as the logistic function occur frequently in marketing and biological applications. The *multilayer perceptron* (Rumelhart et al., 1986, [115]), in particular, achieved widespread (some would say, excessive) popularity.

Nelder and Wedderburn (1972) [102] formalised the Generalised Linear Model, which allowed the normal error term to be replaced by an error term from a large class of distributions (the exponential families) which includes many familiar distributions, such as the binomial, Poisson and the gamma. Tibshirani and Hastie (1990) [126] formalised the Generalised Additive Model, where the model constructed from basis

functions, that is, we approximate f by $\hat{f}$, a linear combination of simple functions $\{B_i\}$.

$$\hat{f}(x) = \sum_i \beta_i B_i(x)$$

The basis functions can be drawn from a wide variety of function types, including tree models (see below) and wavelets (Bruce and Gao 1996) [18]. With a plethora of models to choose from, model selection soon became an issue. A widespread problem was the tendency of many practitioners to use all of the data to estimate their chosen model structure without validating it on a different data set. This caused the model fitting procedure to fit too closely to the training set, incorporating as structural elements of the model features which are only artefacts of the training set. This is called *overfitting*.

1.2 What is a tree?

In graph theory (Hillier and Lieberman, 1990, [71]) a graph is a collection of *nodes*, which may be connected by *arcs* (also known as *edges*).

- Arcs may have a direction of flow associated with them. Such an arc is called a *directed arc*.
- An arc in which flow can occur in either direction is called an *undirected arc*. An undirected arc is thus a directed arc, but the converse is not true. (If no flow is possible between two nodes this is denoted by the absence of an arc connecting them.)

- A sequence of distinct arcs leading from one node to another is called a *path*. (The word 'distinct' is used here to imply that the definition does not allow the retracing of one's steps.)

- A path from node i to node j whose constituent arcs are all either undirected or have a flow towards j is said to be a *directed path*.

- A path from node i to node j whose constituent arcs are undirected or have a flow towards or away from j is said to be an *undirected path*. A directed path is thus an undirected path, though the converse is not true.

- A *cycle* is a path from a node to itself. It is a directed (or undirected) cycle if it is a directed (or undirected) path.

- Two nodes are said to be *connected* if there is an undirected path between them.

- A *connected network* is a collection of nodes such that every pair of nodes is connected.

- Finally, a connected network with no undirected cycles is a *tree*.

Diagrams having the structure of a tree are widely used when visualising a sequence of decisions. Such decision trees are also a natural way of visualising the tree algorithms we discuss below.

1.3 The Automatic Interaction Detector (AID)

Morgan and Sonquist (1963) [101] described a number of problems associated with applying multiple regression models to cross-section survey data. They frequently

found that their models had to incorporate interactions and non-linearity. Classical model-building methodology requires that the model is specified *a priori* and the data are then used to estimate the parameters. In practice, it is hard to say beforehand which interactions will be present and whether particular relationships will be linear or nonlinear. Morgan and Sonquist introduced a technique, the Automatic Interaction Detector, (AID) for use with data consisting of a numeric response variable and categorical predictor variables (factors), whereby the data were repeatedly subdivided on the basis of the reduction in the error sum of squares. This was done as follows. The total sum of squares, $\sum x^2$, can be divided into two parts, the part explained by the mean,Nx^2, and the unexplained part. The explained part can be rewritten as $\frac{(\sum x)^2}{N}$ and the unexplained part is the variation around the mean, hence

$$\sum x^2 = \left\{ \sum x^2 - \frac{\left(\sum x\right)^2}{N} \right\} + \left\{ \frac{\left(\sum x\right)^2}{N} \right\} = \sum (x - \bar{x})^2 + \frac{\left(\sum x\right)^2}{N} \qquad (1.3.1)$$

Now suppose we now divide the data into two subsets of size N_1 and N_2. The explained part becomes $N_1\bar{X}_1^2 + N_2\bar{X}_2^2$. The partition which increases this as much as possible over the original value of NX^2 is to be preferred. The process can be repeated recursively for each of these subsets until the increase in the explained sum of squares is negligible. The result can be expressed as a tree diagram such as Fig. 1.1 which is a modified form of a figure in Morgan and Sonquist's paper. In order to use the tree to make a prediction for a new observation we decide which terminal set it belongs to. For example, we first look at the race variable. Suppose our observation has race not equal to African-American. We then look at the Age variable to see if this person is 65 or older. Suppose they are. We then look at the school variable to see if they are a high school graduate. Suppose they are. Now there are no more

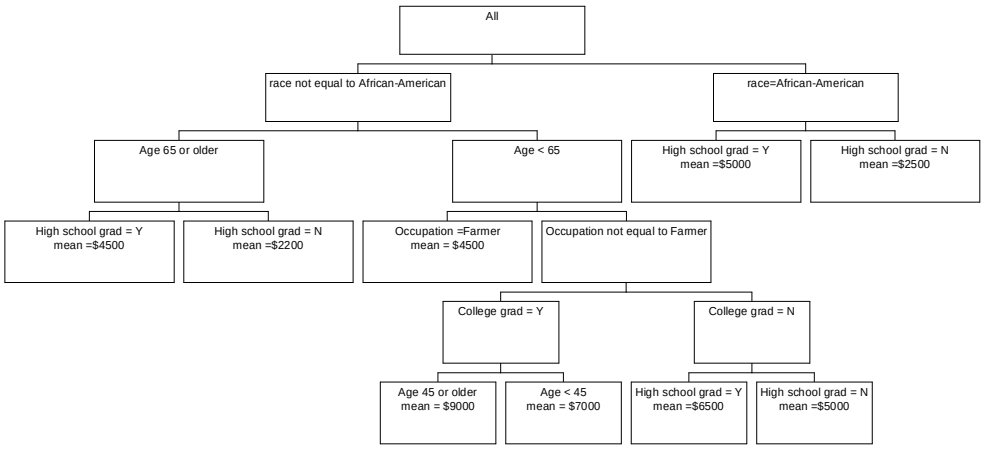

Figure 1.1: Tree diagram, from Sonquist and Morgan.

splits and we are at a terminal node. Our prediction is simply the mean of the values at that node.

Points to note:

- The method is an *algorithm* not a *model*, by which we mean that it is a sequence of instructions that lead us to a number to use as our prediction, whereas a model is a full mathematical description of the data. A model would, for example, enable us to create another data set very similar to this one or, perhaps more usefully, it would enable us to make probability statements about the likely range of error of our prediction, to define and draw inferences about outliers, etc.

- The splitting criterion is that the error sum of squares is decreased by a non-negligible amount. Morgan and Sonquist define this as a decrease of at least 1% of the total sum of squares. This is an arbitrary criterion and even if it should be found to work reasonably well for some data sets one may wonder about its general applicability. As noted by Kass [84], decreasing the error sum

of squares is equivalent to maximising the between-subgroup sum of squares.

- The stopping criterion is that no further splits can reduce the error sum of squares by 2 percent or more. (This limits the tree size to 51 subgroups.)
- The discontinuity associated with the splits is an unattractive feature, as in many contexts we would expect the response function to be smooth.
- The method only deals with categorical predictors and numeric response variables.

The AID method became popular in the late 1960s and early 1970s with the availability of computer programs to carry out the calculations. The method was applied to many areas, including education (Orr, 1972, [106], population studies (Ross and Bang, 1966, [113]) and market research (Assael, 1970, [6]). THAID, a version of AID that uses a categorical response variable instead of a numeric one was developed by Morgan and Messenger (1973) [100]. At the same time, stepwise regression methods were also becoming popular and these operate in a similar spirit of data-driven model building.

1.4 The Chi-Squared Automatic Interaction Detector (CHAID)

1.4.1 Introduction

A perceived deficiency of the AID algorithm was that there was no statistical test of the significance of the AID tree, comparable to the F test used for the overall

significance of a regression model. (Since AID maximises the sum of squares between subgroups it is not valid to test the difference between two subgroups using a t test.) A fundamental reason why such tests are not possible within AID is that AID is not a model. If we want to incorporate such tests AID must be made into a model. From a statistical perspective, the tree-growing algorithm is an automated way of stratifying the data into distinct subsets. Once we have stratified the data set it then takes the form of a collection of subsets of the original data, namely, the bottom nodes of the tree. In order to make a model out of AID we need to represent the observations at the bottom nodes by a probability distribution. Obvious candidates for such distributions are the normal distribution ror numeric responses and the multinomial for categorical responses. Based on these choices, Kass (1976) [84] suggested a way of testing the significance of a new split and suggested that the choice of the next split to make should be made on the basis of which split was the "most statistically significant". Kass also suggested that the stopping criterion should be that one stops when there are no new "statistically significant" splits available to make. Kass's application of the Bonferroni correction was critiqued and refined by Biggs et al. (1991) [11]. Despite the numerous dubious points in the rationale for the CHAID algorithm it was marketed successfully and was widely used by marketing analysts and others (Thrasher, 1991 [124], Houghton and Oulabi, 1993 [76]). By the 1980s some of the leading vendors of statistical software packages had implemented extended forms of CHAID which coped with either numeric or categorical dependent variables and numeric or categorical predictor variables. Although AID and CHAID were generally well received in the social sciences and in the marketing community, there were some critics.

- Einhorn (1972) [43] showed by simulation that unless sample sizes of 2000 or

more were available AID could give spurious results due to small numbers of observations at some leaf nodes. He also drew attention to the fact that many users were not validating their results on fresh data and thus spurious results would go undetected.

- Doyle (1973) [41] drew attention to a number of additional problems with AID exemplified in a paper by Heald (1972) [70]. In a regression model all the observations contribute to estimation of all the parameters. AID does not share this property and thus sample sizes should be large enough to ensure that in each leaf node the test for whether a split should be made has sufficient power. A related issue, not mentioned by Doyle, is that for large sample sizes a CHAID-type significance test will become so sensitive that it will start finding splits which, though statistically significant, are not commercially significant. As sample sizes become very large and sampling error effectively is effectively extinguished, it becomes overwhelmingly likely that spurious splits will be found due to small artefacts in the data.

- If two predictors are correlated then the choice of which one of them should be chosen for the next split may be finely balanced. Each split made has a strong influence on the subsequent evolution of the tree and yet key splitting decisions may, in effect, be taken randomly.

The combined effect of these problems is that tree models (of all types, in fact) are exquisitely sensitive to small changes in the data used to construct them. This fact is apparent to anyone who uses such methods in an applied context for any length of time. Someone may build a tree using all the available data at a certain point in time

and then rebuild it a month later with additional data included, only to find that the tree is now substantially different. Nevertheless, such doubts have not prevented the widespread use of CHAID.

1.5 CART

1.5.1 General

The publication in 1984 of the book *Classification and Regression Trees* by Breiman, Friedman Olshen and Stone [16] was a major landmark in the history of tree models. Breiman et al. constructed their trees using methods that did not involve significance tests for the choice of variable to be split, thus side-stepping some of the problems with that approach. Furthermore, their research showed that the method of splitting is not crucial and they drew attention to the problem of overfitting. To combat overfitting Breiman suggested a strategy of deliberately producing a tree that is too big then reducing it to a smaller one by a formal process which they called *pruning*. They showed that although the splitting criterion used makes little difference, the form of pruning used makes a lot of difference to the predictive accuracy. Breiman et al. developed a sophisticated pruning algorithm called *cost-complexity pruning*. Breiman and his colleagues marketed their software implementation of CART successfully and it has been used in a wide variety of applications. The CART software has a few features not incorporated in other algorithms. It handles missing values automatically by choosing a surrogate predictor variable to replace the one with missing values, and it can be configured to allow splits on linear combinations of predictor variables instead of a single predictor.

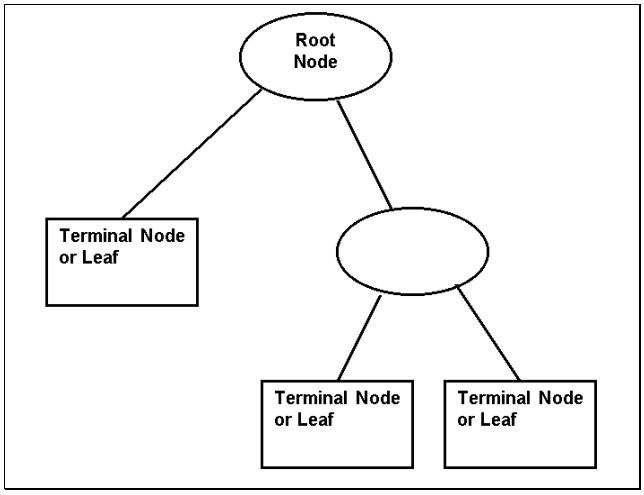

Figure 1.2: Tree terminology.

1.5.2 Splitting criteria

Breiman suggest two different splitting criteria for the case where the response variable is categorical, the *Gini* index and the *twoing* criterion. We want to choose a split that makes the resulting subgroups as different from each as possible, and as homogeneous as possible. Thus if the response variable has levels A and B we would ideally like to find a split which puts all the As in one group and all the Bs in another. A node with all identical responses is said to be *pure*. So we are looking for measures of node purity, or, equivalently, node impurity. Breiman uses conventional computer science terminology for trees. The initial node is called the *root node* and the bottom nodes are called *terminal nodes* or *leaves*. See Fig. 1.2

Suppose we are at terminal node t. Observations at a terminal node are classified according to the majority class at the node. There are K distinct classes. We denote the proportion of observations at t which are in class i by $\hat{p}(i|t)$. The Gini index

impurity measure for trees is given by

$$I(t) = \sum_{i \neq j} \hat{p}(i|t)\hat{p}(j|t) = \sum_{k=1}^{K} \hat{p}(i|t)(1 - \hat{p}(i|t))$$

This can be regarded as an estimate of the misclassification rate at the node, as $\hat{p}(i|t)\hat{p}(j|t)$ is an estimate of the probability that a randomly chosen observation at t is classified as class i when it fact it belongs to class j. This estimate is, however, a *resubstitution estimate*: it is based on data that was used to construct the tree and thus it has a downward bias since the tree has been constructed to have minimum Gini index. The Gini index has a secondary interpretation. If at a node we score each observation 1 if is in class k and zero otherwise, the variance over the node is $\hat{p}(i|t)(1 - \hat{p}(i|t))$, which when summed over all K classes, gives the Gini index. The Gini index is closely related to the Gini *coefficient*, a criterion used for assessment of classification rules. An alternative splitting criterion used by Breiman, not related to an impurity measure, is the twoing criterion. Given that node t is split into a left node t_L and a right node t_R choose the split which maximises

$$\frac{P_L P_R}{4} \left\{ \sum_j |p(j|t_L) - p(j|t_R)| \right\}^2$$

where P_L and P_R are the proportion of observations at t going to the left and right subtrees respectively under the proposed split. The term in braces emphasises the fact that we want the proportions of each class in the left and right subtrees to be as different as possible. The factor outside the braces imposes a preference for equally sized subtrees.

1.5.3 Pruning

The method used by Breiman is to first identify a nested sequence of trees of different sizes, ranging from the initial tree, which is deliberately grown too big, to a tree consisting of just the root node of T_{max}. Interestingly, Breiman first discuss a method which they reject for the reason that it does not lead to a nested sequence of trees, and which can add back on branches that have been pruned off at an earlier stage. They regard this as undesirable, and yet, as we shall see, this is one of the advantages claimed by Chipman, George and McCulloch (1998)[27] for their method. The second stage is to compare the different sized trees to decide on the right size. Breiman developed an ingenious algorithm for generating a nested sequence of subtrees, called *weakest link cutting*. We define the *cost-complexity* measure $R_\alpha(T)$ of a tree T as follows:

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}| \tag{1.5.1}$$

where $R(T)$ is a resubstitution-based estimate of the misclassification cost, $\tilde{T}$ denotes the collection of leaf nodes of the tree T, and $|\tilde{T}|$ is the number of nodes of $\tilde{T}$. The real number α is an arbitrary constant which measures the relative disbenefit of one extra leaf node compared to a unit increase in classification cost. We consider the effect of different values of α and note that, for any tree T, as α changes from very small to very large values the tree with lowest $R_\alpha(T)$ changes from being the entire tree T to being the tree consisting of just the root node of T. We denote by T_{max} the tree which is so large that every observation is in its own leaf node and we note that $R(T_{max}) = 0$. It may be that we can prune off some nodes from T_{max} without increasing the resubstitution error rate $R()$. If so, the the smallest subtree

with this property is denoted T_1, i.e. $R(T_1) = R(T_{max})$. For any subtree T_* of T_1 we define $R(T_*) = \sum_{t' \in \tilde{T_1}} R(t')$. Breiman show that for any non-leaf node t of T_1 we have $R(t) > R(T_*)$. Denote by $\{t\}$ the subtree of T_* consisting of just the root node. Then, from Equation 3, $R_\alpha(t) = R(\{t\}) + \alpha$. For small α we have $R_\alpha(T_*) < R_\alpha(\{t\})$ but as we increase α there comes a point when

$$R_\alpha(T_*) = R_\alpha(\{t\}) \tag{1.5.2}$$

This critical value of α can be calculated by solving Equation 1.5.1.

$$\alpha = \frac{R(t) - R(T_*)}{|\tilde{T_*}| - 1} \tag{1.5.3}$$

Now, given a node t,$t \in T_1$, we define a function $g(t)$,

$$g(t) = \begin{cases} \frac{R(t) - R(T_*)}{|\tilde{T_*}| - 1} & t \notin T_1 \\ +\infty & t \in T_1 \end{cases}$$

The we define the *weakest link* $\bar{t}$ in T_1 as the node such that

$$g(\bar{t}) = \min_{t \in T_1} g(t)$$

Thus the weakest link is the node that first has $R_\alpha(T_*) = R_\alpha(\{t\})$. Breiman's methodology is to prune off at each stage the subtree having as its root node the weakest link. They produce existence proofs that this can be done and show that this leads to a nested sequence of trees. They have an efficient algorithm for finding weakest links. Having produced a nested sequence of trees of different sizes, they then proceed to compare these to determine which is the right sized tree. The comparison

is done by V-fold cross-validation unless the data set is very large, in which case a simple random sample, independent of the training set, is taken to act as a test set.

1.5.4 Information Theoretic Approaches

Breiman's book was influential in the machine learning community and inspired some related work. In particular, Quinlan's 1993 book *C4.5: Programs for Machine Learning* [109] gave detailed documentation and public domain source code (for non-commercial applications) in the C language for an efficient computer program and thus reached a relatively wide audience. It is well known in the computer science community. Although Quinlan has drawn on the ideas of Breiman, the program C4.5 does not seem to be superior to CART except in the area of run-time performance. It does not handle the regression problem (*i.e.* a numeric response variable) at all and misses out some of the attractive features of CART, such as its handling of missing values. C4.5 uses a split criterion based on Shannon information and a different pruning method. Claude Shannon's 1948 paper entitled *A Mathematical Theory of Communication* [118] founded two separate mathematical disciplines: information theory and coding theory. Suppose a message source can emit k different discrete messages with probabilities $p_1, p_2, \ldots, p_k$ and that these events are independent. We define the information content of message i , measured in 'bits', as $I_i = \log_2\left(\frac{1}{p_i}\right)$ and the expected information, or *entropy*, of a message yet to be received, as $\mathbb{E}\{I_i\} = -\sum_{i=1}^{k} p_i \log_2(p_i)$. With a 2 letter alphabet, with letter probabilities p_1 and p_2, the entropy is maximised when the two probabilities are equal and at the minimum when the probabilities are 0 and 1. Thus, although he does not say so, Breiman's idea of a 'purity function' for class probabilities has some basis in

information theory. (The result also holds for the case where the number of classes is greater than two.) Quinlan's original splitting criterion, implemented in his program ID3, was that splits should be made so as to maximise entropy. His later program C4.5 uses 'entropy gain ratio' as the splitting criterion. This compares the percentage increase in Shannon information of each potential split. We assume that the Y values are categories $C_1, \ldots, C_k$ The following description of entropy gain ratio is drawn from Quinlan (1993)[109] :

Suppose at node t we pick an observation at random and determine its class. The information thus gained is $-\log_2(\hat{p}(i|t))$ and the expected information gain over all classes is given by

$$\text{Info}(t) = -\sum_{j=1}^{k} \hat{p}(i|t) \log_2\{\hat{p}(i|t)\}$$

If we have a splitting criterion X which puts the observations at t into subnodes $t_1, \ldots t_n$ the expected information is the weighted sum over subnodes:

$$\text{Info}_X(t) = \sum_{i=1}^{n} \frac{|t_i|}{|t|} \text{Info}(t_i)$$

where $|t|$ denotes the number of observations at node t.

The quantity

$$\text{gain}(X) = \text{info}(t) - \text{info}_X(t)$$

represents the information gain from partitioning node t in accordance with the splitting rule X. Prior to C4.5, Quinlan used gain(X) as the splitting criterion for his decision trees. However, it was empirically found to be biased towards tests with many outcomes. He therefore changed to the *gain ratio* defined as follows. Suppose

we select an observation at random and announce not its class, but the subnode into it is put into by the splitting rule X. By analogy with the definition of info(t) we have

$$\text{split info}(t) = -\sum_{i=1}^{n} \frac{|t_i|}{t} \cdot \log_2 \left(\frac{|t_i|}{t} \right)$$

This represents the information gained by learning the subnode of a case rather than its class. Quinlan considers the ratio

$$\text{gain ratio}(X) = \text{gain}(X)/\text{split info } (X)$$

and interprets it as the proportion of the information created by the split which is useful for the purpose of classification. His C4.5 program chooses the split which maximises the gain ratio, subject to a constraint that the gain must be "reasonably large". The latter constraint is necessary because the ratio becomes unstable if there is little information in the split. C4.5 stops when the information gain becomes small (specifically, smaller than the average gain over all tests so far examined) and then it starts pruning. The C4.5 pruning mechanism is based on using the 75% upper confidence bound (incorrectly referred to as a "25% confidence level") for the resubstitution error rate, assuming these errors follow a binomial distribution. (The idea is to use a pessimistic estimate of the true error rate.) This error rate is then used to compute the expected number of errors with and without pruning off a particular subtree. The decision whether to prune or not is based on whichever of these two leads to the lower value of this expected error rate.

1.5.5 Overview

At first glance, we seem to have here three different approaches to the same problem, and it is natural to wonder how they compare. Before doing that, it may be useful to check whether the authors were in fact addressing the same problems. Morgan and Sonquist were concerned at the need to specify in advance the structure of a linear model (multiple regression, in fact) before its parameters could be estimated. In particular, they found it awkward to determine whether interaction terms would be needed. Breiman et al. were concerned initially to further the work on AID and THAID. They focusedon a real-life practical problem: development of a model to categorise heart attack patients into those who could be sent home shortly after hospital treatment and those who should be kept in hospital for longer. They developed a small, medically credible tree model with better classification accuracy than human physicians. Quinlan's motivation was research in expert systems. For computers to be able to replicate human cognitive tasks such as classification the computer must contain knowledge in the form of a model or an algorithm for the problem. Earlier work in so-called 'expert systems' was based on eliciting this knowledge from a human expert and representing it in some form in the computer. The elicitation process was found to be very time-consuming and difficult, so attempts were then made to construct algorithms which could glean knowledge from data *without* the aid of a human being. This led to self-constructing models such as artificial neural nets (Rumelhart and McClelland, 1986)[114] and decision trees. From a 21st century perspective, forty years after Morgan and Sonquist's early work, we can appreciate the need to use a less restrictive model class and there is a much wider choice of models to choose from these days. Modern non-Bayesian alternatives to trees, at least for

the regression problem, would include Projection Pursuit Regression (Friedman and Stuetzle, 1981)[52], and MARS (Friedman, 1991)[50]. These models allow the user to set up a believable regression model in which the model choice, as far as interactions is concerned, is entirely data-driven. As we shall see later, flexible Bayesian alternatives for both the regression problem and the classification problem can be constructed.

Chapter 2

Bayesian Tree Models

2.1 The Bayesian Approach

The Bayesian approach to estimating the parameters of a model is to specify any contextual knowledge about the parameters in the form of a probability distribution called the *prior distribution* and then compute the conditional distribution of the parameters given the data, called the *posterior distribution*. This is usually done via Bayes' theorem for probability densities:

$$p(\theta|y) = \frac{p(\theta)p(y|\theta}{p(y)} \tag{2.1.1}$$

where y is a vector of observed data, θ is the unobserved vector of parameters of interest. The observed value of y can usually arise in a number of different ways, corresponding to different values of θ, thus the calculation of the denominator is in general complicated. For continuous θ we have $p(y) = \int p(y|\theta)p(\theta)d\theta$ If θ is a vector of k dimensions this a k-dimensional volume integral. The likelihood $p(y|\theta)$ provides the model which links the unobserved parameter θ to the observed values y. The result of all this calculation is the posterior, which holds all information needed for

statistical inference. In practice, summaries of the posterior are often found to be useful: confidence intervals (known as *credible intervals* in Bayesian terminology), quantiles, moments, etc. The choice of summaries depends on the situation at hand.

2.2 Bayesian Computation

Computational problems associated with evaluation of 2.1.1 caused serious problems with the implementation of Bayesian methods for large models until the rediscovery of the Gibbs sampler algorithm by Geman and Geman (1984)[55] . The Gibbs sampler is the simplest example of a class of powerful methods called Markov Chain Monte Carlo methods or MCMC (Gilks et al., 1996)[57] These methods work by sampling the distribution of interest at a set of randomly chosen, but correlated, points. The sample points form a Markov chain constructed to have as its equilibrium distribution the posterior distribution that we wish to compute. Once the chain has converged we can take as many samples from the distribution as we want. Even though we do not have a closed-form formula for the posterior we can use kernel smoothing to plot the shape of the distribution and we can easily find quantiles by ranking the samples. The Gibbs sampler is not suitable for use with Bayesian tree models and neither are conventional MCMC models. Conventional MCMC jumps between spaces of equal dimension, *i.e.* models with the same number of parameters. A more general form of MCMC, the Reversible Jump Markov Chain Monte Carlo (RJMCMC) algorithm (Green, 1995)[59], extends MCMC to jumping between models of different dimensions. RJMCMC is crucial to Bayesian tree models since we need to consider trees of differing sizes (i.e. numbers of leaf nodes). Using RJMCMC we can take a random walk around the space of trees but 'gravitating' to good trees by virtue of the

existence of a limiting distribution. A major problem facing any tree method is the sheer size of the tree space. There is no possibility of visiting a substantial fraction of the possible trees. All of the non-Bayesian tree methods described in chapter 1 are 'greedy' algorithms: they look one step ahead and try to identify the best next step (choice of variable or split point) and that is the one they take. The problem with greedy methods is that they cannot reach a good tree two or more steps away if the first step towards them is not the 'best' choice. In order to find such good trees we need a method that occasionally chooses steps that do not at first sight look as good as other available choices. The RJMCMC method does this. A non-Bayesian method for doing something similar would be simulated annealing (Ripley, 1987)[111]. (In fact, both simulated annealing and the Metropolis-Hastings algorithm, an important MCMC technique, were independently developed from the same paper written by Metropolis et al. in 1953.)[99]

2.3 Bayesian Model Selection

Bayes' Theorem can also be used for the problem of model selection. A prior probability is placed on each model in a defined model class and these are revised by conditioning on the data. In practical model selection problems it is rare that all the models under consideration have the same number of parameters, but the use of RJMCMC allows us to take this in our stride.

2.4 Bayesian Tree Models

There are two Bayesian tree models, one developed by Chipman, George and McCulloch (known hereafter as CGM)[27] and the other developed independently at the same time by Denison, Mallick and Smith (DMS).[38]

2.4.1 Chipman, George and McCulloch's Classification Model

For the classification problem CGM use a stochastic process to provide prior tree probabilities. Let y_{ij} denote the jth observation of y in the ith leaf. $i = 1, \ldots, b, j = 1, \ldots, n_i$.

Let $Y \equiv (Y_1, \ldots, Y_b)'$, where $(y_{i1}, \ldots, y_{in})'$ and define X and X_j similarly.

The data are assumed to come from one of K categories $C_1, \ldots, C_K$ under the multinomial distribution

$$y_{i1}, \ldots, y_{in_i} | \theta_i \text{ Bin}(Y_i|\mu_i) = \prod_{j=1}^{n} \prod_{k=1}^{K} \mu_{ij}^{I(y_{ij} \in C_k)}$$

where $\theta_i = \mu_i \equiv (\mu_{i1}, \ldots, \mu_{iK}), \mu_{ik} \geq 0$ and $\sum_k \mu_{ik} = 1$

Also $p(y_{ij} \in C_k|\mu_i) = \mu_{ik}$

Under the conjugate prior $\mu_1, \ldots, \mu_b|T$ iid $\sim$ Dirichlet$(\mu_i|\alpha) \propto \prod_{j=1}^{K} \mu_{ij}^{\alpha_j - 1}$

$p(Y|X, T)$ where Y is the response, X is a vector of covariates and T is the tree, is marginalised analytically to give

$$p(Y|X, T) = \left(\frac{\Gamma\left(\sum_k \alpha_k\right)}{\prod_k \Gamma\left(\alpha_k\right)} \right)^b \prod_{i=1}^{b} \frac{\prod_k \Gamma(n_{ik} + \alpha_k)}{\Gamma(n_i + \sum_k \alpha_k)} \tag{2.4.1}$$

where

$$\begin{aligned} lcln_{ik} &= \sum_j I\left(y_{ij} \in C_k\right) \\ n_i &= \sum_k n_{ik} \\ k &= 1, \ldots, K \end{aligned}$$

As with standard CART we start with an initial tree, usually the trivial tree consisting of a single leaf node. We then have available to us a selection of moves which can take us through the space of trees. The choice of move is made by a Metropolis-Hastings step. Strictly speaking this is not Reversible Jump Metropolis-Hastings since the continuous parameters have been integrated out and the MH step is only over the discrete parameters. This is what Tierney (1994)[127] calls a hybrid sampler. The available move types are:

1. Move type: GROW

   We randomly choose a leaf, a split variable X_i and a split point s. We split the data at the leaf into the two sets $X_i \leq s$ and $X_i > s$. This provides us with two new daughter nodes representing the partition of the data set. Diagrammatically, this is shown in Fig. 2.1

2. Move type: PRUNE

   Randomly choose a leaf then delete both it and its sister node. See Fig. 2.2

3. Move type: CHANGE

   Randomly choose a node with children but no grandchildren, then change the splitting variable and the split point it uses. See Fig. 2.3

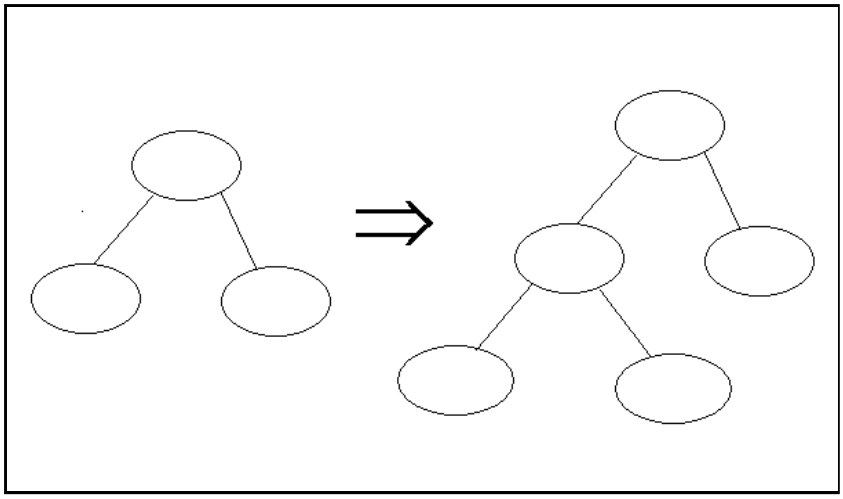

Figure 2.1: Move type GROW

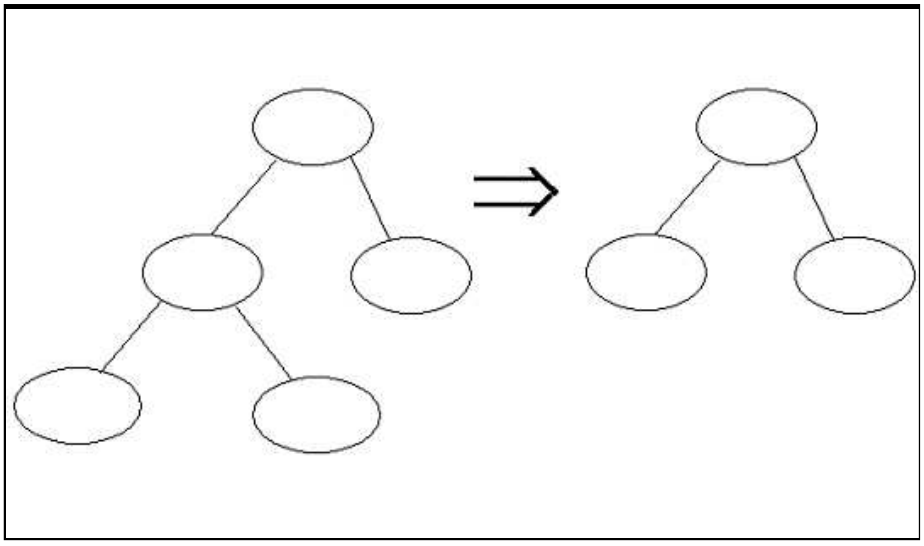

Figure 2.2: Move type PRUNE

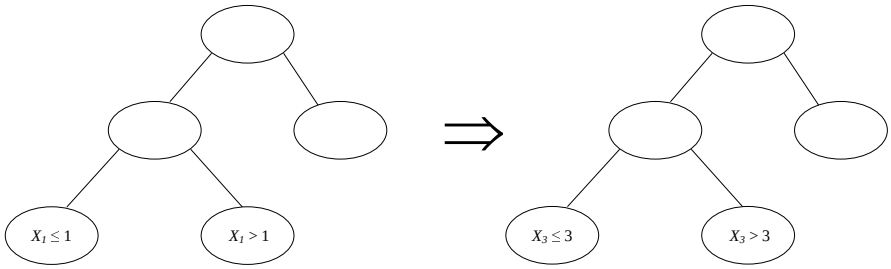

Figure 2.3: Move type CHANGE

4. Other possible move types

   These are the moves implemented by CGM in their software codes, but other move sets are possible. DMS use a move similar to c), but which changes the splitting variable instead of the split point. CGM also suggest a move called SWAP which randomly picks a parent-child pair of nodes which are both intermediate, then swaps their splitting rules. If the sister node already has the same splitting rule then swap the splitting rule of the parent with that of both children. Other move types are possible but it should be borne in mind that many tree spaces will have a large number of sharp local modes and thus it is necessary to move around the tree space in small steps. It would be possible to add or prune off large subtrees but these moves may have very low probability and hence could imply a poor move acceptance rate. The move type is chosen by a Metropolis-Hastings step as follows. Generate a candidate tree T* with probability distribution $q(T_b, T_{i+1})$. Set $T_{i+1} = T^*$ with probability

$$\alpha(T^i, T^{i+1}) = \min\left(\frac{q(T^*, T^i)}{q(T^i, T^*)}, \frac{q(Y|X, T^*)p(T^*)}{q(Y|X, T^i)p(T^i)}, 1\right) \tag{2.4.2}$$

   Otherwise set $T_{i+1} = T_i$.

Under weak conditions this Markov chain has a limiting distribution $p(T|Y,X)$ (Tierney, 1994) [**?**] and thus, in theory at least, it tends to gravitate to 'good' trees. However, the space of trees which can be fitted to even a modestly sized data set is enormous and with current hardware and software technology it is not possible to visit even a small fraction of these trees, nor is it likely that the Markov chain will reach equilibrium in a run time that is likely to be acceptable to users (a small number of hours, say). We should consider the process as a stochastic search, comparable to a simulated annealing search such as that used by Sutton (1991)[122]. We content ourselves with the hope that, although the tree search method is far from comprehensive and optimal, it is nevertheless likely to be superior to the greedy algorithms currently in widespread use.

2.4.2 Denison, Mallick and Smith's Classification Model

DMS also use a multinomial likelihood with a Dirichlet prior but they do not marginalise analytically. They set the Dirichlet prior to be a uniform prior and work with the means of the sampling distributions of the p's instead of drawing from them, i.e.

$$p_{ij} = \frac{\text{Number of observations at leaf } i \text{ which are classified as class } j}{\text{Number of observations at leaf } i}$$

DMS claim that this speeds up their computation. Interestingly, CGM claim that their use of analytical marginalisation speeds up their calculation. (We have put these claims to empirical test in Chapter 3.) For the remainder of the prior specification they assume:

- uniform distributions over the space of splitting variables and then

- uniform distributions over the set of possible values for the splitting points.

Whereas CGM assume that the user may have prior knowledge about which variables to split and provide him with the ability to control this, DMS are sceptical about the usefulness of such flexibility and do not allow the user to control it. DMS treat all trees that have the same number of leaves as having the same prior probability irrespective of shape, but CGM use an additional shape parameter that enables the user give to higher or lower probability to 'bushier' trees. Both CGM and DMS report that these uniform distributions seem to work well in practice, but CGM note that there is an issue if some predictor variables have more levels than others. A particular level in a variable that has many levels will have lower probability than the same level would have in a variable with fewer levels. (This is an aspect of the general Bayesian problem of specifying ignorance by means of prior distributions. One has to be very careful to specify exactly what is intended. Do you want to consider all predictor variables as equally likely *a priori*, or do you want to say that all *levels* of all variables are equally likely?) The DMS approach treats trees with the same number of leaves as equally likely, so the prior for k, the number of leaf nodes, is obtained from a Poisson modified to exclude 0.

$$p(k) = \frac{\lambda^k}{(e^\lambda - 1)}, k = 1, 2, \ldots$$

An arbitrary practical upper bound of 256 is imposed.

2.4.3 Chipman, George and McCulloch's Regression Model

The regression model takes the following form

$$y_{i1}, \ldots, y_{in} | \theta_i \text{ iid} \sim N(\mu_i, \sigma^2), i = 1, \ldots, b \tag{2.4.3}$$

where $\theta_i = (\mu_i, \sigma)$.

CGM refer to this as the mean shift model. Note the common variance across nodes. There is also the mean-variance shift model.

$$y_{i1}, \ldots, y_{in} | \theta_i \text{ iid} \sim N(\mu_i, \sigma_i^2), i = 1, \ldots, b \tag{2.4.4}$$

where $\theta_i = (\mu_i, \sigma_i)$.

As with the classification model, the priors are defined by a stochastic process. For the mean shift model CGM use the standard conjugate prior for $\Theta|T$:

$$\mu_1, \ldots, \mu_b | \sigma, T \text{ iid} \sim N\left(\bar{\mu}, \frac{\sigma^2}{a}\right)$$

and $\sigma^2|T \sim \text{Inv-Gamma}\left(\frac{\nu}{2}, \frac{\nu\lambda}{2}\right)$, which is equivalent to $\frac{\nu\lambda}{\sigma^2} \sim \chi^2_\nu$

As with the classification model, the parameter vector Θ can be integrated out to yield

$$p(Y|X,T) = \frac{ca^{b/2}}{\prod_{i=1}^{b} \sqrt{n_i + a}} \left(\sum_{i=1}^{b} (s_i + d_i) + \nu\lambda \right)^{\frac{-(n+\nu)}{2}} \tag{2.4.5}$$

where

$$\begin{aligned} s_i &= (n_i - 1) \text{ times the sample variance of the } Y_i \text{ values,} \\ d_i &= \frac{n_i a}{n_i + a} (\bar{y}_i - \bar{\mu})^2 \\ y_i &\quad \text{ is the average value of the } Y_i \end{aligned}$$

For the mean-variance model this becomes

$$\mu_1, \ldots, \mu_b | T \text{ iid} \sim N\left(\bar{\mu}, \frac{\sigma_i^2}{a}\right)$$

and

$$\sigma^2 | T \sim \text{Inv-Gamma}\left(\frac{\nu}{2}, \frac{\nu\lambda}{2}\right)$$

Again, Θ can be integrated out to yield

$$p(Y|X,T) = \prod_{i=1}^{b} \left\{ \pi^{-\frac{n}{2}} (\lambda\nu)^{\frac{\nu}{2}} \frac{\sqrt{a}}{\sqrt{n_i + a}} \frac{\Gamma(\frac{n_i+\nu}{2})}{\Gamma(\frac{\nu}{2})} \left(s_i + d_i + \nu\lambda^{-\frac{n_i+\nu}{2}} \right) \right\} \tag{2.4.6}$$

For the mean-shift model CGM suggest that the prior parameters $\nu, \lambda, \bar{\mu}$ and a can be determined automatically from the data. The idea is that since the tree is to be constructed so that the variation in Y is to be minimised, so that ν and λ can be chosen so that s_Y, the sample standard deviation of Y, is in the right tail of the prior for σ. $\bar{\mu}$ and a are to be chosen so that the prior for μ covers the range of Y. When the mean-variance model is used the model can more easily explain different values of σ, so that ν and λ should be chosen to put s_Y near the centre of the prior distribution for σ instead of in the tail. CGM also have a mechanism for adjusting the size and 'bushiness' of the tree. The probability of a split is given by

$$p_{SPLIT}(\eta, T) = \alpha (1 + d_\eta)^{-\beta} \tag{2.4.7}$$

where d_η is the depth of node η, (i.e. the number of nodes above η), $\alpha < 1$ and $\beta \geq 0$.

These parameters are set beforehand in a rather ad hoc way. This splitting rule is one of the differences between the CGM approach and the DMS approach. CGM argue that it gives flexibility, but DMS say that it is unusable in practice as one rarely has much prior information about what the tree should look like. This author found it difficult to set the prior tree size using these parameters. It was necessary to set the tree size to the same as that used for runs of the DMS code, so as to make a fair comparison. A simulation had to be constructed and run in order to find the appropriate values of α and β to make trees of a given size. On the other hand, the resulting trees were found to be smaller than trees constructed with DMS software.

An interesting study (Chiogna, Spiegelhalter et al., 1996)[25] shows how domain knowledge can be used to influence tree structure. They compared trees grown by CART which were then pruned using the usual CART methods, with CART trees pruned manually by domain experts (physicians). It was found that the CART automatic method tended to remove the rarer diseases. The physicians, on the other hand, tended to use an alternating strategy in which diseases were 'peeled off' individually where possible, but where occasionally a fairly major split was made which divided groups of diseases. One of their conclusions was that the resulting trees were similar, *when the trees produced automatically came up with reasonable results.* (My italics.) Clearly, the fact that physicians can relate the training data to possible diseases is a major advantage. In Bayesian statistics we try to capture all prior knowledge in a prior distribution, so we should be looking at ways of capturing this knowledge about diseases. We do not accept the view of DMS that tree topology is unusable information, but we suggest that to make better trees will require much more complex priors than have been used hitherto. We have noted from our own work that trees of the

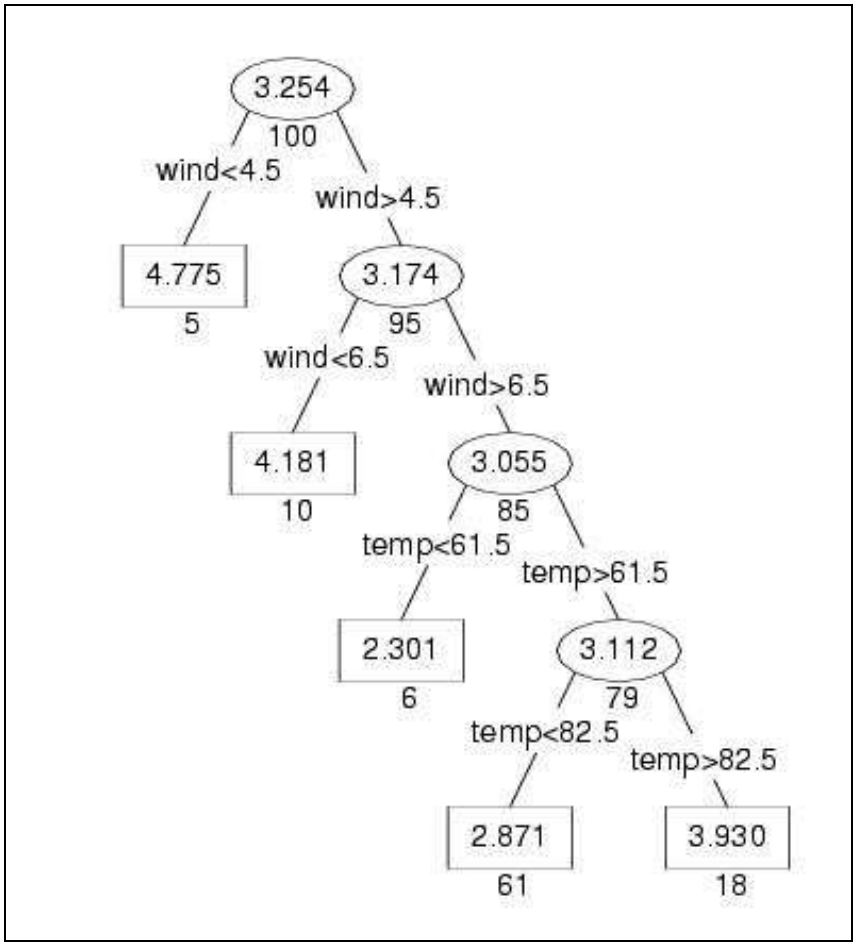

Figure 2.4: Fishbone trees

'fishbone' form occur particularly frequently. See Fig.2.4. The original tree

used by Breiman to demonstrate the value of trees is of this form. If it is the case that a particular subset of the data set is especially important to us, and that subset has typical ranges for the variables, then a fishbone tree may arise as we systematically discard observations which are not in the subset of interest. If there are multiple subsets of interest we can expect multiple fishbone structures. The tree

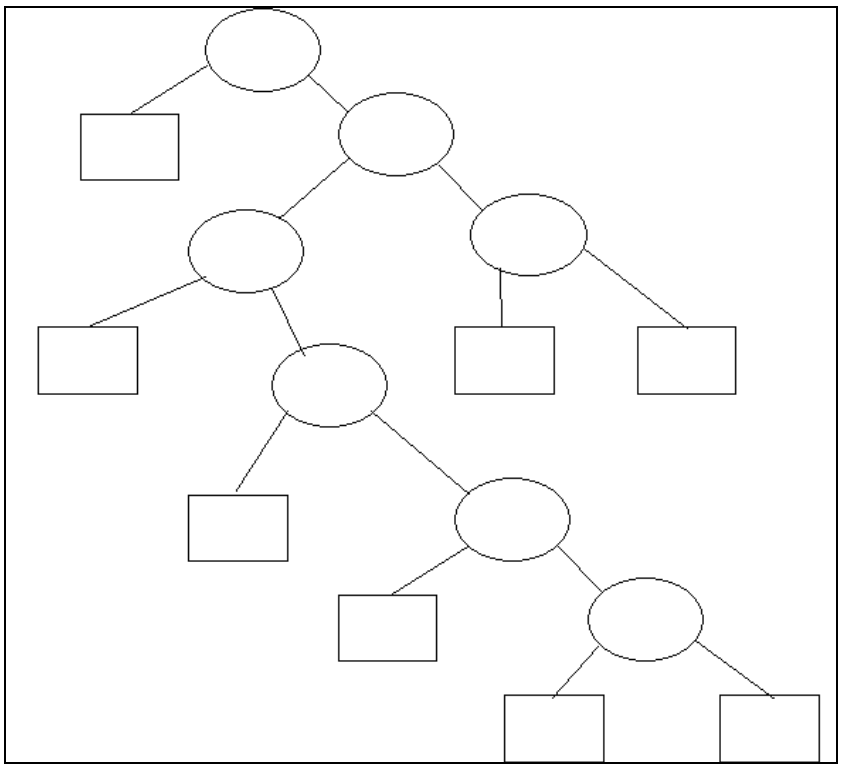

Figure 2.5: Chiogno tree

produced by Chiogno et al had the form of two fishbones. See Fig. 2.5. This structure arose because the subsets of interest corresponded to certain diseases. It might be useful to try to construct priors which 'looked' for structures of this form. How might such priors be constructed? The starting point must the CGM formula for splitting. Their splitting rule looks at depth and bushiness. It could be altered to improve the splitting probability of deep nodes, i.e. using small, or even negative, values of β.

2.4.4 Denison, Mallick and Smith's Regression Tree Model

The regression tree model of DMS treats tree models with the same number of leaf nodes as equivalent to one another: "inference is carried out by assuming that the 'true' model is unknown but comes from the class of models $M_1, M_2, \ldots$ where M_k denotes the model with exactly k terminal nodes."

They treat the topology of the tree as irrelevant whereas, as we have seen, CGM attempt to control it by parameters relating split probabilities to node depth and a 'preferred bushiness' parameter. In the DMS model the overall parameter space Θ is considered as a countable union of subspaces $\bigcup_{k=1}^{\infty} \Theta_k$, where Θ_k is the space from which model M_k is drawn. The parameter vector $\theta^{(k)}$ for trees with k leaf nodes is given by $\theta^{(k)} = \left(s_1, s_1^{\text{var}}, s_1^{\text{split}}, \ldots, s_{k-1}, s_{k-1}^{\text{var}}, s_{k-1}^{\text{split}}\right)$ where s_i is the ith node S_i^{var} is the variable being split at the ith node and s_i^{split} is the split point at which the variable is being split at node i. (DMS have a numbering system for tree nodes.) The joint distribution for $(k, \theta^{(k)}, y)$ is given by

$$p(k, \theta^{(k)}, y) = p(k|y)p(\theta^{(k)}|k, y)$$

Denote leaf node i by $T_i = (j : y_j \in t_i)$ and assume that the y's are normally distributed with unknown mean α_i and unknown mean σ_i^2. A minimum number of observations,T_{MIN}, has to be in a leaf node. The default is 5. The likelihood is thus

$$p(y|k, \theta^{(k)} \propto \prod_{i=1}^{k} \left\{ I(|T_i| > T_{MIN}) \sigma_i^{-1} \exp\left[-\frac{1}{2\sigma_i^2}\right] \left(\sum_{j \in T} (y_j - \alpha_j)^2\right) \right\}$$

where $I(\cdot)$ is the indicator function and $|T_i|$ denotes the number of observations in leaf node i. DMS use an improper uninformative prior for the α_i's and a vague gamma prior for the σ_i^2's.

$$\begin{aligned} \pi(\alpha_i) &= \text{constant} \\ \pi\left(\sigma_i^2\right) &= \text{Gamma}\left(10^{-2}, 10^{-2}\right) \end{aligned}$$

Note that these priors are independent of one another. Rather than drawing from these distribution DMS adopt a similar tactic to the one they used for classification trees: they work with the means of the sampling distributions:

$$\alpha_i = \frac{1}{n_i} \sum_{j \in T} y_j$$

The variances are updated using a Gibbs step:

$$p(\sigma_i^{-2}|y, \theta, \sigma_{-i}^{-2}) = \text{Gamma}\left\{10^{-2} + \frac{n_i}{2}, 10^{-2} + \frac{1}{2}\sum_{j \in T}(y_j - \alpha_i)^2\right\}$$

where the $-i$ subscript refers to all those elements which do not have subscript i.

CGM comment that it is unwise to use an uninformative improper prior when comparing models of different dimension, since this has been demonstrated to be problematic for the construction of Bayes factors (Spiegelhalter and Smith, 1982)[120]. Carlin and Louis [21] draw attention to a similar problem:

A good example...is given by random effects models for longitudinal data, where the dimension of the parameter space increases with the sample size. In such models, the information in the data may be insufficient to identify all the parameters, and, consequently, at least some of the prior distributions on the individual parameters must be informative. (Carlin and Louis, 2000).

This would certainly be a problem if DMS had made *all* the priors improper, but this is not what they did. The key question is whether there is enough information in the data to identify all the parameters. At the present time there is no definitive theoretical underpinning for this approach, but neither does there seem to be any empirical evidence that this issue has caused problems in practice.

A discrete uniform distribution is used to choose the variable to be split to be used at these nodes. More controversially, a uniform is also used for the split point over the available range of the variable. This is a major difference from the CGM approach, which uses a discrete distribution over the observed values. As CGM point out, their method has two advantages. Firstly, it is invariant to monotone transformations of the variable and secondly, it places more weight on regions which for which there are more observations. We agree with the comments of CGM and prefer their version on this issue. As with the classification model the prior for k the number of leaf nodes, is obtained from a Poisson distribution modified to exclude 0.

$$p(k) = \frac{\lambda^k}{(e^\lambda - 1)k!}, k = 1, 2, \ldots$$

2.4.5 Buntine's Bayesian approach

Wray Buntine applied Bayesian ideas to the classification tree problem (Buntine, 1992)[20] though, not, unfortunately, to the regression problem. Although he did not build a practical fully Bayesian tree builder, his analysis sheds light on a number of issues. Buntine imagines a model (a class probability tree, in his terms) which assigns a probability that an example y at a certain leaf is actually of class C. We assume that the sample space $\aleph$ has been partitioned into leaf nodes, so that $\aleph = \bigcup_{i=1}^{k} \aleph_i$

First, some mathematical preliminaries: Let $\alpha_0 = \sum_{i=1}^{C} \alpha_i$ and

$$B_C(\alpha_1, \ldots, \alpha_C) = A^{-(\alpha_0 - 1)} \int_{\sum_{i=1}^{C} \theta_i = A} \theta_1^{\alpha_1 - 1} \ldots \theta_C^{\alpha_C - 1} d\theta_1 \ldots d\theta_{C-1}$$

be the C-dimensional beta function

and

$I(p_1, \ldots, p_C) = \sum_{i=1}^{C} p_i \log_2 p_i$ be the Shannon information (Shannon, 1948)[118] associated with knowing which of C classes, with prior probabilities $p_1, \ldots, p_C$, an observation y belongs to.

The beta function can be expressed in terms of the gamma function (Lebedev, 1972)[90].

$$B_C(\alpha_1, \ldots, \alpha_C) = \frac{\prod_{i=1}^{C} \Gamma(\alpha_i)}{\Gamma(\alpha_0)}$$

where the gamma function is defined as $\Gamma(z) = \int_0^\infty e^{-t} t^{z-1} dt$ for $Re(z) > 0$. For large n the gamma function can be approximated by Stirling's approximation $\Gamma(n) \approx \sqrt{2\pi} e^{-n} n^{n+\frac{1}{2}}$. Using Stirling's approximation we obtain

$$\ln\left(B_C(\alpha_1, \ldots, \alpha_C)\right) \approx \alpha_0 I\left(\frac{\alpha_1}{\alpha_0}, \ldots, \frac{\alpha_C}{\alpha_0}\right) - \frac{C-1}{2} \ln(2\pi) - \frac{1}{2} \ln\left(\frac{\prod_{i=1}^{C} \alpha_i}{\alpha_0}\right)$$

Now, for a particular partition of the data P_V we associate class probabilities at each leaf $p(c_{ij} | x \in \aleph_k) = \phi_{jk}$ with j indexing the classes and k indexing the leaves. Using a multinomial likelihood and a proper but vague conjugate Dirichlet prior, the tree model can be written

$$p(P_V, \phi | \vec{x}, \vec{c}) \propto p(P_V) \prod_{k=1}^{V} \frac{B_C(n_{1k} + \alpha_k, \ldots, n_{Ck} \alpha_k)}{B_C(\alpha_k, \ldots, \alpha_k)}$$

$$p(\phi|\vec{x},\vec{c},P_V) \propto \prod_{k=1}^{V} \frac{1}{B_C(\alpha_k,\ldots,\alpha_k)} \prod_{j=1}^{C} \phi_{jk}^{n_{jk}+\alpha_k-1}$$

If each leaf contains enough examples for Stirling's approximation to apply, the log posterior for the partition P_V can be approximated as

$$-\ln(p(P_V)|\vec{x}\vec{c}) \approx \left(N + C\sum_{k=1}^{V}\alpha_k\right) I(C|P_V) - \ln p(P_V) + \sum_{k=1}^{V}\{(C-1)\Gamma(\alpha_k) - \Gamma(C-1)\alpha_k\} + \text{constant}$$

where $I(C|P_V)$ represents the expected Shannon information gain about the class due to making the partition P_V. The latter two terms are constant across all partitions, so one should choose the partition which maximises the Shannon information. Thus, Buntine obtains Quinlan's information gain criterion as a large-sample approximation to a Bayesian procedure with a Dirichlet prior. The use of Shannon information can also be viewed as an application of maximisation of subjective expected utility, with the Kullback-Leibler divergence as the utility function. See chapter 4 for more details about this. Quinlan takes maximisation of information gain as his splitting criterion without further justification. He is aware that if the number of observations at a leaf node is small then sampling error will cause problems for estimating the information gain and so he makes ad hoc adjustments to try to compensate for this. Buntine's approach is more fundamental and provides a Bayesian justification for Quinlan's information-theoretic approach. (There is however, no such justification for the information gain *ratio* that Quinlan uses in his later work, such as the C4.5 tree algorithm.) Buntine points out that, for small sample sizes, approximating the Dirichlet posterior by Shannon information is a poor approximation since Stirling's approximation is poor. One solution is to use the beta function to calculate

it accurately. So, it is not just a question of sampling error in calculating information gain: a different formula needs to be used. Buntine's work also shows how one could incorporate different costs for different types of classification error. Instead of using Shannon information, the posteriors would be used at the leaf nodes along with utilities, to carry out a decision based on maximised expected utility (see J.Q. Smith 1988,[119]) for the principles of utility theory). As we shall see in Chapter 4, Shannon information gain is an appropriate criterion only in the inference context (i.e. where there is no consideration of consequences), not the decision-making context.

Chapter 3

Stratified sampling for classification

3.1 Introduction

One area of business where decision trees are widely used is the classification of customers by financial institutions for the purpose of deciding whether or not to lend them money. Consider the following example. Assume that you are working for financial institution which lends money has two types of customers applying for loans: good and bad. Good customers will not default on a loan, but bad customers will. (We assume that operational definitions exist for existing good and bad customers, for example, we might define a bad customer as one who has had a payment outstanding for a period of at least 90 days in the past 5 years.) There is an existing process in place for assessing loan applications but it uses personal interviews and is considered very subjective. You have available to you some relevant information about the loan applicants, such as their income, sex and age. You want to construct a classification algorithm so as to classify the applicants into 'good' and 'bad' customers automatically, using the information you have as predictor variables, and to extend loans to

'good' customers but not to 'bad' ones. There are several difficulties associated with carrying out this classification in practice. The problems include

- Constructing a suitable training set
- Differential class sizes
- Differential misclassification costs
- The choice of performance measure for the classification process.

3.1.1 Training set bias

Ideally, the training set available to us would be drawn (by simple random sample, say) from the population of interest, namely customers who apply for a loan. In practice, this is not usually possible as we reject some customers as a bad risk and after we have done this we have no means of knowing whether they would have been a good customer or a bad one. Our database of past and present customers will therefore only contain customers who passed the criteria for obtaining a loan. This depleted population is likely to be very different from the population of applicants for new loans, to which we want to apply our classifier. Training our classifier on our existing database as it stands is likely to produce a classifier that performs poorly for new applicants (Meester, 2000) [98]. There is a large literature devoted to the subject of learning from flawed training sets such as these, under the name of *reject inference*. The problem of reject inference can viewed as a problem in the missing data paradigm of Little and Rubin (1987, [96]). We use the notation of, and summarise the analyses of, Feelders (2003)[45] and Zadrozny ([134]. A vector of variables (that can be numeric or categorical) $x = (x_1, \ldots, x_k)$ is observed for each applicant. This

would be drawn from their application form and would include such items as their age, sex, income and other financial details. A class label y is observed for each accepted applicant. $y \in \{0, 1\}$, where 0 denotes a good loan and 1 denotes a bad loan. The variable a denotes whether the application was successful ($a = 1$) or not ($a = 0$). y is observed if $a = 1$ but not observed if $a = 0$. Little and Rubin distinguish between a number of different missing value situations. In the Missing Completely At Random (MCAR) situation the probability that $a = 1$ does not depend on the value of y, nor on the value of x, i.e. $p(a = 1|x, y) = p(a = 1)$. MCAR corresponds to an application process in which the applicant's details are ignored and a decision is made by an independent random process such as a throw of a die. The MCAR model is not very realistic in the loan application problem, but we note that if MCAR were to apply analysis carried out using just the information from the accepted applicants would be valid. The process that determines 'missingness' is, in Little and Rubin's terminology, *ignorable*. In the Missing At Random (MAR) model a depends on x but is independent of y, conditional on x. That is, $p(a = 1|x, y) = p(a = 1|x)$ which we can write as $a \perp\!\!\!\perp y|x$. This would apply, for example, if our acceptance procedure was to reject everyone with a salary below a certain threshold and accept everyone else, where salary was one of the observed components of x. The third situation in Little and Rubin's scheme is is Missing Not At Random (MNAR), in which a, conditional upon x, still depends on y, the loan outcome. The importance of the distinction between MAR and MNAR is that in the MNAR situation the application acceptance process must explicitly be modelled, whereas in the MAR situation it is usually ignorable. (Specifically, it is ignorable if the parameters of the missingness mechanism are unrelated to those determining y, which is often the case.) However,

even in the MAR case we must take great care with our modelling approach. Some approaches work and some do not. Hand, Mannila and Smyth (2001)[64] distinguish 3 approaches to classification. Suppose we have classes $c_1, \ldots, c_p$ and a feature vector $\boldsymbol{x}$. The vector of model parameters is denoted by θ.

1. The discriminative approach. This is also known as *the diagnostic paradigm.*(Dawid, 1976)[33]. We model the decision boundaries between the different classes without attempting to model the individual class-conditional densities $f(\boldsymbol{x}|\, y_i)$. Usually this consists of finding a suitable discriminant function which optimally distinguishes between the classes. The classical example of this approach is Fisher's linear discriminant analysis. (Fisher, 1936)[47]. The classical CART algorithm is an example of this approach (though Bayesian CART methods are not, as they model the class-conditional densities). This approach produces 'hard' classifications, i.e. no uncertainty is associated along with the class predictions.

2. The regression approach. Here we model the class-conditional densities, but fail to include any prior information concerning the prevalence of the different classes, $p(y_i)$. This approach produces 'soft' classifications, i.e. for each new case that we are predicting a class for we are given a set of probabilities of it being in each of the separate classes. The most likely class is chosen, or we can combine the results with costs or utilities and use a decision analysis to make the actual classification. Logistic regression is an example of this approach, and some tree models do produce class probabilities as outputs, and so fall into this category.

3. The class-conditional approach. This is also sometimes known as *the sampling paradigm.*(Dawid, 1976 [33]). In this approach, we model the class-conditional densities $p(\boldsymbol{x}|\, y_i, \theta)$, but also use prior distributions $p(y_i)$ to model the class prevalences. We use Bayes' theorem to produce the feature-conditional class probabilities $p(y_i|\, \boldsymbol{x})$. This approach is in the spirit of Bayesian analysis, but some practitioners use maximum likelihood (ML) to estimate the components of θ. Ripley (1996 [112]) remarks that the use of ML has to do with the high regard in which it is held generally, and the fact that several pioneers in statistical classification theory have directly or implicitly recommended it. He points out that good estimates of θ do not necessarily translate into good classification performance.

It makes a difference which of the above approaches is used for the reject inference problem. Direct use of the available data (known as *complete case analysis* in Little and Rubin's terminology) is not generally valid for the MAR reject inference problem, but will work here if we use a regression-type approach, such as logistic regression. We would want to fit the model

$$p(y = 1|x) = \frac{1}{1 + \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)}$$

Because we only observe y when $a = 1$ we will in fact be fitting

$$p(y = 1|x, a = 1) = \frac{1}{1 + \exp(\beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p)}$$

However,

$$p(y = 1|x, a = 1) = p(y = 1|x, a = 0) = p(y = 1|x)$$

so we can use $p(y = 1|x, a = 1)$ to estimate $p(y = 1|x)$. If however, we choose to use a discriminative approach such as linear discriminant analysis, we would run into difficulty. From Bayes' theorem

$$P(y = 1|x) = \frac{p(y = 1)p(x|y = 1)}{p(y = 0)p(x|y = 0) + p(y = 1)p(x|y = 1)}$$

The problem here is that we cannot estimate these quantities from the accepted applicants only, since the value of a depends on x and (through x) y. In the case of tree models the splitting criterion for a node is always dependent on $p(y|t)$, where t is a test on one feature variable and $p(y|t, a = 1) \neq p(y|t)$ (Zadrozny, 2004) [134], thus tree models are susceptible to bias from the use of accepted applicant data only. If we wish to use a discriminant analysis approach we must use a model that includes the rejected applicants. One way to do this is through a mixture model. Following Feelders (Feelders, 2003, [45]) we let

$$p(x) = \pi_0 p_0(x; \theta_0) + \pi_1 p_1(x; \theta_1)$$

where the πs are the mixing proportions, the θs are the component-specific parameter vectors and the ps are the component-specific densities. If the applicant is accepted the component is observed, but not if they are rejected. There are thus two components to the likelihood function: where y_j is observed to be i the likelihood has the form $L_j = \pi_i p_i(x_j)$; where y_j is not observed it takes the mixture form $L_j = \pi_0 p_0(x_0) + \pi_1 p_1(x_1)$. For m rejected loans and n accepted loans the observed-data likelihood is

$$L_{obs}(\Psi) = \prod_{j=1}^{m} \left\{ \sum_{i=0}^{1} \pi_i p_i(x_j; \theta_i) \right\} \prod_{j=m+1}^{m+n} \left\{ \sum_{i=0}^{1} z_{ij} \pi_i p_i(x_j; \theta_i) \right\}$$

where $\Psi = (\pi\prime, \theta\prime)\prime$ is the vector of unknown parameters and z_{ij} takes the value 1 when observation j has class label i and zero otherwise. The above likelihood can

be used to produce estimates of the unknowns, either by frequentist methods such as the EM algorithm (Dempster et al. 1977,) [35] or by Bayesian methods such as the Gibbs Sampler. (See Gelman et al. (1995)[54] for examples of the use of the Gibbs sampler for mixture models.) The problems are even greater if the application process is non-ignorable. We have

$$p(y|x) = p(y|x, a = 1)p(a = 1|x) + p(y|x, a = 0)p(a = 0|x)$$

The data will tell us $p(a|x)$ and $p(y|x, a = 1)$, but not $p(y|x, a = 1)$. One approach that has been suggested is a bivariate probit model (Boyes et al. 1998 [13], Greene 1992, [60] and 1998 [61]) consisting of two equations. Again, we follow the description given by Feelders (Feelders, 2003, [45]). The first equation models the acceptance procedure as

$$a_i^* = x_i\alpha + \epsilon_i$$

while the second equation models the loan outcome

$$y_i^* = x_i\beta + \nu_i, \text{for } i =, \ldots, n$$

where a_i^* and y_i^* are unobserved numeric variables. If a_i^* is less than zero then the loan application is rejected and a_i is set to zero. If it is greater or equal to zero then the loan application is accepted and a_i is set to one. Similarly, if y_i^* is less than zero then the outcome of the loan is bad and y_i is set to zero, and if y_i^* is greater than or equal to zero the outcome of the loan is good and y_i is set to one. Of course, y_i is only observed if $a_i = 1$. The disturbances are assumed to be bivariate normal.

$$\begin{pmatrix} \epsilon_i \\ \nu_i \end{pmatrix} = N(\mu, \Sigma)$$

$$\text{with } \mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \Sigma = \begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$$

If $\rho = 0$ the two equations decouple and we have the MAR situation:

$$p(y = 1|x, a = 1) = p(y = 1|x, a = 0)$$

In a practical situation we would hope and expect that $\rho > 0$, i.e. that the accepted applicants have fewer bad loan outcomes than the rejected applicants.

$$p(y = 1|x, a = 1) > p(y = 1|x, a = 0)$$

In fact, several empirical studies (Jacobson and Roszbach, 1998 [78];Greene, 1992,[60], 1998,[61];Boyes et al. 1989 [13]) have found *negative* values of ρ. The reason for this are not entirely clear, but either the models are wrong or the banks' loan acceptance policies are making things worse! The possibility that the models are wrong should not be discounted, as they are based on some fairly strong assumptions, such as normality of the latent variables, which are difficult to verify. Banasik et al. (2003, [7]) found that the bivariate probit model provides very little improvement over models making MAR assumptions. The whole attempt to classify in the MNAR context is one that is has received critical comment. One possibility (Hand, 2002, [66]) is to give loans to a small proportion of applicants who fail the application. So if someone fails the application they might be given a loan with probability 2%, instead of with probability 0%. This soft 'accept/reject' threshold enables the lender to collect the data needed to build better models to be built, while limiting the risk to the lender company.

3.1.2 Differential misclassification costs

Clearly, the misclassification loss associated with classifying a good customer as a bad one may be very different from the cost associated with classifying a bad customer as good one. The cost differential will depend on the context. In credit card fraud situations if we classify a good customer as a bad one we may simply have to ask the cardholder to telephone us before a transaction can be authorised. This is a minor nuisance, but not a major problem. In the case of a loan application we are depriving ourselves of the potential for profit from the interest on the loan. Usually, classifying a bad customer as good is a lot more serious as we are exposing ourselves to a substantial financial risk. It is thus important to accurately reflect the cost structure of the problem. Hand and Vinciotti point out that while statistical modelling strategies typically treat the model building stage and the decision-making strategies as quite separate. One first builds an inference model and then chooses loss functions and one does not need to take the loss functions into account when building the model. Hand argues (Hand and Vinciotti, 2003 [67]) that while this is true if the model is correct, in practice models are often misspecified. One is usually trying to build a model that is 'close to' the true model in some well-defined sense, but there is some latitude in the model-building process that one can make use of to build models that perform well in the parts of the design space that are important for the problem at hand. For example, points which are well away from any decision boundaries may be downweighted in favour of points close to decision boundaries. If one is building a loan application scorecard one can choose to bring the misclassification costs in at the model building stage so as to build a model with the decision boundaries better placed.

3.1.3 Performance measures

It is not uncommon for a practitioner to train his classifier only to find that it declares all the cases to be good customers! This can happen if the classifier is designed so as to look for models which optimise overall classification accuracy. The classifier construction algorithm searches the model space for a 'good' model, and the model where all customers are 'good' is easy to find and, if the minority class is rare, this trivial model has very good overall classification accuracy.The choice of performance measure is crucial. For example, if the overall misclassification *rate* is chosen as the performance measure it be dominated by the misclassification rate of the majority class and this is undesirable. It also assumes that the misclassification costs are equal. A better measure is the total misclassification *cost*, with the correct misclassification costs incorporated (Vinciotti and Hand, 2002) [131]. Other popular methods for assessing performance are the Gini coefficient (a relative of the Gini index of node purity encountered earlier) and certain features of the ROC (receiver operating characteristic) curves (Hand, 2001a) [63]. Suppose we have a two-class problem with classes 0 and 1 and a classifier that predicts the probability of a case being in class 0 or 1 given some feature vector x. Generally, we put the case in question in class 1 if $p(1|x) > c$ for some classification threshold c. For a given classifier, we can adjust the threshold to make the classifier more or less sensitive. We define the *sensitivity* to be the proportion of customers who will go on to be a bad risk who are labelled a bad risk by the classifier. Conversely, the *specificity* is the proportion of good customers who will be labelled as a good risk by the classifier. If we adjust the classifier threshold to make it more sensitive to loan defaulters we will also find that we will classify more good people as being in the bad group. That is, as the

sensitivity has increased so has the the specificity decreased. The ROC curve plots 1 - *specificity* on the horizontal axis versus *sensitivity* on the vertical axis. It shows the trade-off between sensitivity and the specificity. An ideal classifier would have both a sensitivity and specificity of 100% and would occupy the top left corner of the unit square on the ROC curve.A good real-world classifier will stay close to the left hand vertical-axis as we move right-to-left along the horizontal axis (implying that there is little degradation in specificity as sensitivity is increased), then as it nears the top of the unit square it will hug the upper horizontal axis, implying little degradation in sensitivity as specificity is increased. A classifier that randomly assigns classes will lie somewhere along the line $y = x$. Accordingly, the area between the ROC curve and the line $y = x$ is used as a performance measure: the *Gini coefficient* is defined as twice the value of this area. It equivalent to a Mann-Whitney-Wilcoxon test of the hypothesis that a randomly chosen person from one class (say, class 0) will be more likely to be allocated to their correct class (class 0) than a randomly chosen person who is from the other class (class 1) is to be allocated to class 0 (Hand, 1997, [65].

3.1.4 Unbalanced class sizes

Good customers often greatly outnumber bad customers, so that if we take a simple random sample of cases to use as a training set then we find that in order to obtain a substantial sample of bad customers we have to take a very large sample of good customers. (In some related applications, such as fraud detection, non-fraudulent cases may out number fraudulent cases by 100 to 1 or more.)The class size issue can be seen in Bayesian terms as the need to know the prior probabilities of the classes. Even if a suitable performance measure is chosen, a major imbalance in the size of

the classes can cause problems. For example, a linear discriminant classifier uses a pooled estimate of the covariance matrix with weights determined by the class sizes. This will be dominated by the majority class, leading to suboptimal placement of the decision boundary. In principle, decision theory could be used to make use of priors and missclassification costs to set an optimal decision boundary, but many classifiers, and (as we have seen from earlier chapters) decision tree classifiers in particular, do not determine their decision boundaries according to decision theory. If a decision tree is trained on a stratified sample with 50% good and 50% bad customers it will put the decision boundary at the midpoint, without using the costs. The costs may then be involved in the classification decision, but using a decision boundary which has been set without them. For example, the following rule may be used

Rule 1: Allocate to class 1 if $p(y = 1|x) > 0.5$

Hand and Vinciotti (2002) point out that it can happen that $p(y = 1|x) << 0.5$ for all vectors x, in which case no-one is allocated to class 1. They argue that the threshold should be moved from 0.5 to k, and that to minimise the overall costs k should be set at $(1 + r)^{-1}$,

assuming:

- an error of misclassifying a class 1 person as class 0 is r and
- an error of misclassifying a class 0 person as class 1 is 1
- a correct classification incurs no cost

This leads to their Rule 2.

Rule 2: Allocate to class 1 if $p(y = 1|x) > \frac{1}{1+r}$

This can be written in different ways

Rule 3: Allocate to class 1 if $\frac{p(x|y=1)p(y=1)}{p(x|y=0)p(y=0)} = 1/r$

where the prior probabilities $(p(y=0), p(y=1))$ are estimated from the sample proportions.

A commonly used tactic for this problem is simply to train the classifier on a biased sample consisting of equal numbers of good (class 0) and bad (class 1) customers constructed by sampling of class 0. Hand and Vinciotti point out that this is a mistake: class 0 should in fact be sampled down by a factor of r. To sample down to equal numbers is to conflate the issue of misclassification costs with the issue of unequal class sizes. In the machine learning community the terminology used is *undersampling* and *oversampling*. In undersampling the majority class is reduced in size by, for example, simple random sampling (see for example, Breiman et al., 1984)[16]. A more sophisticated version of this approach selectively deletes cases which are 'redundant' or 'close to minority cases'. The redundant cases would be ones that are far from a decision boundary and thus have little impact on the results. The class 0 cases close to class 1 cases are assumed to be the result of noise rather than greater complexity in the decision surface. They can be found by looking for *Tomek links* (Tomek, 1976, [128]. Suppose we have two cases a and b that have opposite class labels, and suppose that there is no point c such that $d(a,c) < d(a,b)$ and no point such that $d(b,c) < d(a,b)$, where $d(x,y)$ denotes the distance between cases x and y. The pair (a,b) is said to be a Tomek link. A search is carried out for the Tomek links and the the majority class case is deleted. The obvious disadvantage of the undersampling approach is that some information is being thrown away. In the oversampling situation the minority class is increased in size. This can be done by duplication of cases chosen by simple random sampling or by bootstrapping (sampling

with replacement). The danger is that one is focussing greater attention of these few cases and overfitting is distinctly possible. To avoid this possibility Kubat and Matwin (Kubat and Matwin, 1997)[88] constructed pseudo-cases by interpolating new 'cases' along the line joining existing cases. Oversampling may be particularly useful where the class 1 cases are particularly few in absolute number. Sometimes the two approaches are combined. The SMOTE (Synthetic Minority Over-Representation Technique) of Chawla et al.(2000)[24] used undersampling of the majority class and creation of synthetic minority cases by interpolation along lines to nearby minority cases. However, we do not think that the problem lies solely with the data and prefer to look again at the foundations of Bayesian classification theory for inspiration.

3.2 A principled Bayesian approach

Aitchison and Dunsmore [1]give a Bayesian characterisation of the general diagnosis problem which we explain briefly. They make the following assumptions, **D1-D8**.

A *case record* or *observation* consists of a class and some feature measurements $(c, \boldsymbol{x})$. The parameter ψ is a vector of proportions, indicating the prevalence of the various types in the population. These proportions may or may not be the same as the prevalence of the classes found in the training set. ψy is known as the *arrival parameter*. We will use this parameter to incorporate our knowledge that the sample is biased with respect to the class proportions to be found in the general population. In situations where the classes occur in the sample in the same proportion as in the population we have

$$p(c_i|\,\psi) = \psi_i$$

But in situations such as the stratified classification problem where we have rigged the sample proportions, we have

$$p(c_i|\,\psi) = p(c_i)$$

i.e. the sample proportions do *not* depend on the population proportions.

D1: *Each case belongs to one and only one of a finite set* $C = \{1, \ldots, k\}$ *of possible classes.*

So, in our example there are two classes, good and bad, and each customer is in one and only one of these, though we may not know which.

D2: *For each case a finite set* $\{x_1, \ldots, x_d\}$ *of possible features may be observed, any feature vector falling within a given sample space* X.

For the cases in our training set we know the feature values and the class, but for new cases we will only know the features values. The class must be predicted.

D3. *The class of probabilistic models considered as possible descriptions of the generation of case records is indexed by a finite-dimensional vector* (ψ, θ) *with* $\psi \in \Psi$ *and* $\theta \in \Theta$.

D4. $p(c_i|\,\psi, \theta) = p(c_i|\,\psi)$ *for every* $\psi \in \Psi, \theta \in \Theta$.

The probability that a randomly chosen element of the training set has a particular class does not depend on the measured features of that element. It does, however, depend on a vector of occurrence probabilities, ψ, the arrival parameter. If we know ψ then we do not need to know θ. Each class is characterised by a probability distribution with θ as the vector of parameters. It is known as the *structural parameter.*

D5. $p(\boldsymbol{x}|\,t, \psi, \theta) = p(\boldsymbol{x}|\,\theta), \boldsymbol{x} \in X$, for every $\psi \in \Psi, \theta \in \Theta$.

The feature vector is completely defined by the parameter vector θ, and only θ.

D6. $p(z|\,\psi,\theta) = \prod_{i=1}^{n} p(z_i|\,\psi,\theta)$ *for every set* $z = (z_1,\ldots,z_n)$ *of case records.*

For given structural and arrival parameters the observations in the training set are independent.

D7. $p(\psi,\theta) = p(\psi)p(\theta), \psi \in \Psi, \theta \in \Theta$

The arrival parameter and the structural parameter are independent of one another.

D8. *For a case of unknown class* u *with feature vector* y, $p(u = c_i|\,\psi) = \psi_i,$, *for* $i = (1,\ldots,d)$ *where* ψ i*s the ith component of* ψ, *and* $\Psi = \left\{\psi : \psi_t \geq 0, \sum_{t\in T} \psi_t = 1\right\}$ *is the d-dimensional simplex.*

The arrival parameter defines the relative frequency of classes to be found in the population. These relative frequencies must add up to unity.

If we denote the set of possible classes $\{c_1,\ldots,c_k\}$ by C and the set of feature vectors by $\boldsymbol{x}$, then Aitchison and Dunsmore's general result is that the posterior for a new case with feature vector y and unknown class u is given by

$$p(\psi,\theta,u = c_i|\,y,z) = \frac{p(\psi|\,c_i)p(u = c_i|\,\psi)p(\theta|\,z)p(y|\,u = c_i,\theta)}{\sum_{i=1}^{d} p(c_i)p(y|\,u = c_i,z)} \tag{3.2.1}$$

This is a very general result which allows to simultaneously predict the class of the unknown case u, and sequentially update our estimates of the parameter vector θ and the arrival parameter vector ψ (i.e. the class prevalence rates) after each new observation. The ability to cope with class prevalence rates that are changing with time could be useful in situations where management activity is actually causing the arrival rates to change. For example, in fraud detection it may be that the level of fraud is being driven downward by improved business practices. Over longer timescales, it may be that the fraud rates run in cycles, whereby it is first driven

downward as management processes win out, then fraudsters develop a new strategy which is at first successful, and this is then countered by more management changes. This Bayesian approach has other virtues. For one thing, it solves the problem of what to do to start up the method once it is installed, but customer data is not yet available. A vague prior could be used for the arrival parameters (a Dirichlet, for example) and the system could learn the values as it goes along. If the data on the majority class are essentially cost-free (no acquisition costs, no processing costs and no other costs) it would be possible to use all of this data to form estimates of θ that are as accurate as possible, on the Bayesian principle that any (relevant) free information should be included. In practice, we think that the information is unlikely to be free, but even then decision theoretic principles similar to those in Chapter 4 are relevant to determining how much data is worth including. In practice, evaluation of Equation 3.2.1 could be a formidable computational exercise in many situations, by virtue of the intractable integrals involved in computing the predictive densities (see below). We are optimistic that this approach is now computationally feasible or will be in the near future. Particle filter (also known as Sequential Monte Carlo) approaches (Doucet et al., 2001) [40] have in recent years been applied to a number of sequential decision problems, some of which, e.g. radar tracking problems, require results in real time (Fearnhead, 1998)[44]. In particular, Amzal et al. (2003)[3] have successfully applied particle filter methods to complex optimal design problems. If, as in this case, we are interested simply in predicting the class of an example on the basis of its measured characteristics y, and a set of past examples z, then this simplifies to

$$p(u = c_i|\, y, z) = \frac{p(u = c_i)p(y|\, u = c_i, z)}{\sum_{i=1}^{d} p(u = c_i)p(y|\, u = c_i, z)} \tag{3.2.2}$$

The distribution $p(y|\, u, z)$ is the *posterior predictive* distribution i.e. the posterior with the model parameters integrated out. In the case where the class proportions in the training sample have been rigged, as here, we need to specify these class proportions values for $p(c_i)$ explicitly. This approach takes into account the different sample sizes of the minority and majority classes. This is not to say that it is unaffected by differential class sizes. The smaller sample size associated with the minority class is still likely to result in worse classification accuracy for that class.

3.2.1 Example with a normal covariate

Suppose that the good and bad customer groups are both characterised by a normal feature vector, perhaps after an appropriate transformation. For example, in a one-dimensional case with income being used as the sole feature, it may that the logarithm of income is approximately normally distributed. We predict whether a new customer will default by making use of knowledge of their features. Let the income for the good customers (on some appropriate scale) be distributed $N(\mu_1, \sigma_1^2)$ and as $N(\mu_2, \sigma_2^2)$ for the bad customers, with sample mean vectors $\overline{x}_1, \overline{x}_2$ and sample variances s_1^2, s_2^2. Let there be a training set with n_1 good and n_2 bad customers. Then $m_1 = \overline{x}_1, m_2 = \overline{x}_2$ are distributed as $N(\mu_1, \frac{s_1^2}{n_1})$, $N(\mu_1, \frac{s_1^2}{n_1})$ and $\nu_1 = \frac{n_1 s_1^2}{\sigma_1^2}, \nu_2 = \frac{n_2 s_2^2}{\sigma_2^2}$ are distributed as χ^2_{n-1}. If we use the conjugate priors

$$\begin{aligned} 1/\sigma^2 &\sim \chi^2(\nu_0, \sigma_0^2) \\ \mu|\, \sigma^2 &\sim N\left(\mu_0, \frac{\sigma^2}{\kappa_0}\right) \end{aligned}$$

the joint prior density is given by

$$p(\mu, \sigma^2) \propto \sigma^{-1}(\sigma^2)^{-(\nu_0/2+1)} \exp\left(-\frac{1}{2\sigma^2}\left[\nu_0\sigma_0^2 + \kappa_0(\mu_0 - \mu)^2\right]\right)$$

Multiplying the prior by the normal likelihood gives the joint posterior:

$$p(\mu, \sigma^2 | \, x) \propto \sigma^{-1}(\sigma^2)^{-(\nu_0/2+1)} \exp\left(-\frac{1}{2\sigma^2}\left[\nu_0\sigma_0^2 + \kappa_0(\mu_0 - \mu)^2\right]\right) \times$$

$$(\sigma^2)^{-n/2} \exp\left(-\frac{1}{2\sigma^2}\left[(n-1)s^2 + n(\overline{x} - x)^2\right]\right)$$

This is known as the Normal-Inverse-Chi-Squared and is denoted by

$$p(\mu, \sigma^2 | \, x) = \text{N-Inv-}\chi^2\left(\mu_n, \sigma_n^2/\kappa_n; \nu_n, \sigma_n^2\right)$$

where

$$\begin{aligned}
\mu_n &= \frac{\kappa_0}{\kappa_0 + n}\mu_0 + \frac{n}{\kappa_0 + n}\overline{x} \\
\kappa_n &= \kappa_0 + n \\
\nu_n &= \nu_0 + n \\
\nu_n\sigma_n^2 &= \nu_0\sigma_0^2 + (n-1)s^2 + \frac{\kappa_0 n}{\kappa_0 + n}(\overline{x} - \mu_0)^2
\end{aligned}$$

The resulting posteriors for μ and σ^2 are given by

$$\begin{aligned}
\mu | \, x &\sim N\left(\mu_n, \sigma^2/\kappa_n\right) \\
\sigma^2 | \, x &\sim \text{Inv-}\chi^2(\nu_n, \sigma_n^2)
\end{aligned}$$

However, what we need is the posterior predictive. Let $\tilde{x}$ be a new observation. Then the posterior predictive is given by

$$p(\tilde{x} | \, x) = \iint p(\tilde{x} | \, \mu, \sigma^2) p(\mu, \sigma^2 | \, x) d\mu d\sigma^2$$

Analytically, this can be shown to be a t distribution. (It can also be evaluated numerically by first drawing from $p(\mu, \sigma^2)$ and using that as input to $p(\tilde{x}|\,\mu, \sigma^2)$ and then integrating.) For the 'good' class it is a t distribution with $n_1 - 1$ degrees of freedom, location $\overline{x}_1$ and scale $\left(1 + \frac{1}{n_1}\right) \frac{\nu_0}{n_1 - 1}$. For the 'good' class it is a t distribution with $n_2 - 1$ degrees of freedom, location $\overline{x}_2$ and scale $\left(1 + \frac{1}{n_2}\right) \frac{\nu_0}{n_2 - 1}$. We want $p(\text{Good}|\,y, z)$ and $p(\text{Bad}|\,y, z)$, which are given by $p(\text{Good}|\,y, z) \propto p(\text{Good})p(y|\,\text{Good}, z)$ and $p(\text{Bad}|\,y, z) \propto p(\text{Bad})p(y|\,\text{Bad}, z)$. These results can be extended to the multivariate case (Aitchison and Dunsmore, 1975)[1].

3.2.2 Incorporating data acquisition costs

Costs and benefits can also be incorporated into the above analysis. For a good customer there will be a positive revenue stream associated with interest payments, on the loan. For the bad customer there might be a sum associated with bad debt which has to be written off. The number of minority class cases is limited by availability (assuming we do not create any synthetic cases), but the number of majority class cases used is under the control of the designer. If there really are no costs or penalties of any kind associated with the use of additional majority class cases then in principle all of them should be used. In practice, it is most unlikely that the use of additional majority class cases is free. It may also be that further cases of the minority class could be made available, but at a high cost, and it is this cost element, rather than lack of availability, that limits the size of the minority sample. In that case we have a decision theory problem to solve in order to find the number of majority and minority class cases which maximises revenue or minimises costs. A general framework for such Bayesian calculations is given by Raiffa and Schlaifer (Raiffa and Schlaifer, 1961,[110])

which they refer to as *preposterior* analysis. We have

1. a set of possible terminal actions $A = a$. In this case the terminal action is to extend a loan to a customer or to refuse the loan request.

2. a set of unknown states of nature $\theta = \theta$. These would be the parameters of the feature distributions for each class, assumed unknown and estimated sequentially as cases are gathered.

3. a set of possible experiments $E = e$ we could carry out. In our case we can classify on the basis of n_1 Good examples and n_2 Bad examples, for any $n_1, n_2 \in 1, 2, \ldots$

4. a set of possible outcomes of our experiments, the *sample space* $Z = z$. This will be the classification accuracy achieved with particular values for n_1 and n_2.

5. a set of loss evaluations $L(\cdot, \cdot, \cdot, \cdot)$ on $E \times Z \times A \times \Theta$. The loss function is intended to model, as accurately as is practical, the disbenefits associated with classification errors. For, example, we may have average cost data that we can use to determine the likely cost of an error each way.

We have to compute the expected loss $\mathbb{E}\left(L(n_1, n_2)\right)$ associated with any pair of values of n_1 and n_2. The procedure would be

1. to create priors for the parameters of the class-specific feature distributions. These should reflect what is known beforehand, if anything. If nothing is known then some form of noninformative prior (either proper or improper) should be chosen.

2. Initial values n_{10} and n_{20} should be chosen for n_1 (the majority class) and n_2 (the minority class). If the minority class is limited by availability then it seems reasonable to use all of it and an equal number of the majority class.

3. The expected classification accuracy for the two classes is then computed, on the basis of the chosen priors and the chosen values for n_1 and n_2.

4. The losses associated with the expected classification accuracy are computed.

5. The space of feasible values of n_1 and n_2 is searched to find the pair associated with the lowest expected loss.

6. This number of cases is chosen and used to train the classifier.

7. The posterior values of the Θ values is computed via 3.2.2

8. The revised classifier is used to classify any new cases that have arisen (thus new cases are always classified by the best classifier available).

9. It is now necessary to decide how many additional cases (if any) should be added to n_1 and n_2.

10. Steps 1 to 9 are iterated regularly, or as new cases arrive.

Once again, this is likely to be a formidable computational exercise, and it remains to be seen whether it is computable in the near future. However, the general nature of the optimal solution can be seen now.

3.2.3 Nonparametric approaches

Traditional statistical models were parametric, but the widespread availability of powerful computing facilities towards the end of the 20^{th} century has allowed nonparametric modelling to flourish, though Bayesian approaches were slower to emerge due to the need to establish a suitable theoretical framework and associated computing strategies. With nonparametric approaches Equation 3.2.1 still applies, but now there is the additional computational penalty associated with estimated the model structure from the data. As far as existing, somewhat imperfect nonparametric methods such as decision trees are concerned, it is clear that approaches that fail to model the class prevalences are going to perform poorly in the face of data with significantly unbalanced class prevalences. Adjusting the training set will not fix the root of the problem. We need to use a model selection process that takes account of class prevalences. Breiman et al.'s Gini coefficient is incorrect in the face of a non-representative training set and needs to be modified. The CART algorithm does allow the user to specify prior class probabilities and this does make a difference. On the one of their data sets (mass spectra) the non-chlorine class outnumbers the chlorine class 10:1. A tree which navely assigns all the data to the non-chlorine class would have a 10% misclassification rate. Correct assignment of the priors changes this to about '5%', with a 3% misclassification rate for the non-chlorine class and 30% for the chlorine class. Assigning equal priors tends to equalise the misclassification rates in general. For this data set, equal priors give a 9% misclassification rate for chlorine and 7% for non-chlorine. The user is advised to try out different priors to find the pair of misclassification rates most acceptable to them. Breiman et al. also have a proposal for implementation of costs in the algorithm. The Gini coefficient of node impurity

at node t is replaced by assuming that we classify a case as class j with estimated probability $p(j|\,t)$. The estimated expected cost is then given by

$$\sum_{i,j} c(i|\,j)p(i|\,t)p(j|\,t) \tag{3.2.3}$$

where is the cost of misclassifying a case from class j as one of class i. This is used instead of the Gini index. Unfortunately, there is a catch. Equation 3.2.3 is symmetric between classes i and j and thus does not deal properly with asymmetric costs. Breiman et al. recommend rigging the priors to achieve the same affect as differential costs. Quinlan's C4.5 program was developed after CART, but is less sophisticated in some ways. It does not have a means of incorporating priors into the information gain ratio criterion, nor does it have costs associated with different misclassifications. We recall that Quinlan first derives the information associated with classifying a case from a set S of cases as belonging to class C_j. He asserts that this has probability $\frac{\text{freq}C_j,S}{|S|}$, where $|S|$ is the number of cases in S. If the training set does not reflect the population this is no longer the case and the subsequent derivation of the information gain (and gain ratio) is wrong. The relevant probability is exogenous to the data and has to be put in as a prior. As pointed out in Chapter 4, use of an information gain model selection criterion in problems where no (or equal) costs are present can be justified on the basis of minimising expected loss, as the appropriate loss function is a logarithmic score function and this leads to the choice of model which maximises the Kullback-Leibler divergence, which is equivalent to the gain in Shannon information. Where costs are involved, this is simply the wrong loss function and a more appropriate one needs to be substituted. The CHAID approach makes no provision for imbalanced samples.

Chapter 4

Issues with Tree Models

4.1 The statistical strategy used in CHAID

Kass [84] introduced the idea of choosing splits on the basis of the 'most statistically significant' of the available next splits. Kass also suggested that the stopping criterion should be that one stops when there are no new 'statistically significant' splits available to make. This is a poor statistical strategy for a number of reasons.

1. First of all, the notion of choosing the 'most significant' of several 'significant statistics' is one that many statisticians would regard as highly dubious. Significance tests were designed to distinguish results which could be spurious, due to sampling error, from those that are unlikely to be caused by sampling error. The 'most significant' split point is not necessarily the best in terms of predictive power: it is simply the one with the least chance of spuriously separating observations because of sampling error. In practice there may be little difference between the significance levels of different splits, as measured by, say a p-value.

2. There are also a number of implementation difficulties with the approach. One first needs to choose a suitable test statistic to work with, and, since one wants

the most significant split amongst several, one needs to calculate the distribution of the maximum of this statistic, i.e. the last order statistic, which is not usually straightforward. The calculations used by Kass involve asymptotic normal assumptions that may not be valid for small sample sizes, or where the distribution over the various predictor categories is skewed.

3. The use of large numbers of classical significance tests, one after the other, makes it almost certain that spurious 'significant' results will be found unless preventive measures are taken. Kass advocated the use of the Bonferroni correction to the test size α. However, the Bonferroni procedure assumes independence between the results, which makes it conservative. It is only a first order approximation and it becomes worse (i.e. very conservative) when correcting for large number of tests. In CHAID the number of tests can run into hundreds of thousands for large data sets.

4.2 Consistency of tree models

The CART book by Breiman et al.[16] discusses the issue of consistent estimation of tree models and includes some proofs. This is an important question. Classical statistical models usually have a fixed number of parameters and are estimated by methods that are provably consistent, such as maximum likelihood. With tree methods the recursive partitioning generates a number of non-overlapping subsets, in which the data can be characterised by a mean and a variance if the response is numeric, or a multinomial if the response is categorical, and so the 'model' is the set of all such mean and variance parameters. Crucially, the number of such parameters is determined by

the data. The issue of whether a model with a data-dependent number of parameters can be consistently estimated was considered by Neyman and Scott (1948)[103], who showed that the usually well-behaved estimation method of maximum likelihood can generate inconsistent estimates when the number of nuisance parameters increases with the sample size. Breiman et al. (1984)[16] gives existence proofs that there is a partition of the data that provides a partition model which is arbitrarily close to the true regression or classification model. It does not show us how we can find such a partition, nor does it tell us how good current tree algorithms are at finding partitions close to the true model. Lee (1998)[91] provides a Bayesian proof of the consistency of artificial neural network models in which the number of hidden nodes is allowed to grow with the size of the training sample. This is a somewhat similar situation to the problem of regression tree consistency, although neural network models are continuous whereas regression trees are not. Nevertheless, it may be that his proof can be extended to the recursive partitioning situation. Note that in some cases a tree model can exactly represent the true regression or classification function, but in other cases it cannot. Tree models divide the design space into box-shaped regions lined up with the coordinate axes. If the true regression or classification function also takes constant values on box-shaped regions which are lined up with the coordinate axes then it can be represented exactly by the tree model. In such a situation, as the sample size is increased, a good tree model should approximate the true regression function well. The number of model parameters needed to define the model is finite, and if the model generation process stops adding more parameters, as it should, then increasing the sample size should provide additional precision. In this situation, consistency should be an achievable goal for a tree algorithm, though it is not clear

that current tree models do in fact achieve it. In particular, the CHAID algorithm might be expected to generate spurious splits when large sample sizes are used, simply because classical significance testing with very large sample sizes tends to reject the null hypothesis whether it is true or not. A more complicated situation is when the true regression or classification function is smooth, or takes constant values on regions with curved boundaries or straight boundaries not parallel to the coordinate axes. If this is the case, then the true regression or classification function cannot be represented exactly by the tree model. As the sample size is increased, finer and finer partitions will be created in order to achieve fidelity to the data and the number of model parameters will increase without bound. For consistency we certainly want the number of parameters to increase more slowly than the number of observations. For example, Lee's proof of the consistency of neural networks assumes that the number of hidden nodes does not increase faster than n^{α} , where n is the number of observations and some $0 < \alpha \leq 1$. Oates and Jensen (1997) [104] provide empirical evidence that with trees grown with C4.5 the tree size increases linearly with the training set size.

4.3 The statistical strategy used in C4.5

The idea of using of information theory to decide upon splits seems a good one, but whereas the information gain criterion seems reasonable the gain ratio seems arbitrary. There seems to have been no attempt to analyse the failure of the gain criterion and correct it. The gain criterion has simply been exchanged for the gain ratio, which has no clear justification. The problem with the gain criterion is simply that it contains no penalty for model complexity. This is addressed by the subsequent

pruning, but this is a less than ideal approach. In fact, a number of model selection criteria have been proposed which have an information theoretic measure of fit plus a model complexity penalty and they have some theoretical justification. The Akaike Information Criterion, (Akaike 1974)[2] or AIC, is the most well known. It given by

$$\text{AIC} = \text{deviance} + q$$

Where the deviance is $-2\times$ (log maximised likelihood) and q is the number of model parameters. The lower the value of AIC the better the model. Akaike's argument is that, given the data and a particular set of models with their parameters set to specific values, the AIC criterion will pick the model which will best estimate the predictive distribution of a new observation. Kass and Raftery (1995)[85] argue that this ignores model uncertainty, and so that AIC will tend to overfit. Schwarz's Bayesian Criterion, BIC, (Schwarz, 1978)[117] is similar but imposes a larger penalty on the number of parameters: it uses $\ln(N)$ instead of 2 for the coefficient of q. BIC is an approximation to the logarithm of a Bayes Factor where a noninformative prior has been used (Kass and Raftery, 1995)[85]. Hastie et al. [123] point out that when comparing models from a model class that contains the true model, BIC converges on the true model as the number of data points tends to infinity whereas AIC does not. On the other hand, the higher model complexity penalty of BIC may be inappropriate in small samples. Accordingly, they recommend the use of AIC when data is scarce and BIC when data are plentiful. Note that AIC and BIC are both relative measures, comparing one model against another, rather than measuring the goodness of fit of a single model in isolation. As far as pruning is concerned, Quinlan mentions the assumption of a binomial distribution at the leaf as potential problem, but this is perhaps the least troublesome issue with the method. The error rate is based on

a resubstitution estimate and thus biased, perhaps greatly so. The use of the 75% upper confidence limit to correct for this is arbitrary. Furthermore, both CART and C4.5 use the expected error rate as a performance measure, and this assumes that errors of different types (making too big a tree versus making too small a tree, have equal losses.) Buntine showed that the information gain approach can be justified by Bayesian arguments, at least when the sample size is sufficiently large. Unlike CHAID and CART, C4.5 did not originate in the statistical community, it came from machine learning, a rapidly growing branch of computer science. There are strong cultural differences between the two communities and this leads to quite different approaches to the same problems. The machine learning community has tended to focus on the great storage capacity of the modern computer and has devoted much attention to the idea of 'scalable' algorithms in the which the computation time is a slowly increasing function (ideally linear) of some measure of the problem size. Statisticians, on the other hand, have tended to focus on the great computational power of the modern CPU and have devised computationally intensive techniques to estimate models that are more realistic than their predecessors. Such models, of course, are not suitable for use on very large data sets. C4.5 seems to be a program designed for good run time performance, but which cuts a number of statistical corners in order to achieve this. The pruning method, in particular, is very lightweight by comparison with CART's cross-validation approach. In view of Breiman's work, which established that pruning is a critical issue, this seems to be serious deficiency. However, none of these problems seem to have seem to inhibited the use of C4.5 in the computer science community.

4.4 Choice of coordinate axes

If we have a conventional statistical model such as a multiple regression, we can have, say, two predictor variables. Generally, we can do this in more than one way. We could put in variables X_1 and X_2 or we could put in $X_1 + X_2$ as one variable and $X_1 - X_2$ as another. It makes no difference to the regression model. (Nor should it, the information content is the same.) However, with a tree model it makes a big difference. We can illustrate this with one of the classic data sets of multivariate analysis. Edgar Anderson [5] collected length and width measurements for the petals and sepals of 150 irises found in Canada's Gaspe peninsula. The irises belonged to three different species: *Iris Setosa, I. Versicolor* and *I. Virginica*. Anderson's data was used a year later by Fisher (Fisher, 1936)[47] to illustrate methods of multivariate analysis. A plot of petal length versus petal width (in millimetres) is shown in Fig. 4.1 If we apply a CART classifier to this data we obtain the classification shown in Fig.4.2. We see that the coordinate axes used by the classifier are not lined up with the data. If we carry out a principal components analysis (PCA) on these data we find that the first principal component accounts for 98% of the variation. The data are thus virtually one dimensional. We plot the values of the first principal component, PC1, in Figure 4.3. To make it a bit clearer we have plotted it against a variable which takes the value 0 in all cases, and some jitter has been added. We then used the CART algorithm to predict species using PC1 and PC2, the first and second principal components, as predictors. The resulting tree is shown in Figure 4.4. Notice that the second principal component is not used at all by the tree. (Notice also that there are two spurious bottom-level splits which could be removed without making any difference.) The data can now be split using splits which are purely vertical and

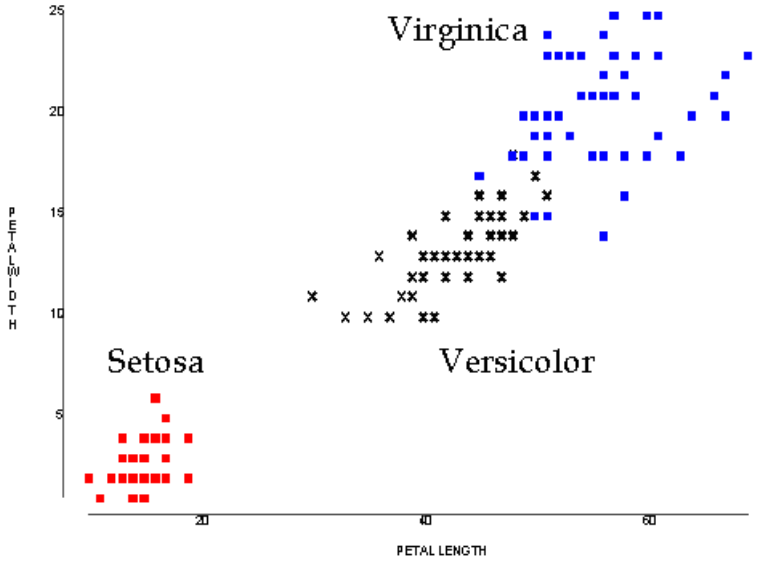

Figure 4.1: Petal length and width for 150 irises.

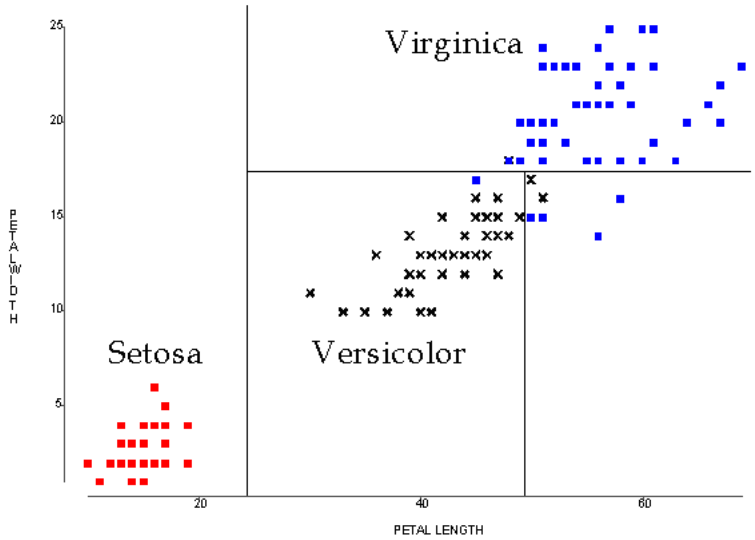

Figure 4.2: A C&RT classification of the iris data.

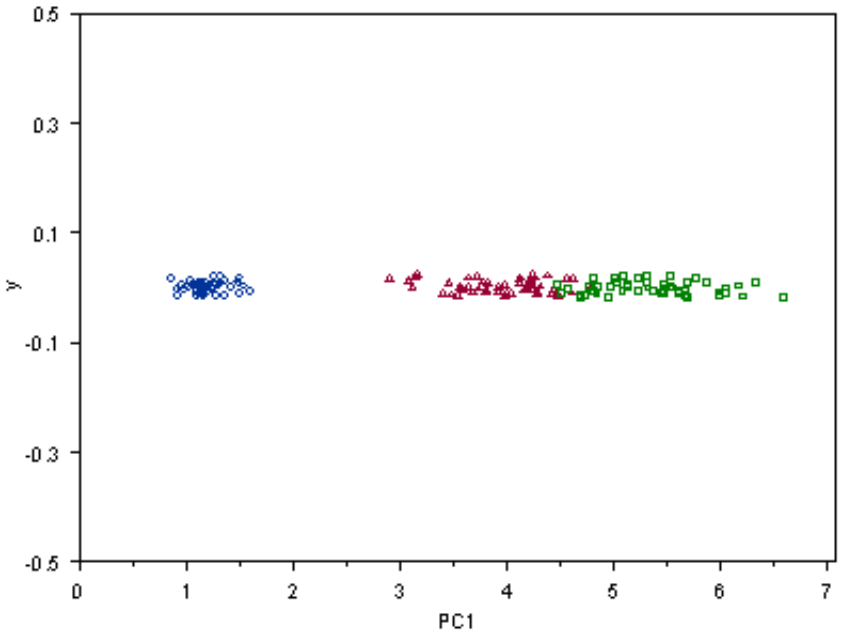

Figure 4.3: The first principal component of the Iris data. (Jitter has been added.)

horizontal and vertical (in this case, purely vertical).

4.5 Representation of categorical predictors

A categorical predictor variable is usually put in directly as a variable containing the category values. This is different to the way that categorical variables are put into conventional statistical models (linear models, generalised linear models etc.) In conventional statistical models a set of $k-1$ dummy variables is used, where k is the number of levels of the categorical variable. Each dummy variable is binary, taking a 1 where the categorical variable takes a specific level and zero otherwise. One of the levels is deemed to be a reference level and does not appear. The dummy variable approach can also be used for trees and the tree that results from this process

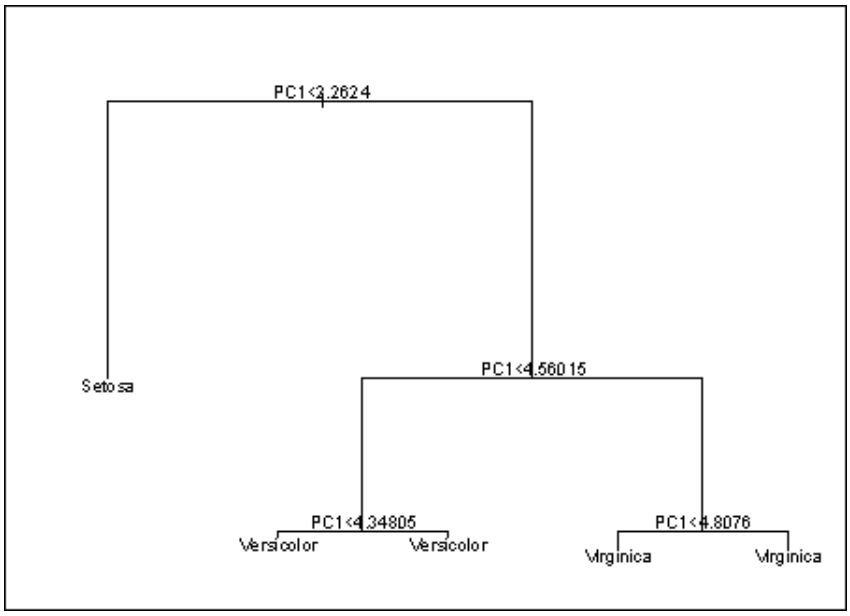

Figure 4.4: Tree Prediction of species using principal components.

is in general different from the tree that results from using the categorical variable directly. In a marketing context it is not unusual to have tens of categorical predictor variables, each with tens of levels. For example, a marketing analyst working for a company selling telephone directories will want to predict whether a particular business, with known characteristics, will purchase space in the directory. Much may be known about the company concerned, especially if they have been a customer in the past, and much of this information will be categorical data. The use of logistic regression frequently fails because the number of dummy variables created may exceed the available computer memory. In the same circumstances a tree model may be capable of being run. However, this discrepancy between the two ways of representing the same data does not really make sense.

4.6 Parsimony of parameters

Ideally, we would like our models to have few parameters but good explanatory power. Classical tree models such as tree models have many parameters and are prone to overfitting. The pruning mechanism does reduce the number of parameters, but this may not be the best way of doing achieving parsimony. An alternative approach is shrinkage, i.e. the use of correlated parameters, which reduces the effective number of parameters. Intuitively, we are building in the reasonable idea that close areas of the predictor space should have similar parameter values. Shrinkage has been tried for classical trees (Hastie and Pregibon, 1990)[68] and Chipman, George and McCulloch's original Bayesian CART paper (Chipman, George, McCulloch, 1998)[27] suggested a hierarchical prior. The Bayesian approach to shrinkage involves the use of hierarchical models (Gelman et al, 1995)[54] and is justified by de Finetti's Representation theorem (O'Hagan, 1988)[105], which says that exchangeable random variables can be considered as draws from a higher level random variable. Classical tree models, despite their hierarchical construction algorithms, are not hierarchical models in this sense, as all the leaf nodes are equivalent. In our view, the incorporation of shrinkage in models with large number of parameters, such as tree models is highly desirable. From the Bayesian point of view both pruning and shrinking are encompassed in unified framework, since they just correspond to different choices of prior, to be used as appropriate in context.

4.7 Model selection, model averaging and stability

Much of the tree literature is concerned with selecting the *best* tree. This author has more than once encountered users of tree models in a marketing department who mistakenly believe that the tree which emerges from their tree algorithm is the 'true' or 'best' tree. In fact, the tree spaces which are typically generated in practical applications such as marketing are so large that at the present moment there is no way of examining more than a tiny fraction of the possible trees. Even if there were to be a best tree according to some criterion, it is likely that are many trees that are almost as good. There are a few issues to be teased out here. First of all, do we have to have a single model as our result? In practical work model uncertainty is usually a major issue (Chatfield, 1995)[23] (Draper, 1995)[42]. If there is no requirement that a single model is used for prediction then we may be better off to take model uncertainty into account and use a model averaging approach. Bernardo and Smith (1997)[10] classify model selection situations into three categories, and they argue that what we should do depends on which of these situations we are in. The three situations are:

1. $\mathcal{M}$-closed

   We have a finite set $\{M_1, M_2, \ldots, M_N\}$ of models. We believe that one of them is the true model but we do not know which one.

2. $\mathcal{M}$-completed

   We have a finite set of models but we do not believe any of them is the true model. That is, we believe that a true model exists but we do not wish to use it. We wish to find the member of the model class which is, in some sense, the

closest to it. We might take this approach if the true model is very complex or computationally intractable compared to the models in our model class.

3. $\mathcal{M}$-open

   We have a finite set of models but we do not believe there is a true model. We are simply interested in finding a model that performs well by empirical criteria: it should fit the training data well and predict well on new data. Our model class is chosen to be sufficiently rich that we have a good chance of finding such a model.

In the $\mathcal{M}$-closed case we may have reasons for thinking that some members of the model space are more likely to be the true model than others. We express this information in the form of prior probabilities. Where we have no information about which model is more probable we give them equal probability. Thus we can use the data to revise these probabilities to form posterior probabilities, which then reflect both our original ideas about the relative chances of each of our models being true given the data we have observed. Let $p(M_i)$ be the posterior probability of model M_i and let $p(\boldsymbol{x}|M_i)$ be the joint probability density function for the data under model M_i. Then we can use the marginal distribution $p(\boldsymbol{x}) = \sum_{i=1}^{N} p(M_i)p(\boldsymbol{x}|M_i)$ as an overall mixture distribution which now incorporates everything we know. In particular, in the case where the models have one variable singled out as a response (dependent) variable, our predicted response from this mixture model is the expected value of the responses from the component models. For tree models, the model that results from this model averaging process is not itself a tree. This is regarded by some as a flaw in the approach by some, who claim that trees are relatively easy to interpret in terms of domain knowledge compared to other model types. However, we disagree with this

view. We do not regard trees as especially easy to understand in domain-knowledge terms unless the trees are very small. In the $\mathcal{M}$-completed and $\mathcal{M}$-open situations we do not believe the true model is in our model class, therefore use of the above model averaging formula is philosophically problematic if probabilities are to be understood as beliefs: all the probabilities should be zero. However, we proceed on the pragmatic basis that the $p(M_i)$ terms are now simply weights which 'reward' a model for relative credibility in terms of the training data. In practice, trees are often used in the $\mathcal{M}$-open situation with no domain knowledge to guide us towards good models: we have to use purely empirical criteria. There are a number of selection methods, both Bayesian and classical and many of the classical methods for model selection have some Bayesian interpretation or basis. For example, cross-validation can be seen as approximate use of the posterior predictive distribution. Most of the classical methods use the deviance, which is clearly the relevant goodness of fit measure, though they may differ in how they penalise model complexity. Model averaging also effectively solves the stability problem for trees, though at a computational cost. This is of great value when the classifier has to convince non-specialists, as large changes in the predicted outputs are particularly damaging to the credibility of the predictive system. A major problem in practice is that the model space is very large and it is not practical to average over *all* the models. Some means of finding a tractable subset to use is needed. There are workable approaches, see for example, Hoeting et al. (1999)[74]. There are several non-Bayesian methods for model averaging, two of the most popular being *bagging* and *boosting*.

- Bagging is a technique invented by Breiman (1995)[14] to improve the performance of tree models and other unstable predictors. We construct a bootstrap

sample as follows. Suppose we have a data set S consisting of N rows. We draw a sample of size N by repeatedly simple random sampling rows from S with replacement until we have N rows in our sample. Call this sample B_1. We construct a large number of other bootstrap samples $B_2, \ldots, B_M$. Values of M somewhere between 200 and 1000 are common. Each bootstrap sample is used to construct a model and the model results are aggregated: in the case of regression models the M model outputs are averaged, in the case of classification models some form of voting is used. One of the potential problems with bagging is that all the models are given equal weight: many of the models may be very similar and thus be effectively over-represented in the averaging process. However, this seems to be more a problem in theory than in practice as bagging works well. The theoretical basis of bagging involves the idea of a bias-variance trade-off. The concept of variance is directly appropriate to regression models and but can be extended in various ways to cover classification models (Friedman, 1996,[51] Kohavi and Wolpert, 1996[86], Tibshirani, 1996[125], James and Hastie, 1997)[79]. Tree models have low bias and high variance and the averaging process involved in bagging reduces the variance. The resulting predictive accuracy can be very good, and comparable with the best predictive methods, such as neural networks. Of course, the resulting model is not a tree and the ease of interpretability of the model has been lost.

- Boosting (Freund and Schapire, 1997)[48] is another ensemble technique which has been used particularly on classification problems. It works by first of all drawing a bootstrap sample using simple random sampling with replacement from the original data set. Simple random sampling gives every row of the

data an equal chance of being in the bootstrap sample. One then constructs a classifier on this bootstrap sample using a weak learner such as a decision tree. The rows which are misclassified by this model are noted and a second bootstrap sample is drawn, but with an increased selection probability for the misclassified rows and with a lower selection probability for the rows that were classified correctly. A new model is constructed on this new sample. We keep doing this until the error rate on the training set is driven to zero. (This does not, of course, mean that the out-of-sample prediction error is zero.) It was originally thought that boosting works in a similar way to bagging, but achieves better results by forcing the model construction algorithm to pay a lot of attention to the hard-to-classify data and building sufficient structure to encompass. The final classifier is a linear combination of the classifiers produced so far. Boosting has been the subject of a lot of subsequent work and it is now known that boosting is in fact quite different to bagging, as it reduces both the variance *and the bias* of the base classifier. The bootstrapping component can be replaced by a weighting of each observation, thus removing the randomisation component of boosting. It also works well when the trees involves are merely *stumps* (simple trees with just two leaves), which is something bagging cannot do. A enlightening paper by Friedman et al. (2000)[49] showed that for the two-class problem boosting is approximately equivalent to an additive model on a logistic scale with maximum Bernoulli likelihood as the model selection criterion. Using this knowledge they designed a superior classifier (LogitBoost) that fits such an additive model more directly and efficiently. A similar approach tree-based additive modelling has been used by Chipman et al. (2006)[26] to design a highly effective Bayesian

regression tree algorithm called BART (Bayesian Additive Regression Trees) that similarly uses a sum of small trees to approximate an additive model.

If we really do have to choose a single model, for some reason, how we do it? The Bayesian approach is that this is a decision problem. We should specify a utility function and compute the expected utility for each of our candidate models and pick the one with the highest expected utility. In practice, if we are considering a huge model space such as the space of tree models we will need to work with tractably small subset of models and choose the best from that subset of models. In the $\mathcal{M}$-completed case we are no longer even interested in finding the true model. However, we might be interested in using the model which is, in some sense 'closest' to the true model, if this is known. For this to work we need a distance measure between models. A *score function* (JQ Smith, 1988)[119] is used to compare two probability distributions, e.g. a probability distribution as reported by an individual, to an empirical probability distribution. For example, if a weather forecaster reports the probability of precipitation as 25% on a large number of occasions a consumer could note the number of such occasions on which precipitation actually occurred. If this is substantially different from 25% then we say that the forecaster is *poorly calibrated* (Dawid, 1982)[34]. Formally, suppose we have an exhaustive set of mutually exclusive hypotheses $\{E_j, j \in J\}$ for some countable set J, and a set of probabilities which are the probabilities of the respective hypotheses as reported by an observer, $\boldsymbol{p} = \{p_j, j \in J\}$. A score function is a mapping from each ordered pair $\{p, E_j\}$ to a real number $u(\boldsymbol{p}, E_j)$. A good property for a score function to have is that it should encourage (i.e. give the highest score) the user to report the probabilities that they genuinely believe are the probabilities of the hypotheses holding true. This is known

as a *proper* score function. Another feature we would want for a score function is that its value is related only to the probability that it attaches to the outcome that actually occurs. The probabilities that it attaches to other events are irrelevant. This is a *local* score function. It can be shown (Bernardo and Smith, 1997)[10] that a local, proper score function must be of the form:

$$u(q, E_j) = A \ln(q_i) + B_j$$

where A and the Bs are arbitrary constants and $A > 0$. Suppose there is a 'true' probability distribution and we report the distribution $q = q_j, j \in J$. The expected loss of utility using a logarithmic score function as our utility is given by

$$\delta(q|p) = A \sum_{j \in J} p_j \ln \left(\frac{p_j}{q_j} \right), A > 0$$

This is the *Kullback-Leibler divergence* (Kullback and Leibler, 1951)[89]. It is not symmetric between p_j and q_j. The continuous version is

$$H(\theta) = \int \left(\frac{p(y|\,\theta)}{f(y)} \right) f(y) \mathrm{d}y = \mathbb{E} \left\{ \ln \left(\frac{p(y|\,\theta}{f(y)} \right) \right\}$$

$H(\theta)$ is the distance between the true distribution $f(y)$ and the model $p(y|\,\theta)$. Since Bayesian models are probability distributions, we can use the Kullback-Leibler divergence as a 'distance' between models. In the $\mathcal{M}$-completed case we would be interested in model selection methods which converged to the model in our model class which is closest in KL distance to the true model. In the $\mathcal{M}$-open situation there is no true model, so we must use purely empirical criteria to choose between models. There are a number of relevant criteria which are either Bayesian or closely related to Bayesian ideas. In summary, individual tree models on their own are unstable, their structure is deceptively interpretable to the uninformed and their predictive

accuracy is fair. When model averaged or used in an ensemble method like bagging or boosting they are stable and highly accurate.

Chapter 5

More Recent Advances

5.1 General

Since this thesis began there have been some advances in the field, and we discuss some of these here.

5.2 Treed models

Chipman, George and McCulloch (2002)[28] extended their original Bayesian CART regression model to a model which fits a linear regression model at each leaf node in a Bayesian CART, instead of just using means or proportions to represent the node. The idea is not new, having been suggested by Karalič (1992)[83]. Quinlan (1992)[108] also implemented treed models as part of the pruning stage of his M5 program. The advantage of treed models is that putting more flexibility at the node level means the tree does not have to be so big. Suppose we have a tree T with a parametric model for Y at each leaf node. For $\boldsymbol{x}$ values associated with leaf i we have $Y|\,\boldsymbol{x} \sim f(\boldsymbol{x}|\,\theta)$. The covariates can thus vary continuously within the space defined by a leaf node and the tree construction algorithm does not have to create

separate leaves to model this. The model used at the node containing covariate $\boldsymbol{x}$ is $N(\boldsymbol{x}'\beta_i, \sigma^2)$ where $\theta_i = (\beta_i, \sigma_i^2)$. This model allows different slopes and variances at each leaf node and CGM give an example in their paper where this seems to be useful. The prior $p(\Theta, T)$ is expressed as $p(\Theta|\, T)p(T)$. The $p(T)$ part is chosen as before in their Bayesian CART paper, described in Chapter 2. Within the specification of $p(\Theta|\, T)$ the specification of $p(\theta)$ is noted to be rather tricky. Too informative a prior can override the limited information in a node, but too diffuse a prior will dilute the likelihood in the sense of the Jeffreys-Lindley paradox. The paradox says that if we have two models $\mathcal{M}_0$ and $\mathcal{M}_1$, where the dimension of $\mathcal{M}_0$ is less than the dimension of $\mathcal{M}_1$, the Bayes factor B_{10} has the properties:

- $B_{10} \to 0$ as the sample size $n \to \infty$ (Lindley, 1957) [92]
- $B_{10} \to 0$ as the prior variances of additional parameters $\to \infty$ (Bartlett, 1957)[8]

Under either of these conditions the Bayes factor model selection approach leads one to choose the simpler model. The first case is especially noteworthy because it conflicts with results from conventional frequentist practice. Suppose that $\mathcal{M}_0$ is a submodel of M_1 and we use a significance test (or a hypothesis test, the distinction does not matter much here) to tell us whether a model parameter should be in our model. Significance tests assume that sampling error is the largest source of error, and as this is steadily eliminated by larger and larger sample sizes they increasing tend to deliver a 'significant' result. Thus, if we use enough data to test the hypothesis that the model parameter is zero it will eventually become 'significant', and we will end up including it in our model. The second result shows that even if a proper prior is used a Bayesian can find himself in trouble by using too vague a prior. Intuitively, if there is no prior information either way, the Bayes factor should point to the simpler

model, in accordance with Occam's Razor. There is a close relationship between Bayes' theorem and Occam's Razor and the latter can be derived from the former in many cases (Jefferys and Berger, 1991)[82]. The addition of models at the tree nodes does seem to be an advance, but the requirement for careful tuning of the model priors is a substantial disadvantage in a nonparametric model. (We use the term 'nonparametric' in the sense it is usually used by Bayesians: a model with a variable number of parameters, determined by the data. The frequentist 'distribution-free' idea, of making inferences without specifying a model, does not make sense to us.)

5.3 Bayesian MARS

MARS, or Multivariate Adaptive Regression Splines, is a nonparametric regression technique invented by Jerome Friedman (1991)[50]. It uses a model which is an extension of the additive model of Friedman and Silverman (1989)[53]. The additive model is given by

$$f(x) = \sum_{j=1}^{p} g_j(x_j) \tag{5.3.1}$$

where the g_j are unknown smooth functions to be estimated from the data. Some popular classes of smooth functions used, (e.g. polynomials, splines) can be expressed in terms of basis functions B_i in which case Equation 5.3.1 becomes

$$f(x) = \sum_{i=1}^{k} \beta_i B_i(x_i) \tag{5.3.2}$$

(Additive models can also be extended to *generalised* additive models by introducing a link function similar to that used in generalised linear models.) In the case of

recursive partitioning, the basis functions are partitions of the data, i.e. $B_i = I \in R_i$ and the mutually exclusive subsets R_i cover the space of the data D, i.e. $\bigcup_{j=1}^{N} R_i = D$. for some positive integer N. MARS extends the additive model by using tensor spline basis functions, in order to allow interaction terms between predictors to be constructed as well. The basis functions are

$$B_i(x) = \begin{cases} 1 & i = 1 \\ \prod_{j=1}^{J_i} [s_{ji}(x_{\nu(j,i)} - t_{ji})_+] & i = 2, 3, \ldots \end{cases} \tag{5.3.3}$$

and we use $[x]_+$ to denote max(x,0). The s_{ji} (sign functions) can only take the values ± 1 The t_{km} are the knot locations. The largest of the J_i is known as the *degree*. The index of the predictor currently being used is given by $\nu(k, m)$. For example, if the degree is 2, a basis function might be

$$(x_2 - 1.3)_+[-(x_5 - 3.7)]_+$$

The use of such truncated linear basis functions leads to functions which are continuous but not differentiable. If a smoother look is considered desirable then truncated cubics can be used instead. Holmes and Denison's Bayesian version of MARS[39] sets θ as the vector of unknown parameters $J_i, s_{ji}, \nu ji$ and t_{ji}. As with Bayesian CART, it computes a Reversible Jump MCMC simulation of the posterior. The model move types are:

1. Change in knot location

2. Add a basis function

3. Delete a basis function

The code used by Holmes and Denison stores every model encountered in the chain,

which makes model averaging straightforward and this provides stability (at the expense of interpretability, of course). Bayesian MARS is a clear improvement over Bayesian CART for regression models, as it does not impose step functions on data which is known to be smooth. However, it retains the practical advantages of the Bayesian approach, such as the natural penalty function to avoid overfitting, and a better (non-nested) search of the model space.

5.4 Bayesian Partition Models

The Bayesian Partition models of Holmes et al. (1999, [75]) and Denison et al. (2002, [36]) divide the data space up into mutually exclusive regions using the idea of a Voronoi tessellation. Suppose we have a finite set of coplanar points $P_1, P_2, \ldots, P_M$. These we call centres. For each P_i we draw a boundary which includes all points nearer to this centre than to any other centre. Such a boundary is called a *Voronoi polygon* and the set of all such polygons for a set centres is called a *Voronoi tessellation*. An example of a Voronoi tessellation is given in Figure 5.1. (Distance is defined in such a way that different directions can be given different weights.) We denote by R_i the region enclosed by the ith polygon, with n_i denoting the number of data points in region i. In the regression case it is assumed that a linear model with a normal error term holds, with the β parameters varying from polygon to polygon, and a globally-constant variance.

$$f(Y_i|\,\theta_i) = N_K(Y_i|\,X_i\beta_i, \sigma_i^2 I) \quad i = 1, 2, \ldots, M$$

In the classification case a multinomial is used:

$$f(Y_i|\,\theta_i) = \prod_{j=1}^{n_{ii}} \left\{ \frac{n_i!}{\prod_{k=1}^{K} n_{ik}!} \prod_{k=1}^{K} p_{ik}^{n_{ik}} \right\}$$

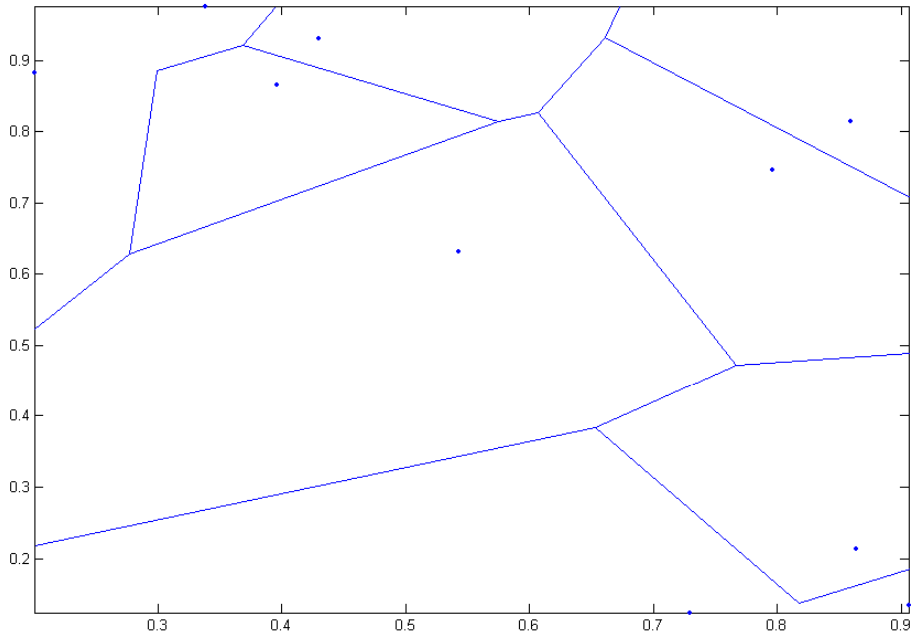

Figure 5.1: Example of a Voronoi tesselation.

The likelihood given the parameters and the tessellation is given by

$$f(Y_i|\,\theta, T) = \prod_{i=1}^{M} f(Y_i|\,\theta_i)$$

The use of the conjugate priors (normal-gamma or Dirichlet as appropriate) within each regions allows the marginal to be computed analytically as

$$p(Y)|\,T) = \int_{\theta} f(Y_i|\,\theta_i, T)p(\theta|\,T)d\theta \tag{5.4.1}$$

The conjugate normal-gamma prior is given by

$$\begin{aligned} p(\sigma^{-2}) &= \text{Gamma}\left\{\tfrac{\gamma_1}{2}, \tfrac{\gamma_1}{2}\right\} \\ p(\theta_i|\,T, \sigma^2) &= N(\beta_i|\,0, \sigma^2 \boldsymbol{\lambda_i}^{-1}), \quad i = 1, \ldots, M \end{aligned}$$

where $\boldsymbol{\lambda_i}$ is the prior precision matrix of β_i and γ_1, γ_2 are hyperparameters. Equation 5.4.1 can be evaluated to obtain

$$p(Y|\,T) = \frac{\gamma_2^{\frac{\gamma_1}{2}}\Gamma\left\{\frac{1}{2}(\gamma_1 + n)\right\}}{\pi^{\frac{n}{2}}\Gamma\left(\frac{\gamma_1}{2}\right)} a^{-\frac{\gamma_1+n}{2}} \prod_{i=1}^{M} |\,\boldsymbol{\lambda_i}|^{\frac{1}{2}}|\,\boldsymbol{V_i}|^{\frac{1}{2}}$$

where $|\cdot|$ denotes the determinant of a matrix, $\Gamma(\cdot)$ the gamma function and

$$\boldsymbol{V}_i = (X_i' X_i + \boldsymbol{\lambda_i})^{-1}$$

$$a = \gamma_2 + \sum_{i=1}^{M} Y' \left(\boldsymbol{I} - \boldsymbol{X_i V_i X_i'}\right) Y_i$$

and $\boldsymbol{V}_i$ is the posterior variance of β_i and $\boldsymbol{I}$ is the identity matrix. This is just the usual Bayesian linear model of Lindley and Smith (1972)[95]. For the classification case a Dirichlet prior is used

$$p(\theta|\, T) = \text{Dirichlet}(\alpha_1, \ldots, \alpha_K)$$

As in the case of Bayesian CART, this leads to

$$p(Y|\, T) = \left\{ \frac{\Gamma\left(\sum_{i=1}^{K} \alpha_k\right)}{\prod_{i=1}^{K} \Gamma(\alpha_k)} \right\} \prod_{i=1}^{M} \frac{\prod_{i=1}^{K} \Gamma(n_{ik} + \alpha_k)}{\Gamma\left(n_i + \sum_{i=1}^{K} \alpha_k\right)}$$

The posterior is evaluated by the same sort of Reversible Jump MCMC scheme as used for the Bayesian CART. The method starts with a single centre for regression or K (the number of classes) centres for classification. The following moves types are used:

- Construct a new region by adding a new centre at a random point from those available
- Delete a region by deleting its centre
- Move a centre to a new, random location
- Alter the distance weighting of different directions

As a partition-based modelling approach, this seems to be a distinct improvement over Bayesian CART, as the regions are not constrained to be (hyper)-rectangular, nor lined up with the coordinate axes. The hierarchical nature of the tree algorithm means that model parameters are related through the tree structure and the resulting correlations make for poor mixing in the posterior space. The BPM structure seems to be a more straightforward approach.

5.5 Generalised Ridge Regression models

The generalised ridge regression approach of Denison and George [37] takes a novel approach to Bayesian model averaging. Consider a data set $\mathcal{D}$ consisting of a response variable y and p predictor variables x_1 to x_p, i.e. of the form $\{y_1, x_{11}, x_{12}, \ldots, x_{1p}, x_{21}, x_{22}, \ldots, x_{np}\}$ and a linear regression model

$$Y = \beta_0 + \boldsymbol{X}\beta + \varepsilon$$

where $Y = (y_1, \ldots, y_n)'$, $\boldsymbol{X}$ is the matrix of x_{ij} values, β is a column vector of coefficients to be determined from the data, and β_0 is an unknown coefficient and ε is a zero mean column vector of independent errors. A straightforward but computationally expensive model averaging approach might be to consider averaging over all 2^p possible regression models, where each predictor can be in or out of the model.

$$\mathbb{E}(Y|\, x, \mathcal{D}) = \sum_{i=1}^{p} \beta_i^* x_i$$

where

$$\beta_i^* = \sum_{j \in \mathcal{M}} \mathbb{E}(\beta_i|\, j, \mathcal{D}) p(j|\, \mathcal{D})$$

and the index j runs over all models in the model space $\mathcal{M}$. Denison and George argue that the β_i^* can be considered as weighted versions of the least-squares estimates of the full model, so that $\beta_i^* = w_i\hat{\beta}$ for some shrinkage weights w_i. The key idea is to place the priors on the shrinkage weights. This is different to the usual approach, which is to have a prior on the inclusion probabilities of each predictor (see George and McCulloch, 1993)[56]. This approach uses every predictor, at least to some extent (see Copas, 1983.) From a theoretical standpoint, shrinkage should reduce the effects of model uncertainty (see Draper, 1995)[42]. James and Stein (1961)[80] showed that for the mean square error (MSE) between the true model parameters and the shrunken estimates can be less than the MSE between the true parameter values and the least squares estimates, for certain choices of the weights. The trick is to find the best weights. There is substantial empirical evidence that shrinkage works (see, for example, Dempster et al., 1977 [35], Volinsky, 1997 [132]). It is a widely used technique in Bayesian circles and is implemented via Hierarchical Bayesian models (see Gelman et al., 1995 [54]). Denison and George's version is similar to a Generalised Ridge Regression (Hoerl and Kennard, 1970 [73], Goldstein and Smith, 1974 [58], Hocking et al., 1976 [72]). One of the big advantages of approaching model averaging from this direction is that the computations scale linearly with p, the number of predictors, instead of exponentially. Denison and George suggest using the additional CPU cycles thus made available to implement more complex nonparametric models.

5.6 BART

Chipman, George and McCulloch (Chipman et al., 2006) [26] have created a regression tree model that they claim has very high predictive performance. It uses a model

rather like an additive model (Hastie and Tibshirani, 1990)[69] but instead of using spline functions as its basis functions it uses small trees, usually a large number (100 or more) of them. This sum-of-trees approach enables additive structure to be modelled well, but the individual trees can also capture complex interactions well, leading to good overall structural flexibility.

We have n observations of $(y, x_1, \ldots, x_p)$ and we assume that

$$y = f(\boldsymbol{x}) + \sigma\epsilon, \qquad \epsilon \sim N(0,1)$$

where the unknowns are f and σ. Let:

T denote the tree structure, including split rules and split points,

$M = \{\mu_1, \ldots, \mu_b\}$ denote the means of the b bottom nodes (leaves)

$g(\boldsymbol{x}; T, M)$ denote the function which maps a particular $\boldsymbol{x}$
to a particular prediction given by μ

$(T_1, M_1), \ldots, (T_m, M_m)$ denote a set of m trees and their μs

The model is given by

$$y = g(\boldsymbol{x}; T_1, M_1) + \cdots + g(\boldsymbol{x}; T_m, M_m) + \sigma z, \qquad z \sim N(0,1)$$

It is similar to an additive model, but with trees used in place of the usual basis functions. Chipman et al. refer to the trees as an "over-complete basis" as there is a high degree of redundancy (although an "over-complete spanning set" would be more accurate terminology). The set of parameters $\theta = \{(T_1, M_1), \ldots, (T_m, M_m), \sigma$ is very large and a strong prior (a 'regularisation' prior) $\pi(\theta)$, is used to keep the trees small. The prior specification process draws heavily on their earlier work on Bayesian

CART:

$$\begin{aligned}\pi\{(T_1, M_1), \ldots, (T_m, M_m), \sigma\} &= \left\{\prod_j \pi(T_j, M_j)\right\} \pi(\sigma) \\ &= \left\{\prod_j \pi(\mu_{ij}|T_j)\pi(T_j)\right\} \pi(\sigma)\end{aligned}$$

and they only need to choose $\pi(T), \pi(\sigma)$ and $\pi(\mu|T)$. The method performs well and has the advantage that it is a full Bayesian model, so that posterior uncertainty is properly captured. It mixes much more efficiently than Bayesian CART and restarting the model leads to very similar results each time, so much so that they have mostly worked with single long runs.

5.7 Non-Bayesian approaches: Random forests, Products of trees

As well as the explosive growth in recent Bayesian approaches, there has been more non-Bayesian work on tree-based models. A couple of examples are illustrated here.

5.7.1 Random forests

Breiman's Random forests [15] idea adds another layer of randomness to the construction of bagged trees. A large number of bootstrap samples of the data are created and a tree is grown on each one. However, whereas in the usual CART construction algorithm, where when a split is required all the predictor variables are considered as candidates, in the random forest version only a subset of predictors is considered. The number variables considered, k, is much less than the number of predictors p, the default value of k is $\sqrt{p}$. As with bagged trees, the classification of a new data point is obtained by majority voting of all the different trees (or averaging in the case of

regression). ('Forest' is Breiman's word for a collection of trees.) However, the choice of k seems to be fixed, whereas its value should really be data-driven. One very nice feature of the method is the ability to come up with an internal estimate of error rate. When a bootstrap sample is taken from a sample of N data, the probability of a particular data point not being chosen is $\left(1-\frac{1}{N}\right)^N$. We recall that one definition of e is $\lim_{n\to\infty}\left(1+\frac{1}{N}\right)^N$ (Chirgwin and Plumpton, 1970)[29], so for large N we have $\left(1-\frac{1}{N}\right)^N \approx e^{-1} = 0.3679\ldots$ i.e. about one-third of the data points are not chosen. These so-called 'out-of-the-bag' points can be used to create error estimates, by sending them down the tree for prediction.

5.7.2 Products of trees

This is a classification model devised by Ferreira, Denison and Hand,[46] which is a generalisation of the Naïve Bayes model (also known as *Idiot's Bayes*). The Naïve Bayes model is a classifier which makes the strong-and rather unlikely- assumption that the univariate components of the feature vector are independent of one another. That is, if the feature vector, consisting of p features, is denoted by x, and the categorical response by Y, then

$$p(x|\,Y=k) = \prod_{\nu=1}^{p} p(x_\nu|\,Y=k)$$

for $k = 1,\ldots,K$. The extraordinary thing about this model is that it actually works well in many situations. Hand (2001b)[62] lists two studies in which Naïve Bayes was the best performing classifier from amongst several, and no less than sixteen other studies in which Naïve Bayes did well. In addition to this, he quotes a Monte Carlo study in which it also does well. Hand points to a possible reason why Naïve Bayes might do well: *it has fewer parameters than most alternatives.* If we think in terms

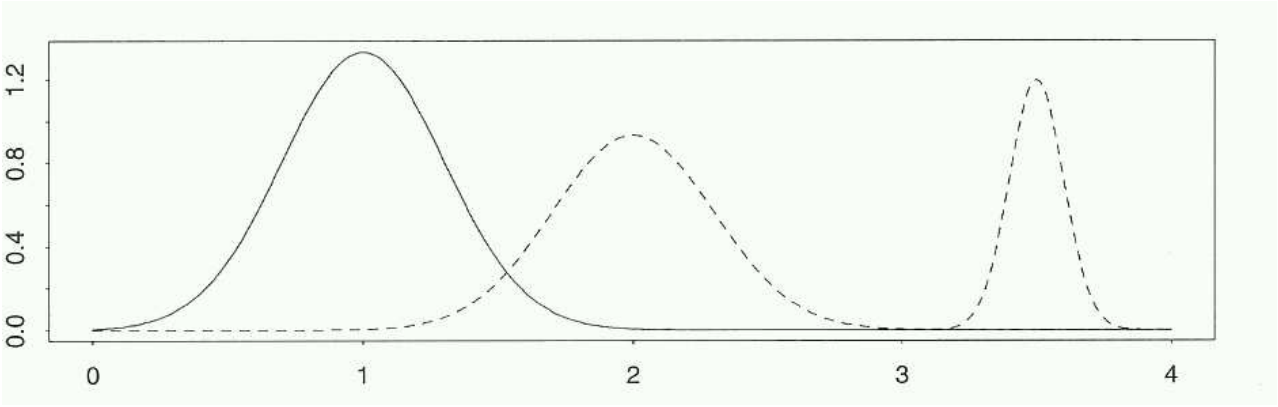

Figure 5.2: Class A: solid line, Class B dotted line.

of a bias-variance trade-off, it may be biased with a low variance. Naïve Bayes uses the discriminative (or diagnostic) paradigm described in Chapter 5, and clearly it is possible to achieve a good classification even if one's model of some of the classes is far from the truth. In Fig. 5.2 (taken from Ripley (1996), [112]) the densities of class 1 (solid line) and class 2 (dashed line) are shown. Fig 5.3 shows the class probabilities versus x for a particular model. Despite the major differences between the actual and estimated class densities this model may perform well in practice. The products of trees model treats the feature vectors separately and searches for a partition that maximise the Shannon information $\sum_{i=1}^{2}\sum_{k=1}^{K} p_{ik}\log(p_{ik})$, where p_{ki} is the relative frequency of class k in partition i. This achieved by a forward stepwise search until a large (probably overfitting) model has been constructed. Then a backwards deletion is used. A weight function is attached to the predictors in a way is similar to the shrinkage weights of GRR. The weights are defined in terms of a 'distance' measure between the relative frequency of class k and the average relative frequency of a class. The distance measure chosen is the l_α norm (Bronshstein & Semendyayev, 1985,[17]),

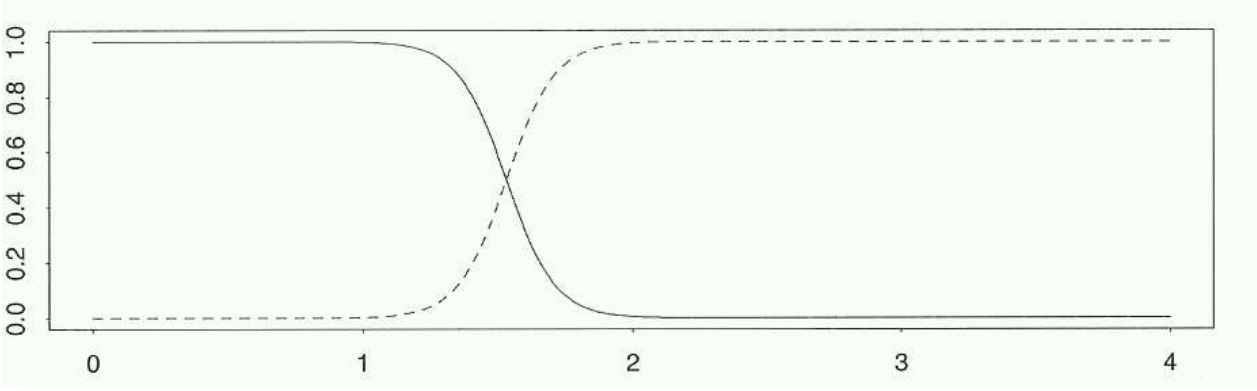

Figure 5.3: Probabilities of Classes versus x, from assumed model.

given by

$$w_\alpha(x_\nu) \propto \left\{ \sum_{k=1}^{K} \left| p_{\nu k} - \frac{1}{K} \right|^\alpha \right\}^{\frac{1}{\alpha}}$$

so

$$p(Y = k|\, x) \propto \left\{ \prod_{\nu=1}^{p} [p_{x_\nu} | Y = k]^{w(x_\nu, k)} \right\} p(Y = k)$$

A uniform prior which gives equal probability to each class is used: $p(Y = k) = \frac{1}{K}$. Application of Bayes' theorem gives

$$p(Y = k|\, x) \propto \left\{ \prod_{\nu=1}^{p} \left(\frac{n_{k\nu j}}{n_{k\nu}} \right)^{w(x_\nu, k)} \right\}$$

where j is the cell into which the variable x_ν falls, $n_{k\nu j}$ the number of class k data points that falls in cell j and $n_{k\nu}$ the number of class k points which have a value for predictor x_ν.

Chapter 6

Experimental Comparisons of Bayesian CART Trees

We have carried out detailed comparisons of the predictive performance of the CGM and DMS tree models, for both classification and regression problems.

6.1 Classification models

6.1.1 The data set

The breast cancer data set from the University of California at Irvine repository of machine learning data sets [130] was used to assess classification accuracy. This data set was contributed by William H. Wolberg of University of Wisconsin Hospitals, Madison. It contains 9 predictive variables concerning cellular characteristics and a class response variable taking the values 1 or 2 denoting benign or malignant tumours respectively. The data used here consists of the 683 observations which do not contain missing values.

6.1.2 Methodology

Ten-fold cross-validation was used to give honest estimates of the classification accuracy. That is, we divided the data set into 10 nearly equal-sized blocks as follows. The data set was put into the SAS statistical package (SAS Software Ltd) [116]. A new column was created and filled with uniformly distributed random numbers in the range [0, 1] using the built-in function RANUNI. The data set was then sorted (i.e. ranked) into ascending order of this random column. The first 10% of the data was then allocated to block 1, the second 10% block 2, and so on up to block 10. Since 10 is not a divisor of 683 the blocks are not all exactly the same size. The exact sizes of the blocks are given in Table 6.1.2 and Table 6.1.2

Program	Training set	Obs	Test set	Obs	Misclassifications	Leaves
CGM	not_block1	615	block1	68	4	9
CGM	not_block2	615	block2	68	6	13
CGM	not_block3	615	block3	68	3	12
CGM	not_block4	614	block4	69	1	10
CGM	not_block5	615	block5	68	0	11
CGM	not_block6	615	block6	68	2	11
CGM	not_block7	614	block7	69	1	10
CGM	not_block8	615	block8	68	2	11
CGM	not_block9	615	block9	68	2	11
CGM	not_block10	614	block10	69	2	9

Table 6.1.2: Ten-fold Misclassification rates: CGM data

Program	Training set	Obs	Test set	Obs	Misclassifications	Leaves
DMS	not_block1	615	block1	68	3	9
DMS	not_block2	615	block2	68	4	15
DMS	not_block3	615	block3	68	2	10
DMS	not_block4	614	block4	69	1	16
DMS	not_block5	615	block5	68	2	9
DMS	not_block6	615	block6	68	1	12
DMS	not_block7	614	block7	69	2	13
DMS	not_block8	615	block8	68	0	16
DMS	not_block9	615	block9	68	2	17
DMS	not_block10	614	block10	69	1	12

Table 6.1.2: Ten-fold Misclassification rates: DMS data

Training sets were constructed as follows. The union of blocks 2 to 10 is called not_block1, the union of blocks 1 to 9 is designated not_block10 and so on with the union of all blocks except block X being designated not_blockX. We use the training set to find a number of trees. Both programs have the ability to find the best k trees on the training set. We then assess these trees on the complementary set of observations which have not been used to construct the trees. So, for example, if we trained on the training set not_block1 then we would use block 1 as our test set. The count of misclassifications on the test set is our criterion of 'best classification'. Where ties occur, the tree with lowest deviance is taken. In a real-life setting where we would have to take a decision about, say, the sort of medical treatment which a breast cancer patient should receive, the simple count of errors that we have used here would not be a good criterion to use and we would be well advised to elicit utilities

and conduct a formal decision analysis (Pratt, Raiffa and Schlaifer, 1965)[77]. In such a context the consequences of prediction errors are different. Misclassifying 'benign' cells as 'malignant' has totally different losses (i.e. costs, interpreted widely) to misclassifying 'malignant' cells as 'benign'. However, we are not actually taking a medical decision here, so counting errors is probably not an egregious thing to do. Both methods have been previously shown to classify better than standard CART (Chipman, George and McCullough, 1998, [27]; Denison, Mallick and Smith, 1998 [38]). Here we compare them with each other.

6.1.3 The Priors

The DMS prior had a prior mean tree size of 2.8 leaves. The setting of the prior mean is a simple matter of setting a single parameter to the desired value in the DMS code, but it is rather more complex in the CGM code. In the CGM code *two* parameters have to be set, α and β. Furthermore, the only way to find out what prior mean will result from these settings is to do a simulation. The α is a tree size parameter, while β is a 'bushiness' parameter. Trees with large β will tend to have all leaves at the same depth. We have not replicated the simulation, but CGM provide the simulation results of a few settings in their paper and we have used those. The setting of α to 0.95 and β to to 1.5 results in a prior mean of 2.9 according to CGM's simulation.

6.1.4 Classification accuracy

The results are shown in and 6.1.2 and 6.1.2 The results are shown in Table 3.1.1. Starting with very similar prior means on the same training set, and comparing the performance on the same test set, the median number of errors is similar for the two

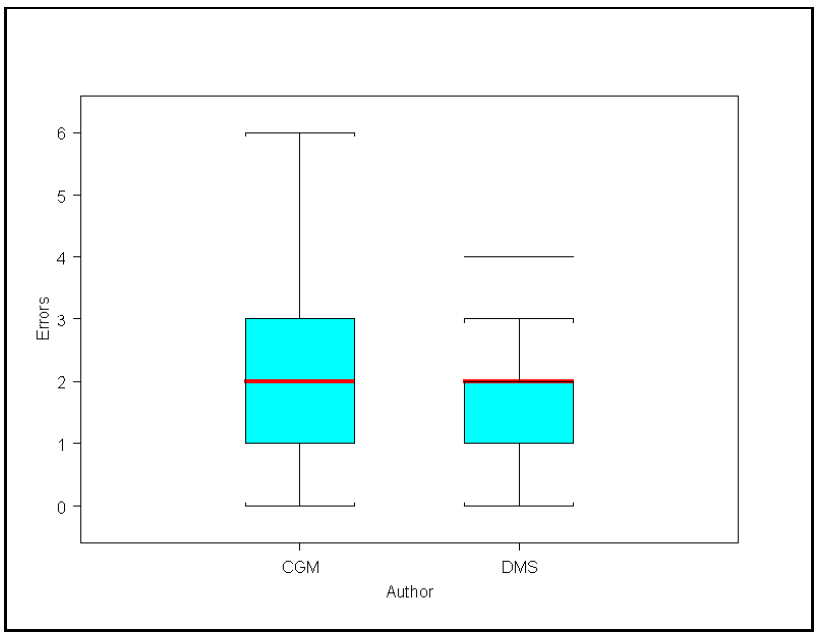

Figure 6.1: A boxplot of misclassification rates

methods but DMS trees are less likely to make a large number of errors. See Fig **??**owever, the single CGM result where 6 misclassification errors were made could be viewed as an outlier. We would like to formally test to see whether DMS trees are more accurate. We could use a classical significance test, such as the McNemar form of χ^2 test, but we prefer to use a model-based approach. (The defects of classical significance tests are well documented. See, for example Anderson, Burnham and Thompson (2000) [4])

We will test the difference between CGM and DMS, as far as misclassification accuracy is concerned, by means of a Poisson regression model. Let the number of misclassification errors made on training set i be modelled by a Poisson distribution

Data	CGM	DMS
Average N° of Misclassifications	2.30	1.80
Var. of N° of Misclassifications	2.90	1.29
Average N° of Leaves	10.70	12.90
Var. of N° of Leaves	1.57	8.99

Table 6.1: The means and variances of the number of misclassifications and the number of leaves.

with parameter λ_i. We assume that the λ_i's are not independent, but are exchangeable. This implies that they can be can be modelled by having them drawn from a common distribution using a hierarchical model (de Finetti's representation theorem). We also allow that there may an effect due to a difference between the CGM trees and the DMS trees. We model this by the parameter β. The indicator variable x takes the value 1 for CGM and 0 for DMS.

$$\begin{aligned} y_i &= \text{Poisson}(\lambda_i) \\ \log(\lambda_i) &= \alpha + \beta x_i \\ \alpha_i &\sim N(\mu_\alpha, \sigma_\alpha^2) \\ \beta &\sim N(\mu_\beta, \sigma_\beta^2 \end{aligned} \tag{6.1.1}$$

If there is a significant difference between CGM and DMS then beta will be significantly different from zero. We compute a 95% credible interval for beta and say that beta is significantly different from zero if the interval does not include zero. We use the vague but proper normal prior $N(\mu_\alpha, \sigma_\alpha^2)$ for the αs and $\mu \sim N(0, 10^4)$, $\sigma_\alpha^2 \sim$ Gamma$(10^{-2}, 10^{-2})$ and the same for the βs, i.e $\mu \sim N(0, 10^4)$, $\gamma_\beta^2 \sim$Gamma$(10^{-2}, 10^{-2})$. We estimated the models using the WinBUGS Markov Chain Monte Carlo (MCMC)

package (Spiegelhalter et al, 1994)[133]. 100 000 iterations were used after a 10 000 iteration "burn-in" to allow convergence to occur. Verification of convergence was carried out informally by examination of the convergence traces for the parameters. The results are as follows. The column entitled 'MC error' is a Monte Carlo estimate of the standard error of the mean. Red results are significantly different from zero. In particular, the beta parameter is not significant, hence we conclude that there is no evidence of a difference in misclassification rates between the two tree methods. The priors were set to a mean of 2.8, and $\ln(2.8) \approx 1.03$, which is inside the 95% confidence interval for α in 7 out of the 10 results and only just outside for another 2 of them.

Node	Mean	SD	MC error	2.5%	Median	97.5%
Alpha[1]	0.7914	0.4317	0.00266	-0.0405	0.7988	1.612
Alpha[2]	1.136	0.4356	0.003366	0.1987	1.166	1.915
Alpha[3]	0.5054	0.4365	0.002175	-0.3279	0.4989	1.36
Alpha[4]	-0.06986	0.4849	0.001414	-1.111	-0.04669	0.8305
Alpha[5]	-0.07196	0.4875	0.001548	-1.114	-0.04963	0.8393
Alpha[6]	0.1453	0.4584	0.001576	-0.7867	0.1468	1.037
Alpha[7]	0.1431	0.4565	0.001567	-0.7943	0.1464	1.027
Alpha[8]	-0.07734	0.4868	0.001577	-1.119	-0.05378	0.826
Alpha[9]	0.3378	0.4404	0.001732	-0.5207	0.3317	1.202
Alpha[10]	0.1461	0.4591	0.001594	-0.7847	0.1466	1.032
Beta	0.0729	0.2449	0.001482	-0.403	0.05673	0.5998

??

Table 6.4: Misclassification Model Parameters

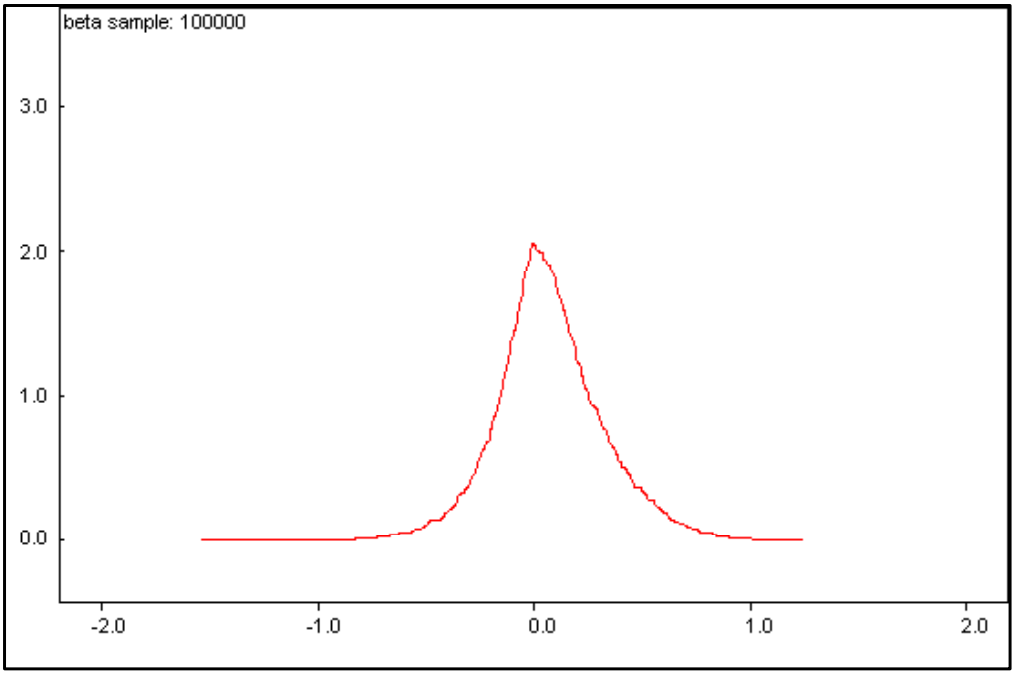

Figure 6.2: Beta Posterior

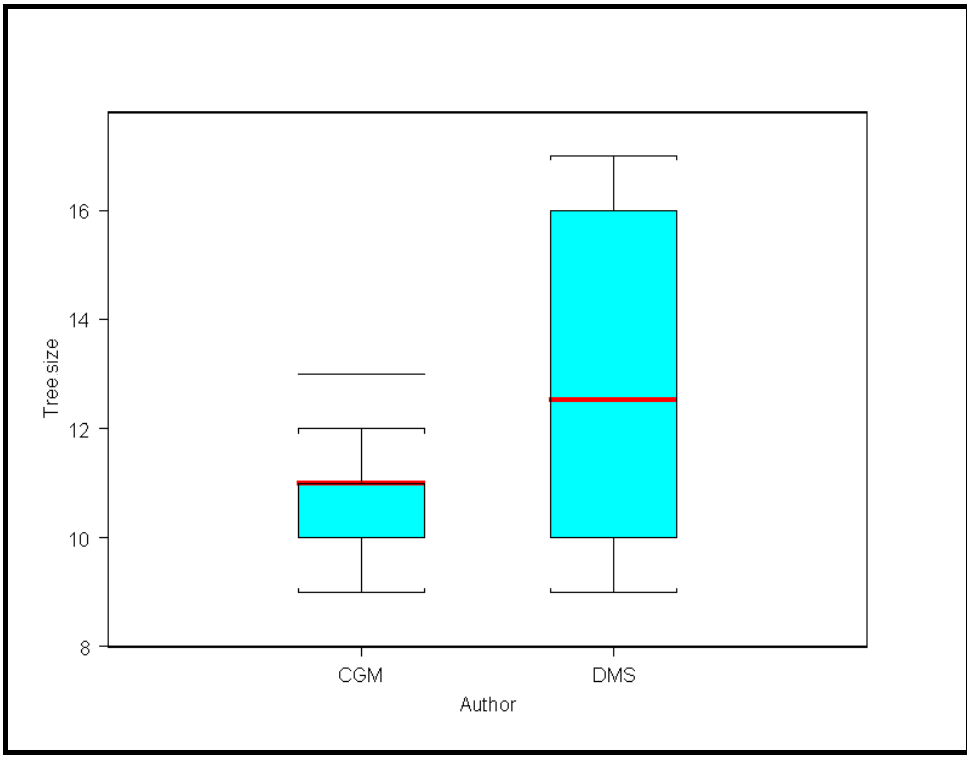

Figure 6.3: Tree size

6.1.5 Tree size

A similar analysis was carried out with respect to tree sizes produced by the two methods. With these settings, corresponding to almost identical prior means, Figure 6.3 suggests that the DMS trees tend to be larger than the CGM trees. This suggests that the trade-off between classification accuracy and tree size is balanced differently in the two methods.

Node	Mean	SD	MC error	2.5%	Median	97.5%
Alpha[1]	2.049	0.2491	8.769E-4	1.536*	2.058*	2.513*
Alpha[2]	2.507	0.2047	7.636E-4	2.086*	2.512*	2.891*
Alpha[3]	2.256	0.2297	8.758E-4	1.787*	2.262*	2.685*
Alpha[4]	2.428	0.2113	7.906E-4	1.995*	2.435*	2.827*
Alpha[5]	2.159	0.238	8.853E-4	1.668*	2.167*	2.603*
Alpha[6]	2.302	0.2236	7.88E-4	1.845*	2.309*	2.723*
Alpha[7]	2.304	0.2235	8.603E-4	1.846*	2.311*	2.721*
Alpha[8]	2.468	0.2074	7.753E-4	2.041*	2.474*	2.855*
Alpha[9]	2.506	0.2038	7.613E-4	2.091*	2.511*	2.888*
Alpha[10]	2.21	0.2331	8.618E-4	1.729*	2.217*	2.646*
Beta	0.1843	0.1281	7.56E-4	-0.05031	0.1798	0.4461

Table 3.5: Tree Size Model Parameters

We use a similar model as before to formally test the effect of method type (CGM versus DMS) on tree size. This time it is tree size, not the number of errors, that is modelled by a Poisson with parameter λ. We use the same priors and number of iterations (10^5 with 10^4 burn-in) as before.

The posterior for beta is shown in Figure 6.4. It shows that the DMS trees tend to be larger, with the bulk of the probability mass to the right of the zero ordinate. Of the 100 000 sample values from the posterior we found that only 6667 (6.7%) of them are less than zero. However, the 95% credible interval spans zero and so we conclude that the evidence for a size difference, while suggestive, is not overwhelming. (As can be seen from Table 3.5 the alpha coefficients are significant, but this is not the focus of our interest.)

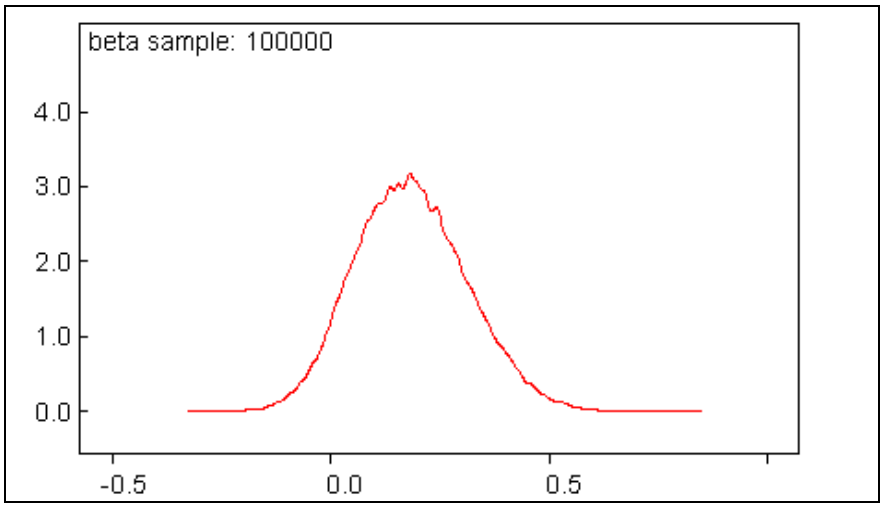

Figure 6.4: Beta Posterior Density in tree size model

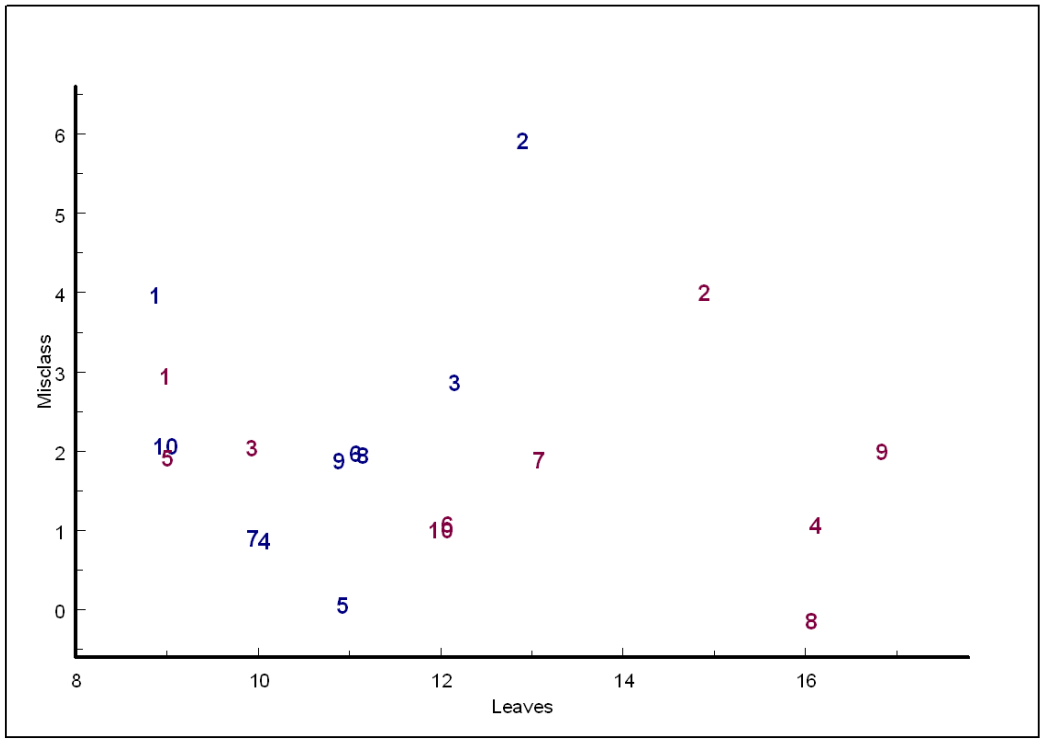

Figure 6.5: Number of leaves versus misclassification rate. Blue = CGM, Red = DMS, Numbers indicate fold

6.1.6 Discussion

In both the CGM and DMS programs the resulting tree sizes seem to bear little resemblance to the prior means. The prior means are 2.9 and 2.8 respectively whereas the mean numbers of leaves are 10.7 and 12.9 respectively. There does not seem to be much point manually setting the prior means for this data set, since they are overwhelmed by the data. Yet this data set is not large by data mining standards. In both cases the user is asked to provide a prior tree-size mean and this may be difficult in practice. A user, such as physician, may have opinions about which variables are most relevant to prediction of malignancy, but it is difficult to see how their medical knowledge could enable them to assess likely tree sizes. Tree size is nonetheless relevant, if only because very large trees, if used as the basis of policy, may in practice imply operational processes that are too complex to be usable. Classical tree methods use automatic mechanisms such as cost-complexity pruning to hide tree-size choices from the user. It is supposedly one of the advantages of the Bayesian approach that assumptions are made explicit, but the degree of control given to the user by CGM and DMS is different. In the DMS approach the user is asked to specify a prior for tree size only whereas the CGM method asks the user for information about 'bushiness' as well. DMS argue that this is inappropriate and unrealistic in practice. One of the chief advantages of the Bayesian methodology is that it allows one to incorporate everything one knows into the analysis. Here we seem to have a mismatch between the information that the method requires and the domain knowledge that the user actually possesses. Ideally, we should be asking the user to specify a number of domain-specific hypotheses. Each hypothesis would have implications in terms of the likely ranges of certain variables. We would then ask the user to attach prior

probabilities to these meaningful hypotheses rather than to something more specific such as tree size or tree bushiness.

Multiple Starts

A key difference between the two techniques is that CGM restart the MCMC run several times, (ten times was used in the experiments here). DMS prefer to do one long run. At the time of writing there is controversy about which of these is the better approach. The DMS code initially restricts the tree size to 5 leaves so that the early part of the tree space is explored thoroughly. This reduces the likelihood that an early bad decision can ruin the whole run. Of course, this feature could be added to the CGM code as well, but it is less critical there, as only one run of the 10 runs would be affected if the chain gets stuck. This work has not answered the question of whether one run is better than, as is appears that there are at least two criteria of interest, size and accuracy, and neither method beats the other on both measures.

Speed of execution

In order to run through 200 000 iterations the CGM code took 15 minutes and the DMS code took 30 minutes on a Sun Sparc Ultra 2. These are elapsed times on a machine with no other users. Thus both programs are acceptably fast for practical purposes on modern hardware without any form of code optimisation. (In fairness, it should be noted that the code for both algorithms was designed for research purposes and was not in any way 'productised' for the general user or optimised for speed.) To assess the time complexity of the two methods runs were carried for $10\,000, 20\,000, 50\,000, 100\,000 and 200\,000$ iterations. The results are shown in Figure

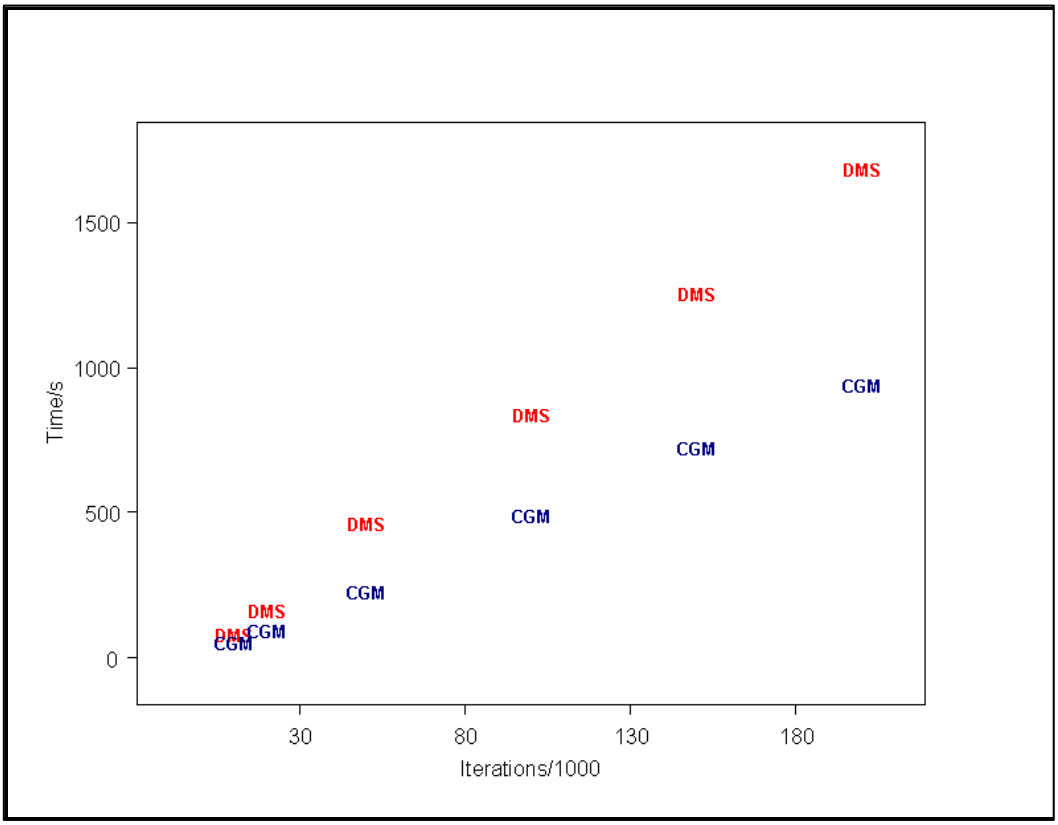

Figure 6.6: Run times by iterations

6.6. As can be seen, the run times are approximately linear in the number of iterations, at least over the range considered. This is not surprising since we are merely executing a loop more times. We would expect non-linearity only where the working data could not be stored in memory. More interestingly, the slope of the CGM algorithm is lower than that of the DMS algorithm, indicating shorter run times for large data sets compared to DMS.

6.2 Regression Models

We did a corresponding analysis for the CGM and DMS regression trees. Accuracy for the regression models was tested on the air data set of Bruntz et al. (1974) [19]. The air data set was an environmental study conducted at various sites in New York state from May to September 1973. The level of ozone was measured on 111 days along with solar radiation, the wind speed in miles per hour and the air temperature in degrees Fahrenheit. The version of this data set used here is the same one as used by DMS in their CART model work. The solar radiation variable is not used at all and the ozone variable has had a smooth effect removed. Following Denison et al. (1998,[38]), who follow Cleveland and Devlin (1988) [30], we work with the cube root of ozone., Two different values of the prior mean have been used: $\lambda = 2$ and $\lambda = 10$.The model we used to test the difference between methods is as follows:

$$\begin{aligned} y &= \alpha + \beta x + \varepsilon \qquad (6.2.1) \\ \alpha &\sim N(\mu_\alpha, \sigma_\alpha^2) \\ \beta &\sim N(\mu_\beta, \sigma_\beta^2) \\ \varepsilon &\sim N(0, \sigma_\varepsilon^2) \\ \mu_\alpha &= \mu_\beta = 0 \\ \sigma_\alpha^2 &= \sigma_\beta^2 = 10^4 \\ \sigma_\varepsilon^2 &\sim \text{Gamma}(10^{-2}, 10^{-2}) \end{aligned}$$

The dummy variable x takes the value 0 for CGM data and 1 DMS data. The model is not ideal, as the y variable is a sum of squares, and so should be regarded as chi-squared variable rather than a normal. Nevertheless, for the purpose of testing

Program	Training set	Obs	Test set	Obs	Sum of squares	Leaves	Settings
DMS	notblock1	100	block1	11	1.334322081	10	$\lambda = 2$
DMS	notblock2	100	block2	11	2.164885507	7	$\lambda = 2$
DMS	notblock3	100	block3	11	2.25927313	12	$\lambda = 2$
DMS	notblock4	100	block4	11	3.909994696	5	$\lambda = 2$
DMS	notblock5	100	block5	11	2.729697487	11	$\lambda = 2$
DMS	notblock6	100	block6	11	0.652424639	12	$\lambda = 2$
DMS	notblock7	100	block7	11	2.229991224	6	$\lambda = 2$
DMS	notblock8	100	block8	11	2.028209355	5	$\lambda = 2$
DMS	notblock9	100	block9	11	2.562012679	6	$\lambda = 2$
DMS	notblock10	99	block10	12	2.506195531	9	$\lambda = 2$

Table 6.2: Regression tree accuracy and tree size. DMS trees.

whether beta is zero or not we believe it to be a reasonable approximation.

6.2.1 Regression accuracy

The accuracy for the regression tree models is measured by the sum of squared differences between predicted values and the true values. This is a quadratic loss function, which implicitly assumes that under-estimating the true value is as good or bad as over-estimating it. In practice, there is often good reason to think that errors of one type are worse that errors of the other and an asymmetric loss function approach should be used. In this exercise we have no reason to prefer errors in one direction over another. The regression results are given in Tables 6.2 and 6.3

Program	Training set	Obs	Test set	Obs	Sum of squares	Leaves	Settings
CGM	notblock1	100	block1	11	3.440796	3	$\lambda = 2.066$
CGM	notblock2	100	block2	11	2.802729	3	$\lambda = 2.066$
CGM	notblock3	100	block3	11	2.474693	3	$\lambda = 2.066$
CGM	notblock4	100	block4	11	5.360386	3	$\lambda = 2.066$
CGM	notblock5	100	block5	11	3.944294	2	$\lambda = 2.066$
CGM	notblock6	100	block6	11	2.471454	2	$\lambda = 2.066$
CGM	notblock7	100	block7	11	3.942554	3	$\lambda = 2.066$
CGM	notblock8	100	block8	11	2.101373	3	$\lambda = 2.066$
CGM	notblock9	100	block9	11	3.224422	3	$\lambda = 2.066$
CGM	notblock10	99	block10	12	3.573155	2	$\lambda = 2.066$

Table 6.3: Regression tree accuracy and tree size. CGM trees. Settings: base = 0.94, power = 4.0

Program	Training set	Obs	Test set	Obs	Sum of squares	Leaves	Settings
CGM	notblock1	100	block1	11	1.444460418	3	$\lambda = 10$
CGM	notblock2	100	block2	11	2.80272915	2	$\lambda = 10$
CGM	notblock3	100	block3	11	2.474692974	4	$\lambda = 10$
CGM	notblock4	100	block4	11	6.638449439	4	$\lambda = 10$
CGM	notblock5	100	block5	11	4.020360349	3	$\lambda = 10$
CGM	notblock6	100	block6	11	4.352127639	4	$\lambda = 10$
CGM	notblock7	100	block7	11	3.282906464	3	$\lambda = 10$
CGM	notblock8	100	block8	11	2.589195382	3	$\lambda = 10$
CGM	notblock9	100	block9	11	3.248715828	4	$\lambda = 10$
CGM	notblock10	99	block10	12	3.573154782	2	$\lambda = 10$

Table 6.4: Regression tree accuracy and tree size. CGM trees. Settings: base = 0.992, power = 1.0E-5

Program	Training set	Obs	Test set	Obs	Sum of squares	Leaves	Settings
DMS	notblock1	100	block1	11	5.057	10	$\lambda = 10$
DMS	notblock2	100	block2	11	2.431	12	$\lambda = 10$
DMS	notblock3	100	block3	11	2.499513	14	$\lambda = 10$
DMS	notblock4	100	block4	11	3.489877	12	$\lambda = 10$
DMS	notblock5	100	block5	11	3.762476	8	$\lambda = 10$
DMS	notblock6	100	block6	11	1.000632	6	$\lambda = 10$
DMS	notblock7	100	block7	11	2.000926	7	$\lambda = 10$
DMS	notblock8	100	block8	11	1.915184	11	$\lambda = 10$
DMS	notblock9	100	block9	11	2.607529	12	$\lambda = 10$
DMS	notblock10	99	block10	12	2.888359	4	$\lambda = 10$

Table 6.5: Regression tree accuracy and tree size. DMS trees.

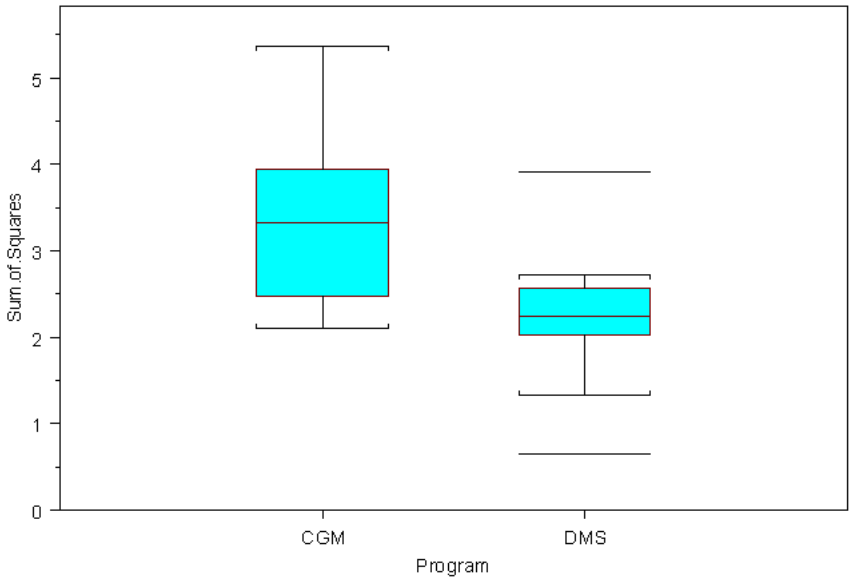

Figure 6.7: :Regression Tree Accuracy (Sum of Squares) $\lambda = 2$

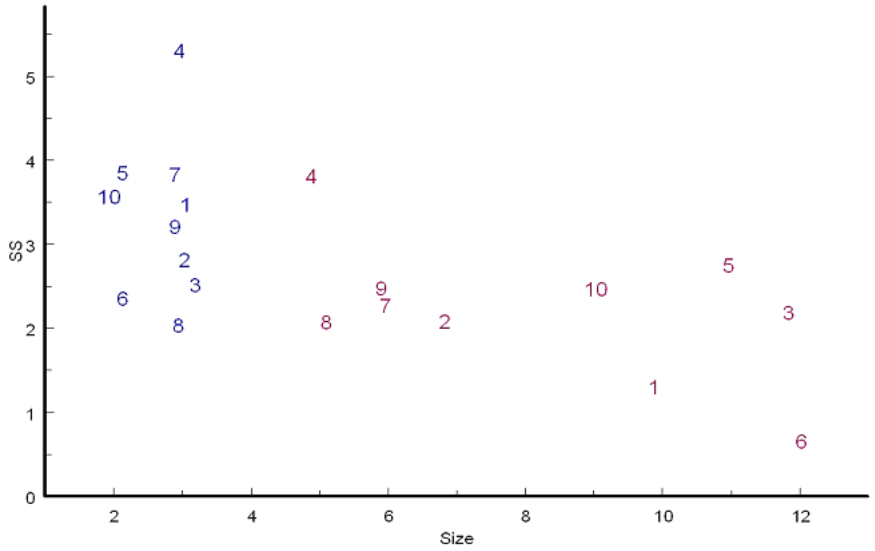

Figure 6.8: :Regression Tree Size versus Error sum of squares. The numbers denote test set folds. Colour denotes model: Blue = CGM, Red = DMS

Node	Mean	SD	MC Error	2.5%	Median	97.5%
α	3.333	0.3058	7.812E-4	2.731*	3.333*	3.941*
β	-1.096	0.428	0.001214	-1.936*	-1.092*	-0.2513*
σ^2	1.214	0.4022	0.002141	0.5583*	1.171*	2.125*

Table 6.6: Regression tree accuracy Model parameters: DMS trees.

First we examine the results for lambda (the prior Poisson mean) equal to 2. Figure 6.7 shows that DMS trees tend to have smaller sums of squares—greater accuracy—than the CGM trees. The results for tree size (Fig 6.8) are, however strikingly different. In fact, the smallest DMS trees are larger than the largest CGM trees. The CGM trees seem to be closer to the prior than the DMS trees are, for reasons that are not at all clear. Table 6.6 shows that all the parameters are significantly different from zero, since the 2.5% and 97.5% percentiles have the same sign in each case.

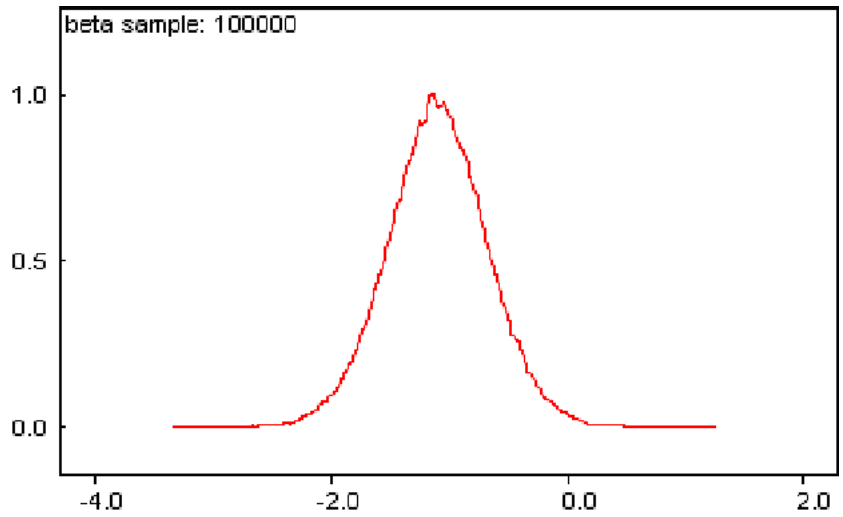

Figure 6.9: :Posterior distribution of beta for regression tre accuracy, $\lambda = 2$

Beta is about -1, and significant, according to our chosen standard. Of the 100 000 beta values used in the chain none of them had positive values, which is pretty strong evidence that DMS trees are more accurate than CGM trees.

6.2.2 Regression tree size

We used the same model to formally test the significance of differences in tree size as we did with the classification models:

$$\begin{aligned} y_i &= \text{Poisson}(\lambda_i) \\ \log(\lambda_i) &= \alpha_i + \beta x_i \\ \alpha_i &\sim N(\mu_\alpha, \sigma_\alpha^2) \\ \beta_i &\sim N(\mu_\beta, \sigma_\beta^2) \end{aligned}$$

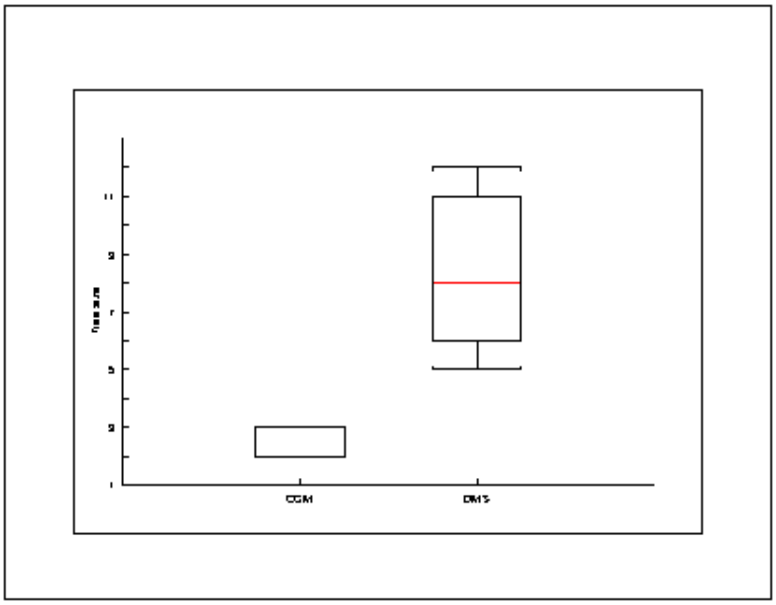

Figure 6.10: Regression tree sizes, $\lambda = 2$

The results are presented in Table 6.7. We again see that beta is significantly different from zero. In fact, in the 100 000 samples of beta, not one of the values was less than zero. This is broadly in line with what we would expect, given the dramatic differences in tree sizes seen in Figure 6.10. DMS trees are larger than CGM trees.

6.2.3 Results for $\lambda = 10$

We repeated the tests on regression tree accuracy and size, but this time used a prior mean of $\lambda = 10$. The same model was used to test the significance of accuracy and size differences. First, we consider accuracy.

Node	Mean	SD	MC Error	2.5%	Median	97.5%
$\alpha(1)$	0.2709	0.3159	0.004076	-0.1746	0.1983	1.047
$\alpha(2)$	0.137	0.2777	0.002935	-0.306	0.08642	0.827
$\alpha(3)$	0.3506	0.3423	0.004679	-0.1187	0.272	1.175
$\alpha(4)$	0.03913	0.2576	0.002066	-0.431	0.0144	0.6506
$\alpha(5)$	0.2702	0.3147	0.004054	-0.1744	0.1979	1.044
$\alpha(6)$	0.312	0.3287	0.004372	-0.1424	0.2354	1.109
$\alpha(7)$	0.09134	0.2671	0.002501	-0.3613	0.05239	0.7427
$\alpha(8)$	0.04088	0.256	0.002088	-0.4265	0.01715	0.6488
$\alpha(9)$	0.09005	0.2653	0.002532	-0.3596	0.0519	0.7378
$\alpha(10)$	0.185	0.2891	0.003347	-0.252	0.125	0.9044
β	1.897	0.2466	0.003651	1.279*	1.947*	2.253*

Table 6.7: Regression tree size, model parameters: $\lambda = 2$.

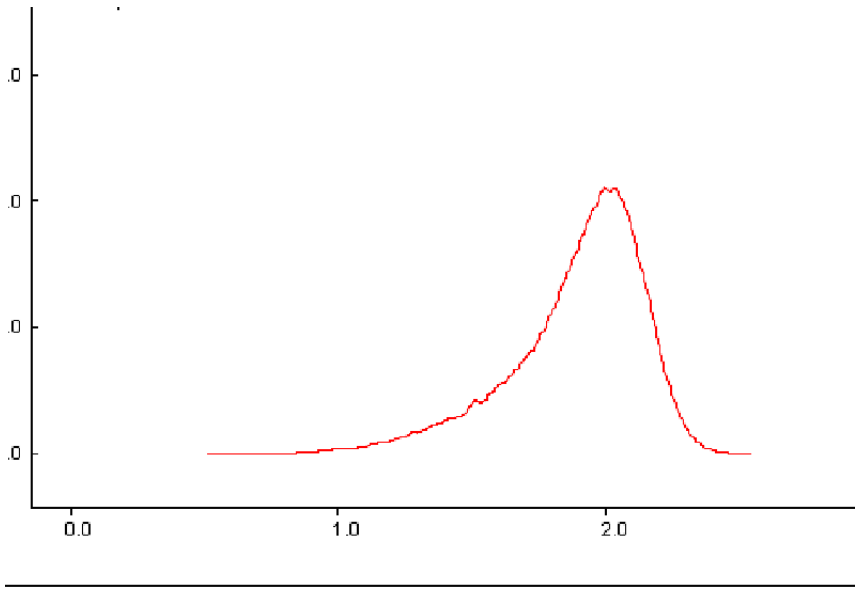

Figure 6.11: Posterior distribution of beta for regression tree size, $\lambda = 2$

Node	Mean	SD	MC Error	2.5%	Median	97.5%
α	3.443	0.4245	0.001307	2.602*	3.443*	4.284*
β	-0.6772	0.6009	0.001933	-1.87	-0.6791	0.512
σ^2	0.623	0.2081	6.871E-4	0.2865*	0.5992*	1.093*

Table 6.8: Parameter estimates, accuracy model ($\lambda = 10$).

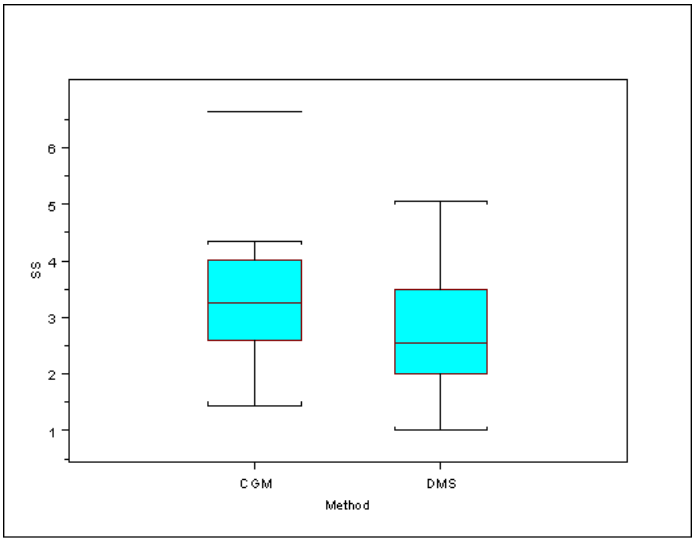

Figure 6.12: Regression Sum of Squares, $\lambda = 10$

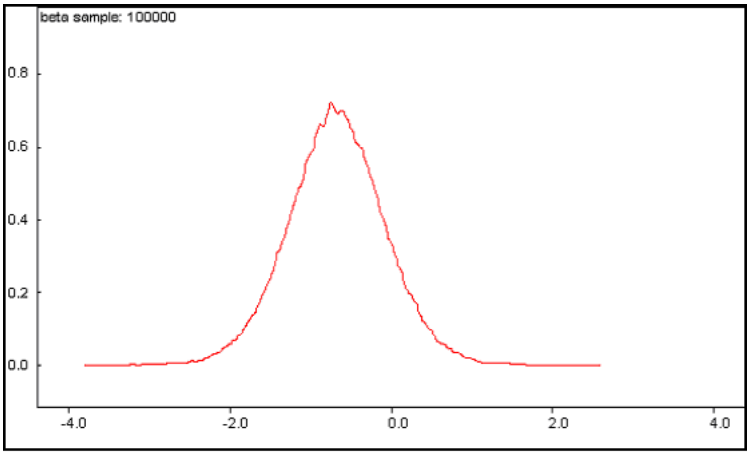

Figure 6.13: : Posterior for beta, Tree accuracy model, $\lambda = 10$

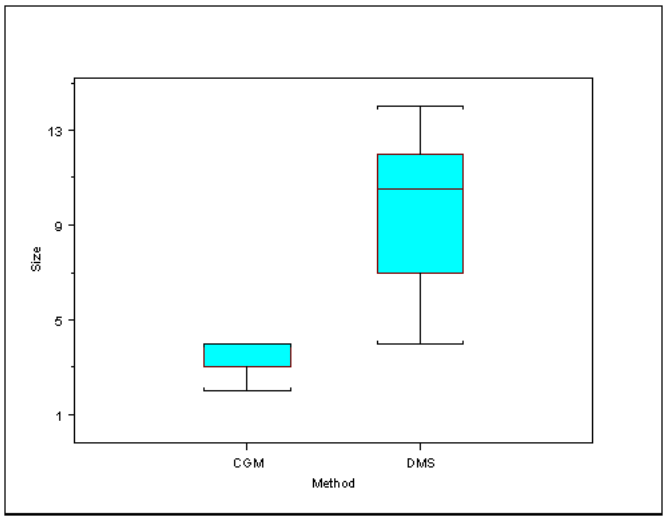

Figure 6.14: : Tree size, $\lambda = 10$

Node	Mean	SD	MC Error	2.5%	Median	97.5%
α	1.149	0.1775	0.001582	0.7883*	1.152*	1.485*
β	1.108	0.2048	0.00184	0.7132*	1.105*	1.515*

Table 6.9: Parameter estimates, tree size model ($\lambda = 10$).

Of 100 000 values for beta, 12 424 are positive. We conclude that beta is not significantly different from zero, implying that there is no firm evidence of a difference in accuracy between the two methods. As for tree size, we see (Figure 6.14 that DMS trees are much larger than CGM trees. The formal model is as before. Convergence has occurred for both alpha and beta.

We see that beta is significantly different from zero, which is what we would have expected from the stark difference apparent in Figure 6.14 All 100 000 values in the posterior for beta were positive, which is strong evidence of a difference between the two methods.

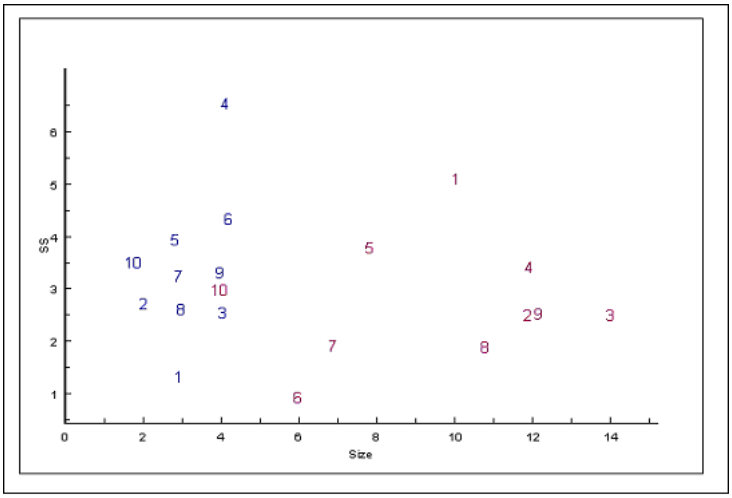

Figure 6.15: : Regression Tree Size vs. Error Sum of Squares. Colour denotes model. (Blue = CGM, Red =DMS) Numbers denote cross-validation folds., $\lambda = 10$

6.2.4 Speed of execution of regression trees

The regression tree work has been carried out some 2 years after the work on classification trees. We have moved from using a Sun Sparc workstation running Solaris to a 1.0 GHz Dell Inspiron 4100 laptop, running Windows 2000 Professional. The increase in hardware speeds in the intervening period has been substantial. The 200 000 iteration regression tree runs are now produced in a few seconds. Accurate timings were impossible to obtain due to lack of operating system facilities for timing programs.

6.3 Summary of Results

For classification, on the breast cancer data set,

- The differences in tree size are not significant

- The differences in tree accuracy are not significant

For regression, on the air data set,

- DMS trees are significantly larger than CGM trees, for both l = 2 and l = 10
- DMS trees are significantly more accurate (but only just) for l = 2 but not for l = 10

The CGM classification algorithm is faster on the breast cancer data set. We have no timing data for the regression models.

6.4 Discussion of Results

We may wonder how the differences in accuracy, tree size and speed come about. It is difficult to be definitive, as the CGM and DMS codes define their priors differently, make use of different move types and are written in different programming languages. The computer programs are reasonably well written (very well written in the case of CGM) and do not seem to have any obvious inefficiencies. There should not be great differences in speed between the C++ code used by CGM and the C code used by DMS, particularly as we have used the same compiler (GNU gcc) for each. (This is possible since C is a subset of C++.) A key difference is that CGM restart their Metropolis-Hastings chain several times (10 in our tests) and they claim that this leads to a better search. DMS claim that "Chipman et al require many restarts of their algorithm, resulting in a very much greater computational burden." (Denison et al , 1998)[38]. This claim is not supported by our experimental work. The CGM approach certainly works well for classification problems, producing equally good trees

in less time. This result may have implications for other MCMC-based models. It may be that multiple restarts is generally a better strategy than a single long chain. For regression problems the DMS trees were always larger. These larger DMS trees were more accurate for some prior choices but not for others.

Chapter 7

The robustness of CART trees to outliers

7.1 Introduction

It is sometimes claimed that tree models are robust to outliers. For example, Breiman et al. (1984) [16]) claim that CART regression trees are fairly robust to outliers.

Tree structured regression is quite robust with respect to the measurement variables, but less so with respect to the response variable. As usual, with least squares regression, a few unusually high or low y values may have a large influence on the sum of squares. However, the tree structure may treat these outliers in a way that both minimises their effect and signals their presence. It does this by isolating the outliers in small nodes. If there are one or a few cases in a node whose y values differ significantly from the node mean, then a significant reduction in residual sum of squares can be derived from splitting these cases off.

Breiman et al, page 253

We conducted some experiments to test these assertions. We again used the air data

of Bruntz et al. (1974) [19] as our testbed. (Atmospheric ozone levels are predicted from knowledge of temperature (°F) and wind speed (miles per hour).

7.2 The base case

We first grew a CART tree using the CART routine supplied with the Splus software system (Insightful Corporation, [121]). The default settings were used (a minimum of 5 observations per leaf, deviance-based cost-complexity pruning). This was the base case with no outliers. This yields the tree in Fig. 7.1. This can be plotted with temperature on the x-axis and wind speed on the $y$axis to show the partition corresponding to the tree. However, some of the data points in predictor variable space are identical to one another. The points {(68, 11.5),(76, 7.4),(76, 10.3),(78,10.3),(81 ,6.9),(81,9.2),(82,8)} appear twice and the point (81,14.9) appears three times.

7.3 Outliers in the y variable

7.3.1 Experiment One

Our first experiment was to add in a single outlier by doubling the y value at the minimum temperature. This leads to the new tree in Fig. 7.3 and the new data plot in Fig. 7.4. We notice that the data plot does not change much but the tree has changed noticeably. The second-level split $\mathsf{Temp} < 77.5$ has been swapped with the third-level split $\mathsf{Windspeed} < 7.15$. That is quite a noticeable change at the top of the tree, where the early splits affect all subsequent splits. Something similar happens when we introduce an outlier at the maximum temperature: we obtain the tree shown in

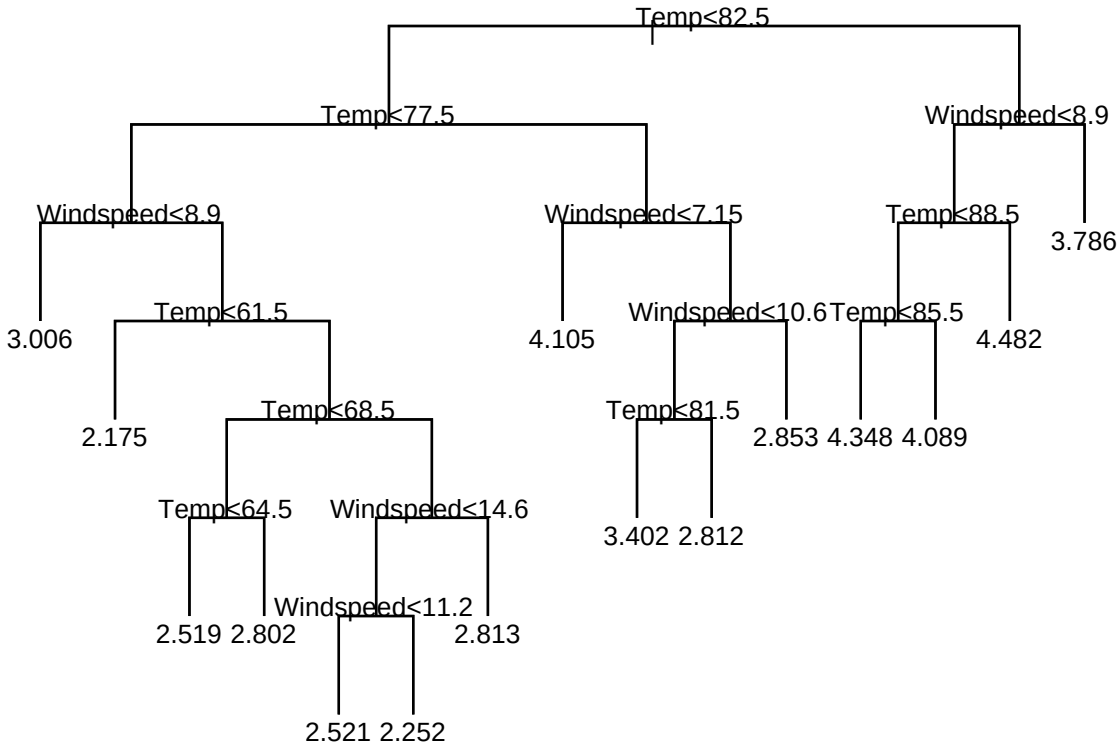

Figure 7.1: Base case tree. No outliers.

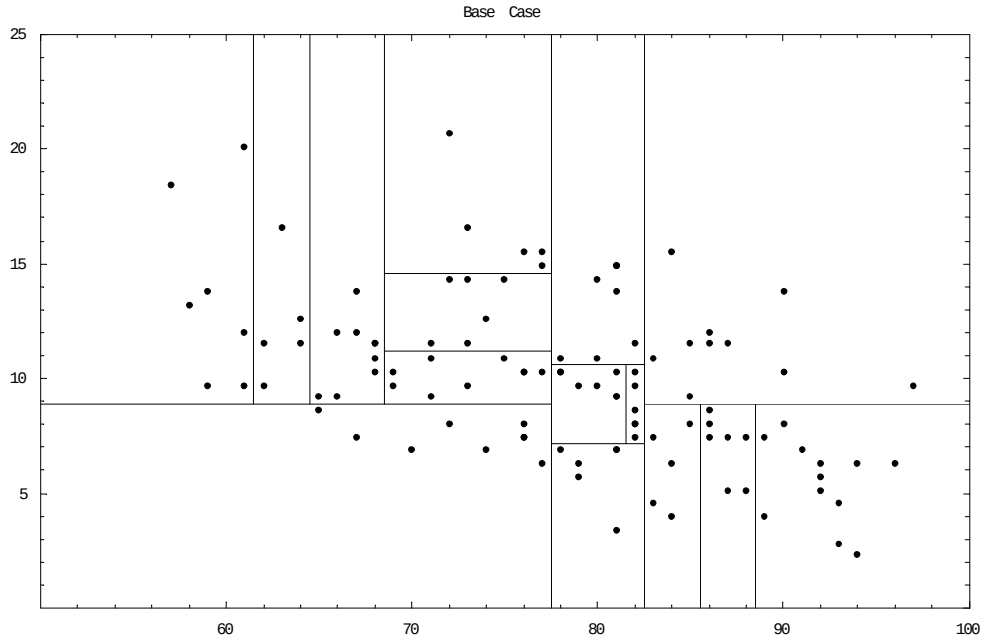

Figure 7.2: Base case: x-axis Temperature (°F), y-axis Wind speed (mph).

Fig. 7.5. Again, the tree has been changed, but the resulting partition is not greatly affected.

7.3.2 Experiment Two

If we now put in 6 outliers, all with a doubling of the y value, at the minima, medians and maxima of both predictive variables we produce the tree in Fig. 7.7 and the corresponding partition is Fig. 7.8. With only 6 outliers, none of them very severe, the tree structure has been substantially altered and so has the induced partition. Six outliers in 111 observations is only 5% of the data. The tree building mechanism has not split off the outliers as promised, because there is a minimum of 5 observations per leaf. In the base case there were 15 leaves and the number of observations per leaf was $\{7, 5, 9, 5, 6, 9, 8, 7, 6, 10, 9, 6, 5, 7, 12\}$.The mean number of observations per leaf was

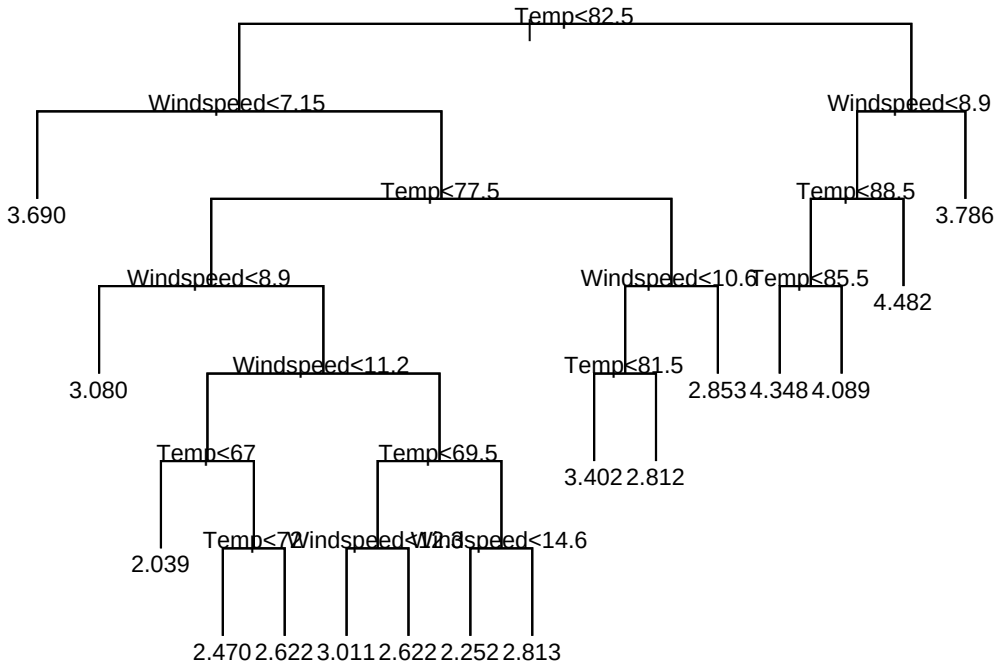

Figure 7.3: Outlier at minimum temperature.

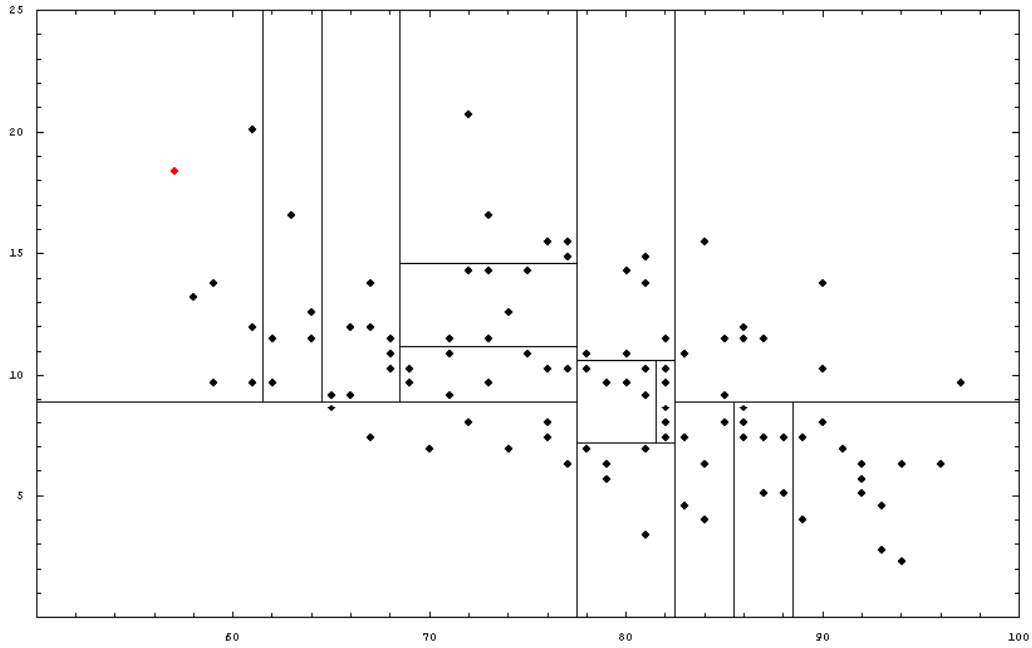

Figure 7.4: Outlier (in red) at minimum temperature.

7.07 and the standard deviation was 2.84. In the 6-outlier case there were 13 leaves and the number of observations per leaf was $\{7, 5, 9, 17, 6, 6, 16, 14, 5, 5, 10, 6, 5\}$. The mean number of observations per leaf was 8.54 and the standard deviation was 19.27. Of the 6 outliers, only two of them are in leaves of the minimal size (5 observations). We could reduce the minimum number of observations per node, but this would have consequences for the base case tree as well as trees with outliers.

7.3.3 Experiment Three

We now put in 12 outliers, which is 11% of the total number of observations. This leads to the tree given in Fig.7.9 which is substantially different to the base case tree.

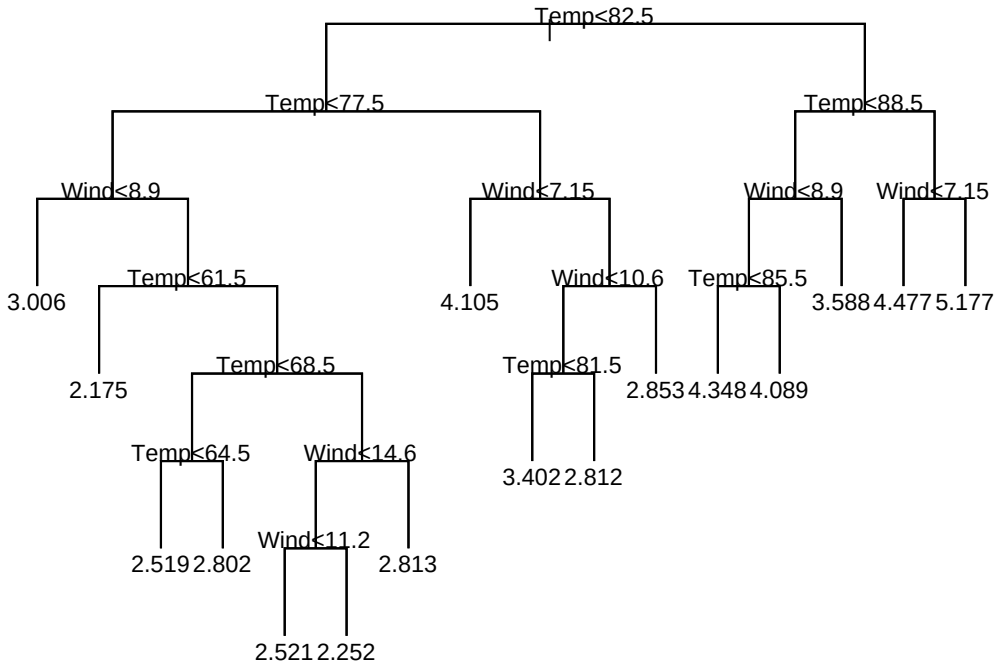

Figure 7.5: Outlier at maximum temperature.

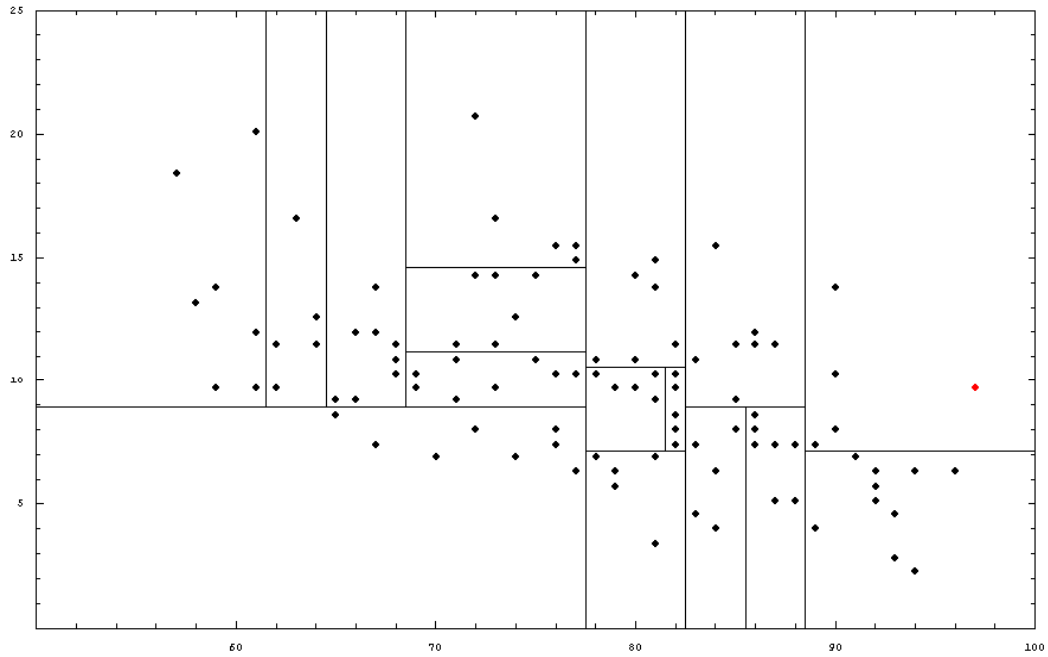

Figure 7.6: Outlier at maximum temperature.

The resulting partition now looks dramatically different from the base case.

7.3.4 Conclusions

We conclude that CART tree models are not particularly robust with respect to outliers in the response variable. In practical tree models, such as where leaves have a minimum of 5 observations, the tree growing mechanism does not necessarily split off the outliers into small leaves, as Breiman stated. Instead, the leaf boundaries are moved substantially, which will reduce predictive accuracy.

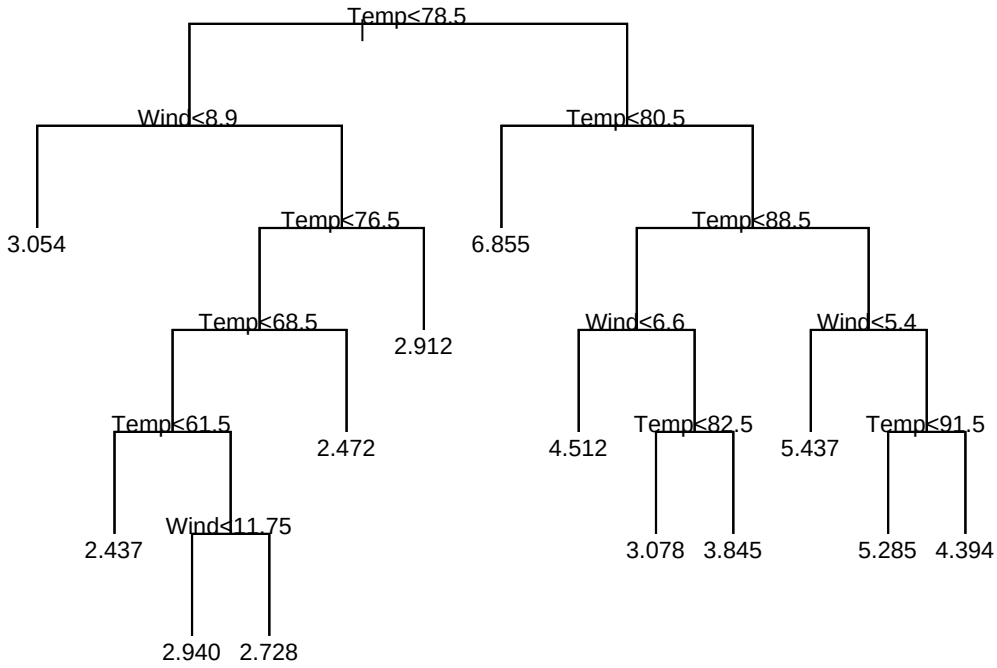

Figure 7.7: 6 outliers.

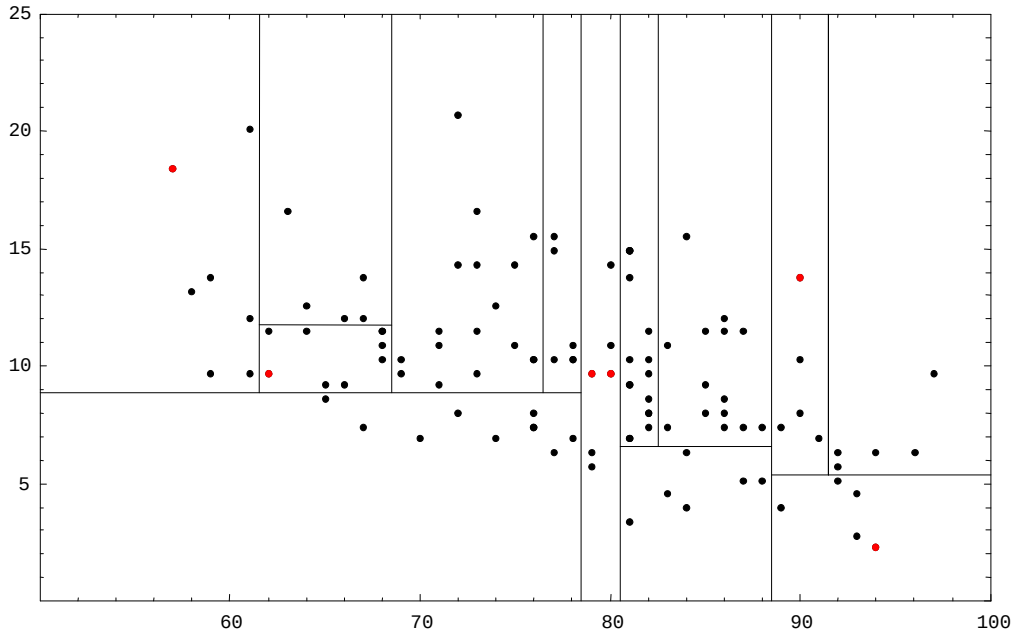

Figure 7.8: 6 outliers.

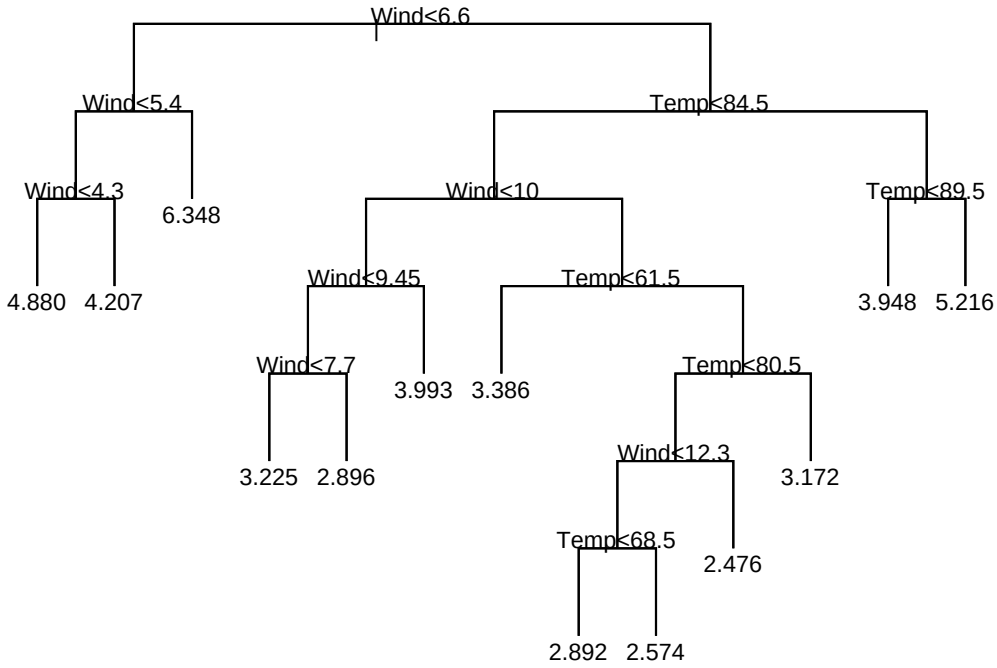

Figure 7.9: 12 outliers.

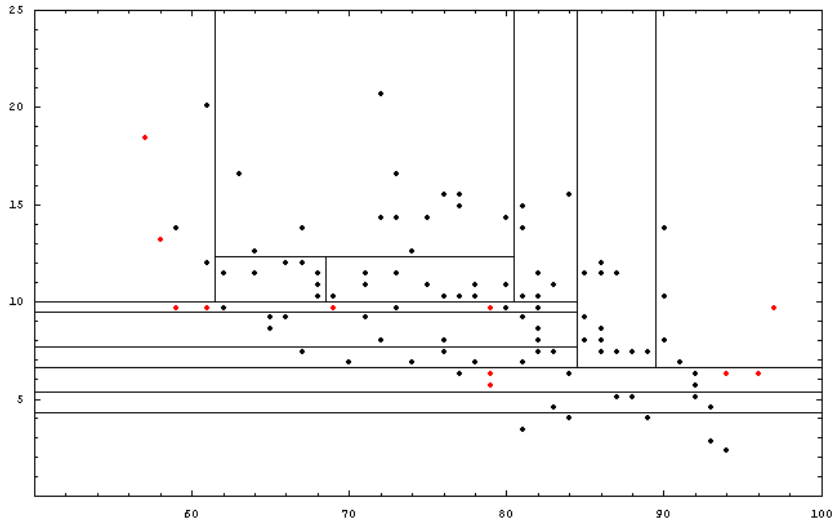

Figure 7.10: 12 outliers.

Chapter 8

Optimal Training Set Size

8.1 Introduction

In order to use tree models the user must have a training set: that is, a set of Y (response) values which must be predicted from a corresponding set of one or more X (predictor) values. The X and Y quantities may be real-valued, interval or ratio-scale numbers or class labels. Many of the large data sets used in practical knowledge discovery applications have hundreds of thousands, or even millions, of rows from which a training set can be selected. In this chapter we will consider the problem of choosing the size of the training set. In order to do this we will model the accuracy of decision trees produced by training on increasingly large training sets. For this we use a well known dataset called **shuttle**. We then construct a loss function encompassing both training time and misclassification costs, and we derive the training set size which leads to minimum loss. A brief discussion of previous work in this area follows. We also consider the problem of optimally augmenting the training set if is found to be too small, using the methods of decision theory.

8.2 Training Set Size

The author has on several occasions observed practitioners using the entire set of available training data as input to a tree constructing algorithm, even when this data set amounted millions of records of data and requires very substantial processing time. This motivated his consideration of how large a training set is actually necessary for building tree models. There are several issues at stake:

1. A tree algorithm partitions the data into supposedly homogeneous groups. Use of too small a training set may mean that some of the resulting groups have few observations. There is then a risk that a group which is actually not homogeneous is not split because there is insufficient statistical evidence on which to justify a split. Increasing the training set will tend to increase the number of observations in this group and this may result in one or more additional splits.

2. Tree models are subject to large model uncertainty errors. Given this fact, it is unwise to focus a great deal of computational effort on elaborating a *single* tree in great depth. As far as predictive accuracy is concerned, it is far better to explore a number of trees in moderate depth and combine them by model averaging in order to reduce the model uncertainty.

3. Training an algorithm takes time and this must be regarded as a cost. Empirical work, both ours and others (Oates and Jensen, 1997), suggests that accuracy does not continue to increase indefinitely as the training set size increases. If one considers ever larger training sets there must come a point when the cost exceeds the value of the information obtained. Somewhere before this there must be an optimum training set size which delivers enough accuracy for our purpose, yet

does not take any longer than necessary. The search for this optimum is the subject of this paper.

4. There is empirical evidence that large training sets lead to large trees (Oates and Jensen, 1997, [104]). This rises problems with how their use: with small trees, domain specialists can examine the tree and see if it accords with their experience and knowledge, whereas with large trees this is not possible, so large trees are difficult to interpret. If the purpose of creating the tree is to guide future action, the information contained in a large tree may be more than can reasonably be put into action and is therefore wasted effort.

5. If further data analysis is to take place, computationally intensive methods (e.g. modern nonparametric methods (Hastie and Tibshirani, 1990, [69]) could be used on a small sample, but this would not be possible with the entire data set because of the computational burden.

6. Use of the entire data set for training purposes leaves no data to evaluate the accuracy of the method, since one needs data unseen by the training algorithm for an honest assessment.

8.3 The Shuttle Data Set

The data set is in the University of California at Irvine Machine Learning archive (UCI archive, [130]) and was supplied by Jason Catlett of the Basser Department of Computer Science, University of Sydney, New South Wales, Australia. The data was collected by NASA and concerns the position of radiators within the Space Shuttle. The data set is already divided up for the user into a training set of 43 500 and a

Class	Name	Train	Test
1	Rad flow	34108 (78.41%)	11478 (79.16%)
2	Fpv Close	37 (0.09%)	13 (0.09%)
3	Fpv Open	132 (0.30%)	39 (0.27%)
4	High	6748 (15.51%)	2155 (14.86%)
5	Bypass	2458 (5.65%)	809 (5.58%)
6	Bpv Close	6 (0.01%)	4 (0.03%)
7	Bpv Open	11 (0.03%)	2 (0.01%)

Table 8.1: Shuttle classes.

test set of 14 500 observations. All of the 9 predictor variables are numeric and the response variable is a class label with 7 different values. Approximately 80% of the observation are class 1. There are no missing values. There appears to be little noise in the data set, as very high classification accuracies are achievable.

8.4 Modelling The Error Rate of a Decision Tree

In order to explore the relationship between training set size and the error rate of the resulting tree we ran an experiment. A large well-known data set was chosen (shuttle) and the training set was randomly partitioned into 20 approximately equal-sized pieces, labelled Block 1 to Block 20. A decision tree was grown using Block 1 as the training set. The tree method used was the Classification and Regression Tree facility in the commercial statistics package Splus (Insightful Corporation, Seattle, USA,[121]). The algorithm used is a variant of the CART algorithm. This tree was applied to a separate test set and the number of misclassifications was recorded. A new training set was then constructed as the union of Blocks 1 and 2, and a second decision tree was grown on this larger training set. Again, the tree was applied to

the test set and the number of misclassifications recorded. This process was repeated using larger and larger training sets until the entire available training data set was used. This sequence of runs is called Sequence 1. At this point the random number seed was changed and a second partitioning of the training set was carried out. A sequence of tree-growing runs was conducted on this re-arrangement of the data, and this is Sequence 2. Finally, the random number seed was changed again and a third sequence of tree-growing runs was done. At the end of this process we have a sample of three observations for each size of training set considered. Each observation represents the number of misclassification errors of a tree grown with a training set of that size. We are thus in a position to model how error rates vary with training set size. The graph of misclassification errors against training set size for the three sequences is shown in Fig. 8.1 From visual inspection of Fig. 8.1 we see that

1. Each sequence behaves rather erratically, making it difficult for any smooth model to fit the data well

2. The sequences are noticeably different to one another

3. c) The sequences are not monotone, i.e. it is possible to add more data and obtain a worse misclassification than before.

We modelled the number of misclassification errors by a Poisson distribution with parameter λ_i. We wanted to model λ_i as a smooth, decreasing function of the training set size n_i so, after exploring a range of models, we arrived at the following model:

$$\begin{aligned} \lambda_i &= \alpha_i + \beta \log(n_i) + u_i \\ u_i &= \rho u_{i-1} + \varepsilon \end{aligned} \tag{8.4.1}$$

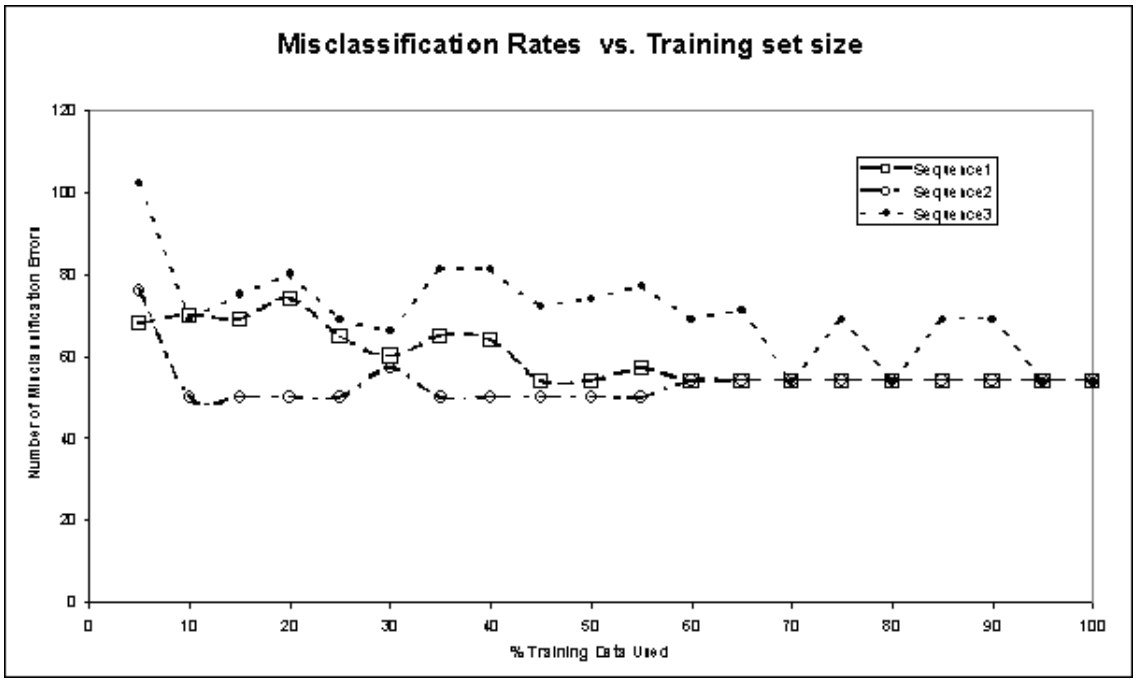

Figure 8.1: : Misclassification errors for the three sequences, versus training set size.

where n_i is the size of the training set and α and β are constants to be estimated from the data. The u_i term is an autocorrelated error term with correlation coefficient ρ. This is necessary because the λ values are correlated, being mostly based on the same data. The model outputs are probability distributions for α and β, not just point estimates with confidence intervals. These probability distributions represent our uncertainty about the true values of α and β after we have seen this data. This methodology enables the uncertainty about α and β to be properly carried through into a subsequent decision analysis. The parameters were given vague but proper prior distributions:

$$\begin{aligned} \alpha_i &= N(0, \sigma_i^2) \\ \sigma_i^2 &= \text{Inverse-Gamma}(10^{-3}, 10^{-3}) \\ \beta_i &= N(0, 10^{-4}) \\ \rho &= \text{Uniform}[-1, +1] \end{aligned}$$

The variability of the data clearly is not constant, so the model allows different values of α to have different variances. The model was fitted using Markov Chain Monte Carlo (Gilks et al. 1996) methods in a Bayesian modelling environment (WinBUGS,see Spiegelhalter et al., 1997) There are 20 posterior distributions for the 20 different values of λ . The first two values are shown along with the last two (see Fig. 8.2) Fig. 8.3 shows the data from Fig. 8.1 but also the estimated curve of λ versus training set size, along with 95% credible intervals for the observed data. Note that these are not credible intervals for the estimate of λ, but intervals such that, with probability 0.95, the *data* should lie if the model is true. The λ curve tells us the

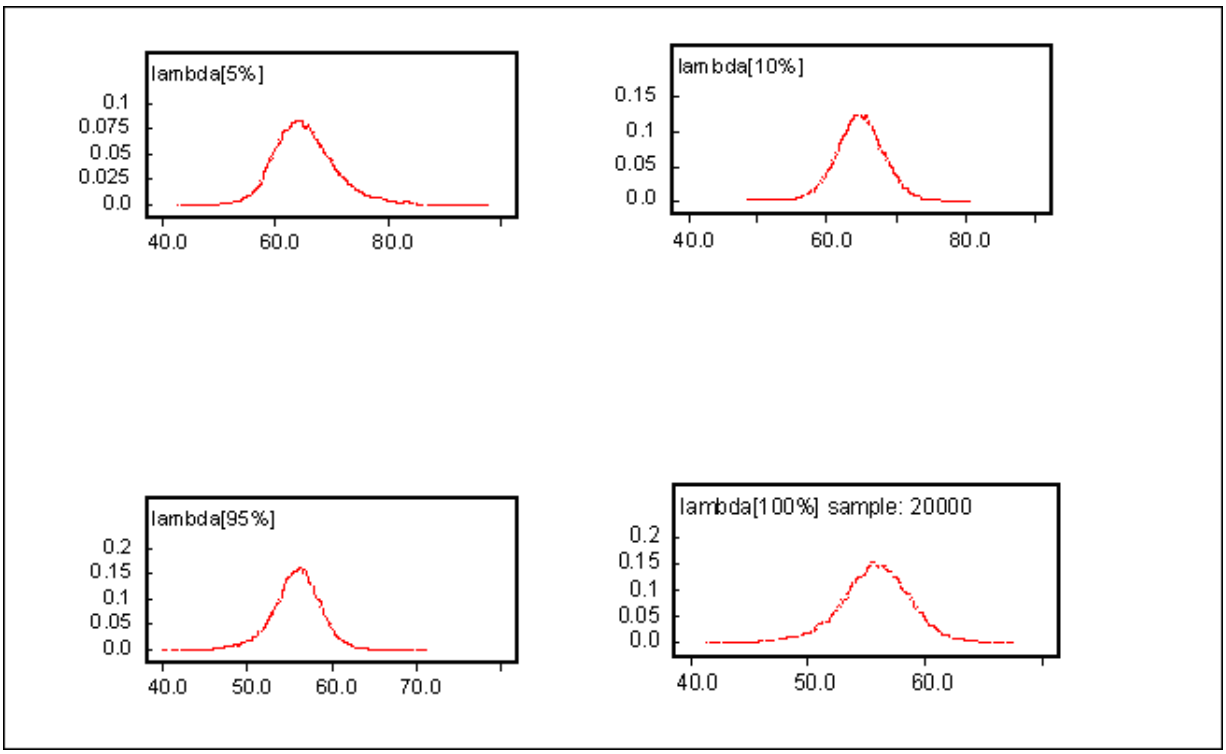

Figure 8.2: : The first two and last two posteriors for lambda.

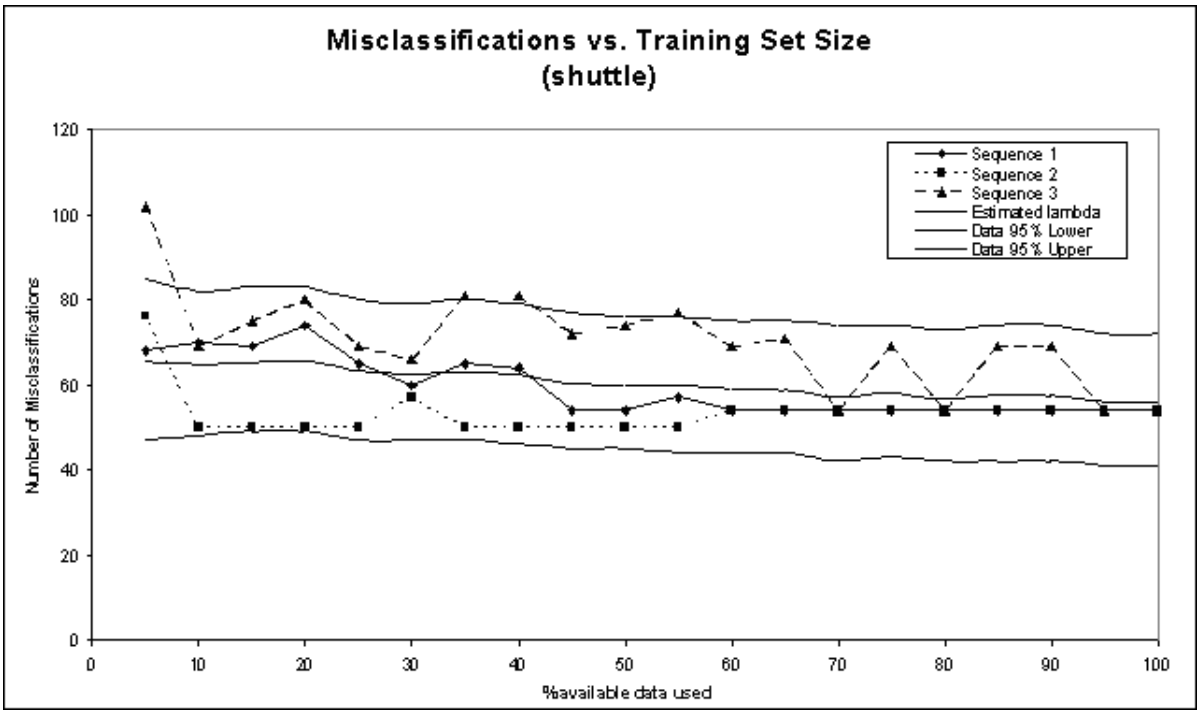

Figure 8.3: : Misclassification errors for the three sequences, plus estimated lambda curve and 95% credible intervals for the observed data.

expected number of misclassification errors for a given size of training set, assuming our model is true. The number of errors starts high then drops sharply and levels off, with increasing sample size adding little in the way of predictive accuracy. After we saw the graph of the loss function we realised that it was important to focus on the early part of the curve, where from 1% to 10% of the available training set was used. Thus the entire process was repeated, this time using 1% increments rather than 5% increments, up to a maximum of 10%.

8.5 Model Fit and Sensitivity Analysis

A number of different models were tried and the one chosen was the best fitting of those tried. Nevertheless, there is still some evidence of lack of fit and we do not regard this model as definitive for this data set. From Fig. 8.3 we see that 4 out the 60 data points, i.e. 7%, are outside the 95% confidence limits, instead of the nominal 5%. All four of these outliers are from sequence 3. The R^2 statistic for the model is a rather poor 0.103, indicating that the model explains only about 10% of the total variability in the data. This comes about for two reasons. Firstly, the unexplained variation is high. Most of this, about 60%, is attributable to Sequence 3, which is much more variable than the other two. Secondly, the explained variation (the variation of λ about the overall mean) is low because λ is not departing much from the mean most of the time. For the second set of runs, with training set sizes in the range 1% to 10%, the R^2 statistic was a much improved 0.398. The unexplained variation was still high, but this time the predicted λ departs sharply from the overall mean in places, thus boosting the explained variation. The 95% confidence intervals for the observed data were obtained from the posterior distributions of the λ parameters. For

example, visual inspection of the posterior distribution of $\lambda[1]$ (Fig. 8.2) shows it to be approximately normal. The posterior mean has been estimated as 65.39 and the posterior standard deviation as 5.4930. We drew 100 000 random deviates from this normal distribution and, for each one of them, we derived 100 000 Poisson deviates with this value of λ. We then took the 2.5 and 97.5 percentiles of these Poisson deviates as our 95% confidence intervals for the observed data. This was done for all 20 λ values. The posterior mean of the λ values is plotted on the graph in Fig. 8.3 along with the confidence intervals for the observed data, and the observed data itself. The assumed model implies that λ tends to infinity as n tends to infinity. (α is found to be negative and β is positive.) We would prefer a model which implies that λ tends to a finite asymptote as n tends to infinity. A number of such models were explored. Additional model parameters were added to capture the preferred asymptotic structure but the posteriors obtained were the same as the priors put in, indicating that the data in this data set provides no information regarding these additional parameters. Accordingly, the extra parameters have been dropped. A quadratic model was also tried, but the resulting posterior distribution for the corresponding coefficient had substantial probability mass either side of the origin, indicating that the data were consistent with a value of zero. As we stated earlier, we do not consider our chosen model to be definitive, but it fits well enough for the purpose at hand. The conclusions arising from its use are particularly clear-cut and we do not think it likely that small increases in fit would change these conclusions to any significant degree.

8.6 Loss Functions

We model the impact on the user of training sets of different sizes by means of a loss function. There are two components to our loss function. The first part represents the cost of training. (Note that the cost of collecting the data is considered a sunk cost and is specifically excluded from these calculations.) The training time is proportional to $n\log(n)$, as training times for trees are dominated by the time required to sort the records, which for an efficient method is proportional to $n\log(n)$ (Kronsjö, 1979)[87]. It is worth noting that in practice, the cost of the training time is often more complex than this. For example, if a tree may be trained in a 2 hour slot in an overnight batch run the tree may be used the next day. If the training job takes longer than 2 hours to run it may have to be done in pieces over two or more nights. This additional delay before the tree is available could have serious commercial consequences. The second component of the loss function represents the cost of a classification error. We assign a 1 for each classification error and a 0 for each correct classification and represent the loss by the square of the number of errors, multiplied by a constant. Our loss function is thus

$$L(n) \quad = \quad c_0 n\log(n) + c_1\mathbb{E}^2(n) \tag{8.6.1}$$

where $\mathbb{E}(n)$ is the expected number of errors produced by a tree trained on a training set of size n. We set $c_0 = 10$ and $c_1 = 1000$ i.e. the additional cost of a classification error is substantially higher than the cost of processing an additional row in the training set. The optimal decision is the one which minimises the expected value of the loss function. In this case the optimal training set size is the value of n

Figure 8.4: : Expected loss versus training set size.

which minimises the expected value of L. The chart of expected loss versus the used percentage of the training set is shown in Fig. 8.4 The main message is that training set sizes around 4000 or 10% are optimum for this data set. In order to explore this region more fully a second set of runs was carried out with training set sizes in the range 1% to 10% in 1% increments. (Fig. 8.4 is the combination of these two loss function charts.) The expected loss increases to a high value until we are using about 3% of the training data (about 1300 observations) and then declines to a minimum when about 10% of the available data is used (i.e. about 4000 observations). After that, the loss slowly (and somewhat unsteadily) increases again. Thus our advice to

a user whose circumstances were adequately modelled by our choice of loss function would be to use a sample of about 4000 for their tree building. Note that in sampling theory the accuracy depends on the absolute number in the sample, not the percentage of the population used, so it is the 4000 that matters, not the figure of 10%.

8.7 Previous Work on Sample size and Accuracy

Generally, (but as we have seen above, not always) using a smaller training set size will lead to more misclassifications than would be the case with a large training set size. Catlett [22] provides empirical evidence that this is so for a number of data sets. He compared trees grown from a training data set with those grown from a 50% simple random sample of the training data. A statistical hypothesis test (a t-test) showed that the results from the 50% sample were worse in terms of classification accuracy than those from the full data set and the differences were significant at the conventional 5% significance level. Catlett's work has been drawn to our attention as proof that sampling is generally, or usually, unjustified in knowledge discovery work. We believe that this is an invalid conclusion from Catlett's work, for several reasons.

1. If a decision is to be taken, such as whether to sample or not (or more generally, what sample size to choose) significance tests are inappropriate. Formally, significance tests are inadmissible for decision-making and the methods of decision theory should be used instead.

2. Significance tests are intended to rule out the effects of sampling error. However, sample sizes in used for tree models in practice are often such that sampling error is virtually non-existent and thus significance tests may give completely

> spurious results. When very large sample sizes are used the dominant error is likely to be a small bias in the design. Such a bias might occur, for example, if the number of measurements to be collected was not specified in advance. In such a situation there is a very real risk that bias could cause the null hypothesis to be wrongly rejected. See Jefferys (1990) [81] for an example of this.

In fact, Catlett does not claim that sampling is *generally* inappropriate. As he points out:

Whether this [difference in accuracy] is large enough to be significant to a client obviously depends on the application.

Oates and Jensen investigated the accuracy of C4.5 tree models versus training set size as part of an investigation into the relationship between training set size and the size of the resulting trees. They tested 19 data sets from the UCI Machine Learning archive. The data sets used were much smaller than the shuttle data set used here (typically a few hundred observations, the largest being **australian** with 690 observations). No attempt was made to model the misclassification rate or derive optimal training set sizes. A simple method was used to determine the point at which accuracy (percentage of correct classifications) levelled off: if three successive values in the sequence had error rates within 1% of the error rate of the full training set the accuracy was assumed to have levelled off. For 16 of the 19 data sets the accuracy levelled off before 100% of the available training data was used. In 13 of the 19 data sets the accuracy levelled off when 50% or less of the available data was used for training. The only data set which needed 100% of the training data was **tic-tac-toe**, which is a noise-free data set. We consider the Oates and Jensen results consistent with, and supportive of, our own.

8.8 Optimal Augmentation of the Training Data

If the training data set one has is too small and has to be increased, how should this be done? In the experiments considered so far we have used simple random samples, but in practice some parts of the data set will be more important than others. For example, it may be more important to increase the number of observations close to decision boundaries. In statistical terms, the recursive partitioning algorithm stratifies the data, that is, divides up the space into subsets of observations which are homogeneous, the leaves of the tree. The principles of stratification, as used by survey statisticians to decide upon the number of strata in a survey Dalenius, 1957 [32]), can also be applied to the choice of the number of leaves for a decision tree. For a numeric response variable the within-leaf variances should be substantially lower than the total (within-leaf plus between-leaf variances) and the leaf means should be well separated. Thus when augmenting the training set we would be most interested in obtaining more data for leaves which currently have a high variance. For a categorical response variable we would ideally want all the observations at a leaf to have the same class label, so we would want to augment 'impure' leaves which have a mixture of class labels. Formally, the problem of deciding which leaf to augment with additional data can be regarded as a decision problem to be solved by the universal principle of maximising expected utility (Lindley, 1991, [94]]. We should calculate the expected extra utility which would result from adding one observation to each leaf. (Utility is the negative of loss.) The leaf for which this expected additional utility is greatest is the one which we should actually augment by one observation. We should then repeat the process, adding one observation each time to the leaf which most needs it. (In practice we might add more than one observation at a time!) In order to do this we

model the data at a leaf as arising from some probability distribution of known type but with unknown parameters. For a numeric response we might assume a normal with unknown mean and variance, whereas for a categorical response we could use a multinomial distribution. In any case, there is a vector q of unknown parameters to be estimated from the data. We can apply the data at the leaf to a vague prior $p(\theta)$, and use Bayes' theorem to obtain a posterior distribution $p(\theta|x)$, where x represents the current data at the leaf. We can then treat this posterior distribution as a prior distribution for the thought experiment of adding the extra observation x'. This calculation leads to another posterior $p(\theta|x, x')$ which incorporates the additional knowledge that we would hope to gain from the additional observation x'. The value of this additional knowledge is expressed via a utility function $u(p(\theta|x, x'), \theta))$. In a pure inference problem (i.e. where there are no misclassification costs involved) it can be shown (Bernardo and Smith, 1997) [10] that the appropriate utility function for a probability distribution is a logarithmic score function:

$$u(p(\theta|x, x'), \theta) = A \log(p(\theta|x, x')) + B(\theta)$$

where $A > 0$ is an arbitrary constant and $B(\cdot)$ is an arbitrary function of θ. Using this score function it can be shown (Bernardo and Smith, 1997, [10]) show that the expected increase in utility from the additional observation x' is given by

$$A \int p(\theta|x, x') \log \left(\frac{p(\theta|x, x')}{p(\theta|x)} \right) d\theta$$

This formula can be interpreted in several ways. On the one hand it is the expected increase in utility obtained from the extra observation. A second interpretation is obtained by noting that it is also the Kullback-Leibler divergence between the prior and the posterior distributions describing the data. A third interpretation is possible if

we use logarithms to the base 2. The above formula then corresponds to the Shannon information of the data (Shannon, 1948)[118]. Maximising expected Shannon information is thus a special case of maximising expected utility. (Note that if we know that different types of misclassification have different costs the logarithmic score function is no longer appropriate, and hence neither is maximising expected Shannon information.) As an example, consider a two-class classification problem where there are 7 observations at a particular leaf. Five of the observations are of class 1 and two are of class 2. We use a uniform distribution as our prior and make use of the fact that the uniform is a special case of the beta distribution, which is conjugate for the binomial likelihood appropriate for a two-class problem. Conjugacy means that our posterior will be another beta distribution. The likelihood is of the form

$$p(y|\theta) \propto \theta^5(1-\theta)^2$$

and the prior is

$$p(\theta) \propto \theta^{\alpha-1}(1-\theta)^{\beta-1}$$

i.e. $\theta \sim Beta(\alpha, \beta)$. The values $\alpha = \beta = 1$ give a uniform distribution, so these are what we will use. The posterior is given by

$$\begin{aligned} p(y|\theta) &\propto \theta^5(1-\theta)^2\theta^{\alpha-1}(1-\theta)^{\beta-1} \\ &= \theta^{5+\alpha-1}(1-\theta)^{2+\beta-1} \\ &= \text{Beta}(\theta|6,3) \end{aligned}$$

This posterior now becomes our prior for the imaginary experiment of adding another observation. The addition of an additional single observation will either change this to a Beta$(\theta|7,3)$ or a Beta$(\theta|6,4)$ depending on whether the new observation is in

class 1 or class 2 respectively. In the former case the utility increase (or information gained, if you prefer) is given by

$$\begin{aligned} & \int_{\theta=0}^{\theta=1} \frac{\Gamma(10)}{\Gamma(7)\Gamma(3)}\theta^6(1-\theta)^2 \log\left(\frac{\frac{\Gamma(10)}{\Gamma(7)\Gamma(3)}\theta^6(1-\theta)^2}{\frac{\Gamma(9)}{\Gamma(6)\Gamma(3)}\theta^5(1-\theta)^2}\right) d\theta \\ = & \quad 252 \int_{\theta=0}^{\theta=1} \theta^6(1-\theta)^2 \log\left(\tfrac{3\theta}{2}\right) d\theta \\ = & \quad 3.82268800665 \times 10^{-2} = I_1 \end{aligned}$$

In the latter case the utility increase is

$$\begin{aligned} & \int_{\theta=0}^{\theta=1} \frac{\Gamma(10)}{\Gamma(6)\Gamma(4)}\theta^5(1-\theta)^3 \log\left(\frac{\frac{\Gamma(10)}{\Gamma(6)\Gamma4)}\theta^5(1-\theta)^3}{\frac{\Gamma(9)}{\Gamma(6)\Gamma(3)}\theta^5(1-\theta)^2}\right) d\theta \\ = & \quad 504 \int_{\theta=0}^{\theta=1} \theta^5(1-\theta)^3 \log\{3(1-\theta)\} d\theta \\ = & \quad 0.148564938185 = I_2 \end{aligned}$$

As one might expect, there is more information to be obtained from observing the minority class. The probability of the observation being of class 1 is given by θ, while the probability of it being in class 2 is given by $1-\theta$. Thus the expected value of the utility increase caused by adding one observation at this leaf is given by $9.33959090832 \times 10^{-2}$ The above calculations should be done for each leaf node and the leaf with the largest utility increase is the one that should next be augmented.

8.9 Discussion

The methodology shown above provides a means of analysing a data set and deciding what would have been the optimum size of training set for a context described by a

particular loss function. Since the method involves trying out different training set sizes in order to build a model it does not directly help the user who wants to know the optimum training set size for his data and his problem context. Nevertheless, we hope that the principles and methodology exhibited here may lead to more such analyses being carried out, from which general guidelines may emerge. Our result for **shuttle** agrees with the work of Oates and Jensen on other data sets, suggesting that, in practice, many data sets will not require the full training data in order to achieve satisfactory results and that use of a sample is acceptable and preferable. We firmly believe that, appropriately applied, sampling methods are highly beneficial to the knowledge discovery process and we encourage their use.

In every branch of science we lack the resources to study more than a fragment of the phenomena which might advance our knowledge A sample is NOT a last resort, to be used when a complete investigation is impossible. Rather, it is the first resort: it is the answer to the question, "What is the best way to do the job?" or "What kind of sample will do it, and how big must it be to deliver the information that is required?"
Sample Design in Business Research. W Edwards Deming

Chapter 9

The future of Bayesian nonparametrics

9.1 A Brief background to nonparametrics

In the early part of the 20$^{\text{th}}$ century the two main tools of inference were classic hypothesis tests and linear models with normal errors. Hypothesis tests assumed normality or were nonparametric. Nonparametric hypothesis tests were developed to deal with cases where the normality assumptions clearly do not apply. The nonparametric tests made fewer distributional assumptions-such as simple symmetry arguments-and thus sacrificed power for robustness. The latter part of the 20$^{\text{th}}$ century was characterised by the escape from the limitations of the linear model with normal errors. The generalised linear model allowed a wider range of error terms, but still with a linear predictor. Classical statisticians in the 1960s and 70s pioneered data-driven approaches to model-building. Stepwise regression methods became popular for prediction and generalised cross-validation (Craven and Wahba, 1979)[31] became established as a fitting technique. Bayesians were slow to capitalise on this work, perhaps because the concept of 'distribution-free' models runs completely counter to

Bayesian thinking. To a Bayesian, the correct way to model uncertainty is by probability. To attempt to do inference without distributions is simply wrong-headed. Lindley (1982)[93] pointed the Bayesian community towards nonparametrics as an area that Bayesians could usefully develop. The key to Bayesian density estimation was to ignore the name 'non-parametric' and treat it as the exact opposite. A better name for the Bayesian approaches would be 'hyper-parametric'. Bayesian methods of density estimation apply Bayesian methods at a higher level of abstraction: a prior distribution is placed on a large space of candidate distributions, rich enough to approximate the true distribution to an acceptable degree. Conditioning on the data then leads to a posterior distribution of distributions. Similarly, Bayesian nonparametric model-building consists of putting a distribution on the model structure. One way of creating the prior distribution is the method used by Chipman, George and McCulloch: a stochastic process. The model search then becomes a stochastic search using Reversible Jump MCMC as the search algorithm, without necessarily reaching convergence. The use of a coherent Bayesian approach implies a natural penalty for model complexity. We now consider where are we on the road to having everything we would want in the area of Bayesian nonparametrics.

9.1.1 Model flexibility

We now have a wide range of nonlinear functions for building models, including multivariate linear splines, multivariate adaptive regression splines, natural cubic splines, smoothing splines, radial basis functions, neural networks, piecewise linear functions. For time series in the time domain there are autoregressive and moving average models (and their extensions, such as VAR). In the frequency domain several types of

wavelet bases are available: Haar wavelets, least asymmetric wavelets, Daubechies wavelets. The MATLAB® Wavelet Toolbox includes another 15 wavelet families. In short, we have just about all the model flexibility we need. No doubt there will be advances in model flexibility (new types of basis functions) but it is unlikely to make a dramatic difference to the art of model-building.

9.1.2 Computational issues

At the present time we have considerable computing resources available, but we could make good use of a lot more. It is still the case that models must be constrained to something that is computable. Large models in the areas of econometrics and environmental sciences currently may have hundreds or thousands of parameters and there is still cope for better understanding through larger models. There are two sides to the future of computation: hardware and algorithms. As far as hardware is concerned, there does not appear to be a consensus on the progress that can be expected. In 1964 Gordon Moore announced his famous 'law' that the number of transistors per square inch on processors chips was doubling every year. Since that time, Moore's law has become widely accepted as a given in the semiconductor industry. In reality Moore's law is not so well corroborated by data. Over the years its definition has been changed, for example to memory chips and other non-processor chips and the doubling time has been redefined (Tuomi, 2002)[129]. Nevertheless, despite the dubious status of Moore's law, it is clear that there has been rapid, sustained growth in the capabilities of semiconductor chips over many decades. It is also clear that the laws of physics imply constraints on what can be computed in a certain time (Lloyd, 2000, [97]). Long before these hard physical constraints begin

to bite major engineering challenges will probably be encountered, associated with working at ever-smaller scales of measurement. A further level of uncertainty surrounds the possibility of the quantum computer. Present-day computers are based on the idea of a Turing machine. The Turing machine serves as both an analogy for the Von Neumann design of sequential instruction computer, and also for the idea of an algorithm. Despite the enormous success of the Von Neumann-style computer, physicists have pointed out that the design of the Turing machine conflicts with the laws of quantum mechanics. The process of reading the tape is equivalent to a measurement and thus cannot be carried out perfectly. A redesign of the Turing machine has led to a new theory of computation which is still being worked out. In a classical computer, N bits of information represents 2^N integer values, each either 0 or 1. In a quantum computer, N qubits (quantum bits) can represent any complex vector of unit length in 2^N dimensions (Pike, 2000)[107]. This seems like a vast in the information content, but on attempting to read out the contents of such a vector the wave function collapses and only N bits of information result. Worse still, it does not even necessarily give the right answer. Different answers come out with different probabilities and one generally has to average a number of runs to obtain the right answer. Such a computer, if it can be constructed, would certainly have a very high degree of parallelism associated with its internal state during processing, and this leads to some intriguing possibilities. It might be able to explore all the possible paths in an algorithm simultaneously. For a CART tree method, this would enable it to explore all possible splits on all possible predictors, i.e. a complete search of the space of trees, even though this is vast. At the moment the quantum computer is not close to construction, but offers a tantalising future. It is not possible to say when or even if

it will appear, but substantial research resources are being devoted to it owing to its immense potential for breaking codes and ciphers.

9.1.3 Model complexity

If arbitrarily large models are possible then it is all the more important to make sure that overfitting does not occur. Bayesian model search methods contain a built-in penalty against overfitting, though this is not foolproof if the model class does not contain the true model (as is often the case) or if there are not enough data. In many fields where a great deal of data is available (e.g. geological data logging, naval sonar time series) Bayesian models might be considered irrelevant because there is sufficient data to swamp the effects of any reasonable prior. In fact, we believe that as the dimensionality of the data increases the importance of the prior increases. Use of frequentist methods is similar to the use of Bayesian methods with improper priors, and this causes certain kinds of problems. For example, computation of Bayes factors involves the use of priors, and Bayes factors do not work with improper priors if the models being compared have different dimensions. (Bernardo and Smith, 1994)[10]. Bayes factors are very helpful for model selection and search, since they involve a natural penalty term for model complexity (Kass and Raftery, 1995)[85] which has a strong justification from first principles. Bayes factors also appear in the transition kernel for Reversible Jump Monte Carlo Markov chain model search, and thus such model searches inherit the reluctance to overfit. High dimensional model searches are difficult enough without having to contend with overfitting problems. To summarise, model complexity may not be a major challenge in the future and the issue of overfitting can be overcome by appropriate use of coherent methods.

Bibliography

[1] J. Aitchison and I.R. Dunsmore, *Statistical Prediction Analysis*, Cambridge University Press, 1975.

[2] H. Akaike, *A new look at statistical model identification*, IEEE Trans. Automatic Control **19** (1974), 716–727.

[3] B. Amzal, F.Y. Bois, E. Parent, and C.P. Robert, *Bayesian optimal design via interacting MCMC*, Cahiers du Ceremade **0348** (2003).

[4] David R. Anderson, Kenneth P. Burnham, and William L. Thompson, *Null hypothesis testing: Problems, prevalence and an alternative*, J. Wildl. Manage. **64(4)** (2000).

[5] E. Anderson, *The Irises of the Gaspe Peninsula*, Bulletin of the American Iris Society **59** (1935), 2–5.

[6] Assael, *Segmenting markets by group purchasing behaviour: and application of the A.I.D. method*, J. Marketing Res. **7** (1970), 153–158.

[7] Crook J.N. Thomas L.C. Banasik, J.L., *Sample selection bias in credit scoring model.*, Journal of the Operational Research Society **54** (2003), 822–832.

[8] M. Bartlett, *A comment on D.V. Lindley's statistical paradox*, Biometrika **44** (1970), 533–534.

[9] D.M. Bates and D.G. Watts, *Nonlinear Regression Analysis and its Applications*, John Wiley, New York, 1988.

[10] J. Bernardo and A.F.M. Smith, *Bayesian Theory*, John Wiley, 1997.

[11] D. Biggs, B. De Ville, and E. Suen, *A method of choosing multiway partitions for classification and decision trees*, Journal of Applied Statistics **18/1** (1991), 49–62.

[12] G.E.P. Box and D.R. Cox, *An Analysis of Transformations*, J Roy. Statist. Soc., Series B **26** (1964), 211–252 (with discussion).

[13] Hoffman D.L. Boyes, W.J. and S.A. Low, *An econometric analysis of the bank credit scoring problem*, Journal of Econometrics, **40** (1989), 3–14.

[14] L. Breiman, *Bagging predictors*, Machine Learning **26, No. 2** (1995), 123–140.

[15] ———, *Random forests*, Machine Learning **45, No. 1** (2001), 5–32.

[16] L. Breiman, J. Friedman, R. Olshen, and C.J. Stone, *Classification and Regression Trees*, Chapman and Hall, 1984.

[17] I.N. Bronshstein and K.A. Semendyayev, *Handbook of mathematics*, Van Nostrand Reinhold, 1985.

[18] A. Bruce and H.-Y Gao, *Applied Wavelet Analysis with Splus*, Springer, 1990.

[19] S.M. Bruntz, W.S. Cleveland, B. Kleiner, and J.L. Warner, *The dependence on ambient ozone on solar radiation,temperature and mixing height*, Symposium on atmospheric diffusion and air pollution, American Meteorological Society, Boston, 1974, pp. 125–128.

[20] W. Buntine, *A Theory of Learning Classification Rules*, Ph.D. thesis, School of Computing Science,University of Technology, Sydney, 1992.

[21] B.P. Carlin and T.A. Louis, *Bayes and Empirical Bayes Methods for Data Analysis*, 2nd ed., Chapman and Hall/CRC, 2000.

[22] Jason Catlett, *Megainduction: Machine Learning on very large databases*, Ph.D. thesis, Basser Department of Computer Science, Queensland University of Technology, 1991.

[23] C. Chatfield, *Model uncertainty, data mining and statistical inference*, J. Roy. Statist. Soc., Series A **158** (1995), 419–466 (with discussion).

[24] N. Chawla, K. Bowyer, and W.P Kegelmayer, *SMOTE: synthetic minority oversampling technique.*, International Conference on Knowledge-based Systems, 2000.

[25] M. Chiogna, D.J. Spiegelhalter, R.C.G. Franklin, and K. Bull, *An empirical comparison of expert-derived and data-derived classification trees*, Statistics in Medicine **15** (1995), 157–169.

[26] George E.I. Chipman, H.A. and R.E. McCulloch, *Bayesian additive regression trees*, Proceedings of the Eight Valencia Conference on Bayesian Statistics (to appear), 2006.

[27] H. Chipman, E. George, and R McCullough, *Bayesian CART Model Search*, JASA **93** (1998), 935–960.

[28] ———, *Bayesian treed models*, Machine Learning **48** (2002), 299–320.

[29] B.H. Chirgwin and C Plumpton, *A Course of Mathematics for Engineers and Scientists*, vol. 1, Pergamon Press, 1970.

[30] W.S. Cleveland and S.J. Devlin, *Locally weighted regression: an approach to regression analysis by local fitting*, **83** (1988), 597–610.

[31] P. Craven and G. Wahba, *Smoothing noisy data with spline functions: estimating the correct degree of smoothing by the method of generalized cross-validation*, Numerische Matematik **31** (1979), 377–403.

[32] T. Dalenius, *Sampling in Sweden*, Published by Almqvist and Wicksell, 1957.

[33] P. Dawid, *Properties of diagnostic data distributions*, Biometrics **32** (1976), 647–658.

[34] ———, *The Well-calibrated Bayesian*, Journal of the American Statistical Association **77** (1976), 605–613 (with discussion).

[35] A. Dempster, M. Schatzoff, and N. Wermuth, *A simulation study of alternatives to ordinary least squares*, J. Am. Statist. Assoc **72** (1976), 77–106.

[36] Adams N.M. Holmes C.C. Denison, D.G.T. and D.J Hand, *Bayesian partition modelling*, Comp. Statist. Data Anal. **Volume 38,Issue 4 (February 2002)** (2002), 475–485.

[37] D. Denison and E.I. George, *Bayesian prediction using adaptive ridge estimators*, Tech. report, Imperial College, University of London, 2001.

[38] D. Denison, B. Mallick, and A.F.M. Smith, *A Bayesian CART algorithm*, Biometrika **85** (1998), 363–377.

[39] ———, *Classification with Bayesian MARS*, Machine Learning **50** (2003), 159–173.

[40] A. Doucet, N de Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*, Springer, 2001.

[41] Doyle, *The use of Automatic Interaction Detector and similar search procedures*, Op. Res. Quart. **24,** (1973), 465–467.

[42] D. Draper, *Assessment and propagation of model uncertainty*, J. Royal Statistical Society, Series B **57** (1995), 45–97.

[43] H. Einhorn, *Alchemy in the behavioural sciences*, Pub. Op. Quart. **36** (1972), 367–378.

[44] P Fearnhead, *Sequential Monte Carlo methods in Filter Theory*, Ph.D. thesis, Merton College, Oxford, 1998.

[45] A. Feelders, *An overview of model based reject inference for credit scoring*, Banff Credit Risk Conference, Unpublished, 2003.

[46] Denison D.G.T Ferreira J.T.A.S. and Hand D.J., *Data mining with products of trees*, Cascais, Portugal, 2001.

[47] R.A. Fisher, *The use of multiple regression in taxonomic problems*, Annals of Eugenics **7** (1936), 179–188.

[48] Y. Freund and R. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, Journal of Computer and System Sciences **55(1)** (1997), 119–139.

[49] Hastie T. Friedman, J. and R. Tibshirani, *Additive logistic regression: a statistical view of boosting*, The Annals of Statistics **28, No. 2** (2000), 337–407.

[50] J. H. Friedman, *Multivariate Adaptive Regression Splines (with discussion)*, Annals of Statistics **19** (1991), 1–141.

[51] ———, *On Bias, Variance and the Curse of Dimensionality, Technical Report*, Tech. report, Department of Statistics, Stanford University, 1996.

[52] J. H. Friedman and W. Stuetzle, *Projection Pursuit Regression*, Journal of the American Statistical Association **76** (1981), 817–823.

[53] J.H. Friedman and B. W. Silverman, *Flexible parsimonious smoothing and additive models*, Technometrics **31** (1989), 3–39 (with discussion).

[54] A. Gelman, J.B. Carlin, H. Stern, and D.B. Rubin, *Bayesian Data Analysis*, 2nd ed., CRC Press, 1995.

[55] S. Geman and D. Geman, *Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images*, IEEE Trans. Pattern Anal. Mach. Intelligence **6** (1984), 721–740.

[56] E. George and R.E. McCulloch, *Variable selection via Gibbs sampling*, J. Am. Statist. Assoc. **88** (1993), 881–889.

[57] W. Gilks, D. Spiegelhalter, and S. Richardson, *Markov Chain Monte Carlo in Practice*, Chapman and Hall, 1996.

[58] M. Goldstein and A.F.M. Smith, *Ridge-type estimators for regression analysis*, J. Roy. Statist. Soc. Series B **36** (1974), 284–291.

[59] P.J. Green, *Reversible Jump Markov Chain Monte Carlo computation and Bayesian model determination*, Biometrika **82** (1995), 711–32.

[60] W.H. Greene, *A statistical model for credit scoring, working paper ec-92-29*, Tech. report, Leonard N. Stern School of Business, 1992.

[61] _____, *Sample selection in credit-scoring models*, Japan and the World Economy **10** (1998), 299–316.

[62] D. Hand, *Idiots' Bayes-not so stupid after all?*, International Statistical Review **69,3** (2001), 85–398.

[63] _____, *Measuring diagnostic accuracy of statistical prediction rules*, Statistica Neerlandica **53** (2001), 3–16.

[64] D. Hand, H. Mannila, and P. Smyth, *Principles of Data Mining*, MIT Press, 2001.

[65] David J. Hand, *Construction and assessment of classification rules*, John Wiley, 1997.

[66] D.J. Hand, *No title*, Philadelphia Conference on Credit Risk Modeling and Decisioning, 2002.

[67] D.J Hand and V. Vinciotti, *Local versus global models for classication problems: Fitting models where it matters*, The American Statistician **57, No.2** (2003), 124–130.

[68] T.J. Hastie and L. Pregibon, *Shrinking trees*, Tech. report, AT and T Bell Laboratories Technical Report, 1990.

[69] T.J. Hastie and R.J. Tibshirani, *Generalized Additive Models*, Chapman and Hall/CRC, 1990.

[70] G. I. Heald, *The application of the A.I.D. programme and multiple regression techniques to the assessment of store performance and site selection*, Op. Res. Quart. **23** (1972), 445–457.

[71] F.S. Hillier and G. Lieberman, *Introduction to Operations Research*, 5 ed., McGraw-Hill, 1990.

[72] R.R. Hocking, F.M. Speed, and M.J. Lynn, *A class of biased estimators in linear regression*, Technometrics **18** (1976), 425–437.

[73] A.E. Hoerl and R.W. Kennard, *Ridge regression: biased estimation for nonorthogonal problems*, Technometrics **12** (1970), 55–67.

[74] J. Hoeting, D. Madigan, and C. Volinsky, *Bayesian Model Averaging: A tutorial*, Statistical Science **14** (1999), 382–401.

[75] Denison D.G.T. Holmes, C.C. and B.K. Mallick, *Bayesian partioning for classification and regression*, Tech. report, Imperial College London, 1999.

[76] D. Houghton and S. Oulabi, *Direct market modelling with CART and CHAID*, Journal of Direct Marketing **7** (1993), 3.

[77] H. Raiffa J. Pratt and R. Schlaiffer, *Introduction to Statistical Decision Analysis*, MIT Press, 1965.

[78] T. Jacobson and K. Roszbach, *Bank lending policy, credit scoring and value at risk. sse/efi working paper series in economics and finance 260*, Tech. report, Stockholm School of Economics, 1998.

[79] G. James and T Hastie, *Generalizations of the bias/variance decomposition for prediction error*, Tech. report, Department of Statistics, Stanford University, 1997.

[80] W. James and C.M. Stein, *Estimation with quadratic loss*, Proc. 4th Berkeley Symposium 1 (J. Neyman and E.L. Scott, eds.), 1961, pp. 361–380.

[81] W.H. Jefferys, *Bayesian Analysis of Random Event Generator Data*, Journal of Scientific Exploration **4 No 2** (1990), 153–169.

[82] W.H. Jefferys and J. Berger, *Sharpening Ockham's Razor on a Bayesian Strop, Technical Report 91-44c*, Tech. report, Department of Statistics, Purdue University, 1991.

[83] A. Karalič, *Employing linear regression in regression tree leaves*, Proceedings of ECAI-92, John Wiley and Sons, 1992.

[84] G.V. Kass, *Significance testing in automatic interaction detection*, Applied Statistics **24, 2,** (1976), 178–179.

[85] R Kass and A. Raftery, *Bayes factors*, J. Amer. Statist. Assoc. **90** (1995), 773–795.

[86] R. Kohavi and D. Wolpert, *Bias plus variance decomposition for zero-one loss functions*, Machine Learning, Proceedings of the Thirteenth International Conference, 1996.

[87] L. Kronsjö, *Algorithms: Their complexity and efficiency*, John Wiley, 1979.

[88] M. Kubat and S. Matwin, *Addressing the curse of imbalanced training sets: one sided selection*, Proceedings of the Fourteenth International Conference on Machine Learning, 1979, pp. 179–186.

[89] S. Kullback and R.A. Leibler, *On Information and Sufficiency*, Annals of Mathematical Statistics **22** (1951), 79–86.

[90] N.N. Lebedev, *Special functions and their applications*, Dover, 1972.

[91] H. Lee, *Consistency of posterior distributions for neural networks, Technical Report 676*, Tech. report, Department of Statistics, Carnegie-Mellon University, 1998.

[92] D.V. Lindley, *A statistical paradox*, Biometrika **44** (1957), 187–192.

[93] ———, *Scoring rules and the inevitability of probability*, International Statistical Review **50, (1)** (1982), 1–26.

[94] ———, *Making Decisions*, John Wiley, 1991.

[95] D.V. Lindley and A.F.M. Smith, *Bayes estimates for the linear model (with discussion)*, J Roy. Statist. Soc. B **34** (1972), 1–41.

[96] R. J. A. Little and D. B. Rubin, *Statistical analysis with missing data.*, John Wiley, 1997.

[97] S. Lloyd, *Ultimate physical limits of computation*, Nature **406** (2000), 1047–1054.

[98] S. Meester, *Reject inference for credit scoring model development using extrapolation*, Tech. report, Mimeo, CIT Group., New Jersey., 2002.

[99] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, E.H. Teller, and E. Teller, *Equations of state calculations by fast computing machines*, Journal of Chemical Physics **21** (1953), 1087–1091.

[100] J.N. Morgan and R.C. Messenger, *THAID: a sequential search program for the analysis of nominal scale dependent variables*, Tech. report, Ann Arbor: Institute for Social Research, University of Michigan, 1998.

[101] J.N Morgan and J.A. Sonquist, *Problems in the analysis of survey data, and a proposal*, Journal of the American Statistical Association **58** (1963), 415–434.

[102] J.A. Nelder and R.W.M. Wedderburn, *Generalized Linear Models*, J Roy. Statist. Soc. A **135** (1972), 370–384.

[103] J. Neyman and E.L. Scott, *Consistent estimates based on partially consistent observations*, Econometrica **16** (1948), 1–32.

[104] T. Oates and D. Jensen, *The effects of training set size on decision tree complexity*, Proceedings of the Sixth International Conference on Artificial Intelligence and Statistics, 1997.

[105] A. O'Hagan, *Probability: Methods and Measurement*, Chapman and Hall, 1988.

[106] L. Orr, *The dependence of transition proportions in the educational system on observed social factors and school characteristics*, J. R. Stat. Soc. A. **135** (1972), 74–95.

[107] R. Pike, *An Introduction to Quantum Computation and Quantum Communication*, Tech. report, Lucent Technologies, 2000.

[108] J.R. Quinlan, *Learning with continuous classes*, Proceedings of the 5th Australian Joint Conference on Artificial Intelligence, World Scientific, 1992.

[109] ———, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.

[110] H. Raiffa and R. Schlaiffer, *Applied Statistical Decision Theory*, Wiley Classics Library, 2000.

[111] B. Ripley, *Stochastic Simulation*, John Wiley, 1987.

[112] ———, *Pattern Recognition and Neural Networks*, Cambridge University Press, 1996.

[113] Ross and Bang, *The A.I.D. computer program, used to predict adoption of family planning in koyang*, Popul. Studies **20** (1966), 61–75.

[114] D. E. Rumelhart and J. L. McClelland (eds.), *Parallel Distributed processing : Explorations in the microstructure of cognition. Volume 1: Foundations*, MIT Press, Cambridge, MA, 1986.

[115] D.E. Rumelhart, G.E. Hinton, and R.J. Williams, *Learning representations by backpropagating errors*, Nature **323** (1986), 533–536.

[116] *The SAS Statistical System, The SAS Institute, Cary, NC, USA, url-http://www.sas.com.*

[117] G. Schwarz, *Estimating the dimension of a model*, Ann. Statist. **6** (1978), 461–464.

[118] Claude Shannon, *A Mathematical Theory of Communication*, Bell System Tech. J. **27** (1948), 379–423 and 623–656.

[119] J.Q. Smith, *Decision analysis: A Bayesian approach*, Chapman and Hall, 1988.

[120] D. Spiegelhalter and A.F.M. Smith, *Bayes Factors for linear and log-linear models with vague prior information*, J Roy. Statist. Soc. B **44** (1982), 377–387.

[121] *The Splus statistical system, Insightful Corp., USA, url: http://www.insightful.com.*

[122] Clifton D. Sutton, *Improving classification trees with simulated annealing*, Proceedings of the 23rd Symposium on the Interface (E. Keramidas, ed.), Interface Foundation of North America, 1991.

[123] R. Tibshirani T. Hastie and J. Friedman, *Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer Verlag, 2001.

[124] Rosanna Thrasher, *CART: A recent advance in tree-structured list segmentation methodology*, Journal of Direct Marketing **5, 1** (1991), 1.

[125] R. Tibshirani, *Bias, variance and prediction error for classification rules*, Tech. report, Department of Statistics, University of Toronto, 1996.

[126] R. Tibshirani and T. Hastie, *Generalized Additive Models*, Chapman and Hall, 1990.

[127] L. Tierney, *Markov chains for exploring posterior distributions (with discussion)*, The Annals of Statistics **22** (1994), 1701–1762.

[128] I Tomek, *Two modifications of cnn*, TTransactions on Systems Man and Communications **6** (1976), 769–772.

[129] I. Tuomi, *Moore's Law*, url: http://firstmonday.org/issues/issue7UNDERSCORE11/tuomi 2002.

[130] UCI, *UCI Machine learning archive*, url: http://www.ics.uci.edu/ mlearn/MLRepository.html, 9999.

[131] Vinciotti V. and Hand D.J., *Scorecard construction with unbalanced class sizes*, The Journal of the Iranian Statistical Society **2** (2002), 189–205.

[132] C.T. Volinsky, *Bayesian model averaging for censored survival data*, Ph.D. thesis, University of Washington, Seattle, 1997.

[133] *BUGS: Bayesian inference Using Gibbs Sampling, version 0.3.*

[134] Bianca Zadrozny, *Learning and evaluating classifiers under sample selection bias*, Proceeding of the 21st Internatonal Conference on Machine Learning, Banff, Canada, 2004.