Bayesian CART Model Search

by

Hugh Chipman, Edward I. George and Robert E. McCulloch[1]

The University of Waterloo,
The University of Texas at Austin
and
The University of Chicago

September 1995, revised December 1997

Abstract

In this paper we put forward a Bayesian approach for finding CART (classification and regression tree) models. The two basic components of this approach consist of prior specification and stochastic search. The basic idea is to have the prior induce a posterior distribution which will guide the stochastic search towards more promising CART models. As the search proceeds, such models can then be selected with a variety of criteria such as posterior probability, marginal likelihood, residual sum of squares or misclassification rates. Examples are used to illustrate the potential superiority of this approach over alternative methods.

Keywords: binary trees, Markov chain Monte Carlo, model selection, model uncertainty, stochastic search, mixture models.

1 Introduction

CART models are a flexible method for specifying the conditional distribution of a variable y, given a vector of predictor values x. Such models use a binary tree to recursively partition the predictor space into subsets where the distribution of y is successively more homogeneous. The terminal nodes of the tree correspond to the distinct regions of the partition, and the partition is determined by splitting rules associated with each of the internal nodes. By moving from the root node through to the terminal node of the tree, each observation is then assigned to a unique terminal node where the conditional distribution of y is determined. CART models were popularized in the statistical community by the seminal book of Breiman, Friedman, Olshen and Stone (1984). A concise description of CART modeling and its S-PLUS implementation appears in Clark and Pregibon (1992).

Given a data set, a common strategy for finding a good tree is to use a greedy algorithm to grow a tree and then to prune it back to avoid overfitting. Such greedy algorithms typically grow a tree by sequentially choosing splitting rules for nodes on the basis of maximizing some fitting criterion. This generates a sequence of trees each of which is an extension of the previous tree. A single tree is then selected by pruning the largest tree according to a model choice criterion such as cost-complexity pruning, cross-validation, or even multiple tests of whether two adjoining nodes should be collapsed into a single node.

In this paper we put forward a Bayesian approach for find-

[1]Hugh Chipman is Assistant Professor of Statistics, Department of Statistics and Actuarial Science, University of Waterloo, Waterloo, ON N2L 3G1, hachipman@uwaterloo.ca. Edward I. George is the Ed and Molly Smith Chair in Business Administration and Professor of Statistics, Department of MSIS, University of Texas, Austin, TX 78712-1175, egeorge@mail.utexas.edu. Robert E. McCulloch is Professor of Statistics, Graduate School of Business, University of Chicago, IL 60637, rem@gsbrem.uchicago.edu. For helpful discussions and suggestions, the authors would like to thank Wray Buntine, Jerry Friedman, Mark Glickman, Augustine Kong, Rob Tibshirani, Alan Zaslavsky, an associate editor, two anonymous referees and the participants of the 1995 International Workshop on Model Uncertainty and Model Robustness held in Bath, England This work was supported by NSF grant DMS 94.04408, Texas ARP grant 003658130, and research funding from the Graduate Schools of Business at the University of Chicago and the University of Texas at Austin.

ing CART models. The two basic components of this approach consist of prior specification and stochastic search. The basic idea is to have the prior induce a posterior distribution which will guide the stochastic search towards more promising CART models. As the search proceeds, such models can then be selected with a variety of criteria such as posterior probability, marginal likelihood, residual sum of squares or misclassification rates. Alternatively, a posterior weighted average of the visited models can be easily obtained. In a sense our procedure is a sophisticated heuristic for finding good models rather than a fully Bayesian analysis which presently seems to be computationally infeasible.

Our approach begins with the specification of a manageable prior distribution on the set of CART models. This entails the specification of a prior on the tree space and of a prior on the conditional distributions determined at the terminal nodes of each tree. Combining this prior with the tree model likelihood yields a posterior distribution on the set of tree models. A feature of this approach is that the prior specification can be used to downweight undesirable model characteristics such as tree complexity or to express a preference for certain predictor variables. In this way the posterior will put higher probability on the "better trees".

Although the entire posterior cannot be computed in nontrivial problems, Metropolis-Hastings algorithms can still be used effectively to explore the posterior. For this purpose, we construct particular versions of such algorithms which stochastically search for good tree models by rapidly gravitating towards regions of high posterior probability. As opposed to conventional greedy approaches which restrict the search to a particular "tree sequence", such algorithms search over a much richer class of candidate trees. Furthermore, by restarting our posterior search at trees found by other methods, our approach offers a systematic way to improve on alternate methods.

The potential of our approach is illustrated in depth in Section 7 where we apply it to the breast cancer data used by Breiman (1996) and Tibshirani and Knight (1995). This data consists of measurements on 683 breast tumors which might be useful for predicting whether a tumor is benign or malignant. For various CART model priors, repeated stochastic search of the posterior quickly found many models which were better than than those found by greedy approaches. For example, Figure 1 displays a five node CART models found by our procedure. The misclassification rate of this tree is 18 compared to a misclassification rate of 30 for the best five node greedy tree. In this example, we also considered restarting our posterior search at trees found by bootstrap bumping (Tibshirani and Knight (1995)), and found improvements roughly 80% of the time.

A related Bayesian approach to classification tree modeling was proposed by Buntine (1992) which, compared to our approach, uses similar priors for terminal node distributions, different priors on the space of trees, and deterministic, rather than stochastic, algorithms for model search. Priors for tree models based on Minimum Encoding ideas were proposed by Quinlan and Rivest (1989) and Wallace and Patrick (1993). Oliver and Hand (1995) discuss and provide an empirical comparison of a variety of pruning and Bayesian model averaging approaches based on CART. Paass and Kindermann (1997) applied a simpler version of our stochastic search approach (based on an early draft of this paper) and obtained results which uniformly dominated a wide variety of competing methods. Other alternatives to greedy search methods include Sutton (1991) who uses simulated annealing, Jordan and Jacobs (1994) who use the EM algorithm, Breiman (1996), who averages trees based on bootstrap samples, and Tibshirani and Knight (1995) who select trees based on bootstrap samples. Finally, during a revision of this paper, we became aware of a similar parallel development of Bayesian CART approach which is by Denison, Mallick and Smith (1997). Their approach is contrasted with ours in Section 8.

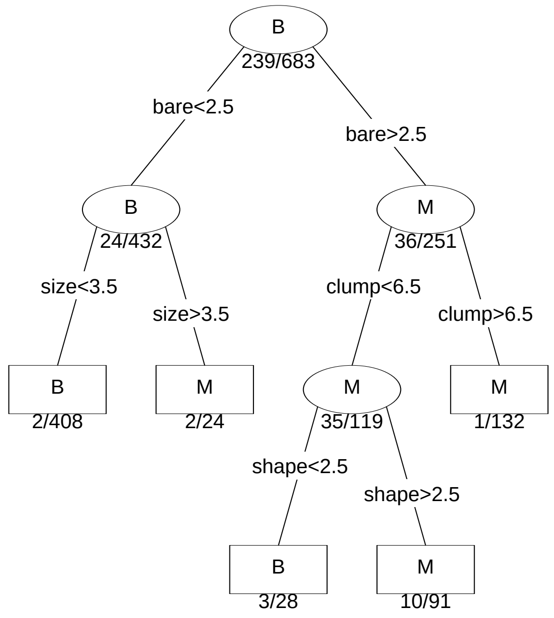

Figure 1: A five node tree found by by stochastic search. The overall misclassification rate is 18. The letters B and M, which refer to benign and malignant tumors, indicate the response which is in the majority in each node. The misclassification rates and number of observations are given below each node.

The paper is structured as follows. Section 2 describes the general structure of a CART model. Section 3 describes prior specifications for trees. Section 4 describes prior specifications for terminal node models. Section 5 outlines computational strategy for posterior exploration. Sections 6 and 7 compare the performance of our approach with competing methods on simulated and real data. Section 8 concludes with a discussion.

2 The Structure of a CART Model

We begin with a discussion of the general structure of a CART model so that the nature of the space on which we must place our prior is understood. A CART model describes the conditional distribution of y given x, where x is a vector of predictors ($x = (x_1, x_2, \ldots, x_p)$). This model has two main components: a tree T with b terminal nodes, and a parameter $\Theta = (\theta_1, \theta_2, \ldots, \theta_b)$ which associates the parameter value θ_i with the i^{th} terminal node. If x lies in the region corresponding to the i^{th} terminal node then $y|x$ has distribution $f(y|\theta_i)$, where we use f to represent a parametric family indexed by θ_i. The model is called a regression tree or a classification tree according to whether the response y is quantitative or qualitative, respectively.

The binary tree T subdivides the predictor space as follows. Each internal node has an associated splitting rule which uses a predictor to assign observations to either its left or right child nodes. The terminal nodes thus identify a partition of the observation space according to the subdivision defined by the splitting rules. For quantitative predictors, the splitting rule is based on a split value s, and assigns observations for which $\{x_i \leq s\}$ or $\{x_i > s\}$ to the left or right child node respectively. Note that this formulation is general enough to handle arbitrary splitting functions of the form $\{h(x) \leq s\}$ versus $\{h(x) > s\}$ by simply treating $h(x)$ as another predictor variable. For qualitative predictors, the splitting rule is based on a category subset C, and assigns observations for which $\{x_i \in C\}$ or $\{x_i \notin C\}$ to the left or right child node respectively.

For illustration, Figure 2 depicts a regression tree model where $y \sim N(\theta, 2^2)$ and $x = (x_1, x_2)$. x_1 is a quantitative predictor taking values in [0,10], and x_2 is a qualitative predictor with categories (A, B, C, D). The binary tree has 9 nodes of which $b = 5$ are terminal nodes which subdivide the x space into 5 nonoverlapping regions. The splitting variable and rule are displayed at each internal node. For example, the leftmost terminal node corresponds to $x_1 \leq 3.0$ and $x_2 \in \{C, D\}$. The θ_i value which identifies the mean of y given x is displayed at each terminal node. Note that θ_i decreases in x_1 when $x_2 \in \{A, B\}$, but increases in x_1 when $x_2 \in \{C, D\}$. (See also Figure 4).

If y were a qualitative variable, a classification tree model would be obtained by using an appropriate categorical distribution at each terminal node. For example, if y was binary with categories C_1 or C_2, one might consider the Bernoulli model $P(y \in C_1) = \theta = 1 - P(y \in C_2)$ with a possibly different value of θ at each terminal node. A standard classification rule for this model would then classify y into the category yielding the smallest expected misclassification cost.

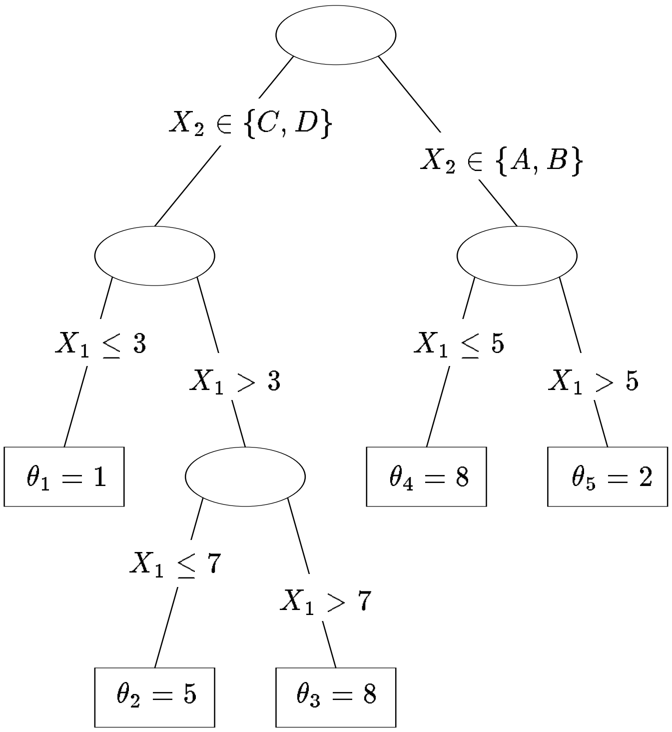

Figure 2: A regression tree where $y \sim N(\theta, 2^2)$ and $x = (x_1, x_2)$.

When all misclassification costs are equal, this would be the category with largest probability.

Association of the individual y values with the terminal nodes is indicated by letting y_{ij} denote the j^{th} observation of y in the i^{th} partition (corresponding to the i^{th} terminal node), $i = 1, 2, \ldots, b$, $j = 1, 2, \ldots, n_i$. Define

$$Y \equiv (Y_1, \ldots, Y_b)', \text{ where } Y_i \equiv (y_{i1}, \ldots, y_{in_i})', \tag{1}$$

and define X and X_i analogously. For CART models it is typically assumed that, conditionally on (Θ, T), y values within a terminal node are iid, and y values across terminal nodes are independent. In this case, the CART model distribution for the data will be of the form

$$p(Y \mid X, \Theta, T) = \prod_{i=1}^{b} f(Y_i \mid \theta_i) = \prod_{i=1}^{b} \prod_{j=1}^{n_i} f(y_{ij} \mid \theta_i). \tag{2}$$

Although we emphasize the iid case, more general models can be considered at the terminal nodes. For example, one might use linear relationships such as $E(y_{ij} \mid x_{ij}, \theta_i) = x_{ij}\theta_i$ at the ith node. This would allow for modeling the mean of Y by piecewise linear or quadratic functions rather than by constant functions as is implied by the iid assumption.

For regression trees, we consider two models where $f(y_{ij} \mid \theta_i)$ is normal, namely the mean shift model

$$y_{i1}, \ldots, y_{in_i} \mid \theta_i \ \text{ iid} \sim N(\mu_i, \sigma^2),\ i = 1, \ldots, b, \qquad (3)$$

where $\theta_i = (\mu_i, \sigma)$, and the mean-variance shift model

$$y_{i1}, \ldots, y_{in_i} \mid \theta_i \ \text{ iid} \sim N(\mu_i, \sigma_i^2),\ i = 1, \ldots, b, \qquad (4)$$

where $\theta_i = (\mu_i, \sigma_i)$. For classification trees where y_{ij} belongs to one of K categories, say $C_1, \ldots, C_K$, we consider $f(y_{ij} \mid \theta_i)$ to be generalized Bernoulli (i.e. simple multinomial) namely

$$f(y_{i1}, \ldots, y_{in_i} \mid \theta_i) \ = \ \prod_{j=1}^{n_i} \prod_{k=1}^{K} p_{ik}^{I(y_{ij} \in C_k)} \quad i = 1, \ldots, b, \quad (5)$$

where $\theta_i = p_i \equiv (p_{i1}, \ldots, p_{iK})$, $p_{ik} \geq 0$ and $\sum_k p_{ik} = 1$. Note that $P(y_{ij} \in C_k \mid p_i) = p_{ik}$.

Since a CART model is identified by (Θ, T), a Bayesian analysis of the problem proceeds by specifying a prior probability distribution $p(\Theta, T)$. Because Θ indexes the parametric model for each T, it will usually be convenient to use the relationship

$$p(\Theta, T) = p(\Theta \mid T)p(T), \qquad (6)$$

and specify $p(T)$ and $p(\Theta \mid T)$ separately. This strategy, which is commonly used for Bayesian model selection (George 1998), offers the advantage that the choice of prior for T does not depend on the form of the parametric family indexed by θ. Thus, the same approach for prior specification of T can be used for regression trees and classification trees. Another feature is that conditional specification of the prior on Θ more easily allows for the choice of convenient analytical forms which facilitate posterior computation. We proceed to discuss specification of $P(T)$ and $P(\Theta \mid T)$ in sections 3 and 4 below.

3 Specification of the Tree Prior $p(T)$

Instead of specifying a closed form expression for the tree prior $p(T)$, we specify $p(T)$ implicitly by a tree-generating stochastic process. Each realization of such a process can simply be considered as a random draw from this prior. Furthermore, many specifications allow for straightforward evaluation of $p(T)$ for any T, and can be effectively coupled with efficient Metropolis-Hastings search algorithms, as is shown in Section 5.

To draw from our prior we start with the tree consisting of a single root node. The tree then grows by randomly splitting terminal nodes, which entails assigning them splitting rules, and left and right children nodes. The growing process is determined by the specification of two functions $p_{SPLIT}(\eta, T)$ and $p_{RULE}(\rho \mid \eta, T)$. For an intermediate tree T in the process, $p_{SPLIT}(\eta, T)$ is the probability that terminal node η is split, and $p_{RULE}(\rho \mid \eta, T)$ is the probability of assigning splitting rule ρ to η if it is split. The stochastic process for drawing a tree from this prior can be described in the following recursive manner:

1. Begin by setting T to be the trivial tree consisting of a single root (and terminal) node denoted η.

2. Split the terminal node η with probability $p_{SPLIT}(\eta, T)$.

3. If the node splits, assign it a splitting rule ρ according to the distribution $p_{RULE}(\rho \mid \eta, T)$, and create the left and right children nodes. Let T denote the newly created tree, and apply steps 2 and 3 with η equal to the new left and the right children (if nontrivial splitting rules are available).

In what follows, we restrict $p_{SPLIT}(\eta, T)$ and $p_{RULE}(\rho \mid \eta, T)$ to be functions of the part of T above (i.e. ancestral to) η. This insures that the process does not depend on the order in which terminal nodes are considered in steps 2 and 3. As a consequence, for any internal node, the subtrees stemming from the right and left children are independent. While in some cases this may be restrictive, we find it to be a useful simplifying assumption.

We now proceed to discuss a variety of specifications for p_{SPLIT} and p_{RULE}. As will be seen, these specifications can be based on simple defaults, or can be made elaborate to allow for special structure.

3.1 Determination of Tree Size and Shape by p_{SPLIT}

As we have defined it, a CART tree T is composed of a binary tree and an assignment of splitting rules. To understand the role of p_{SPLIT}, it is useful to ignore the splitting rule assignment and focus on the distribution of binary trees obtained by successively splitting terminal nodes with probability p_{SPLIT} as in step 2 above. This would be the distribution of binary tree components of T obtained by steps 1-3 above if there were an infinite supply of splitting rules.

Let us first consider the simple specification, $p_{SPLIT}(\eta, T) \equiv \alpha < 1$. Under this specification, the probability of any particular binary tree with b terminal nodes is just $\alpha^{b-1}(1-\alpha)^b$. This generalization of the geometric probability distribution is obtained by noting that a binary tree with b terminal nodes must have $(b-1)$ internal nodes. Note that setting α small will tend to yield smaller trees and is a simple convenient way to control the size of trees generated by growing process.

The choice $p_{SPLIT}(\eta, T) \equiv \alpha$ is somewhat limited because it assigns equal probability to all binary trees with b terminal nodes regardless of their shape. A more general form which

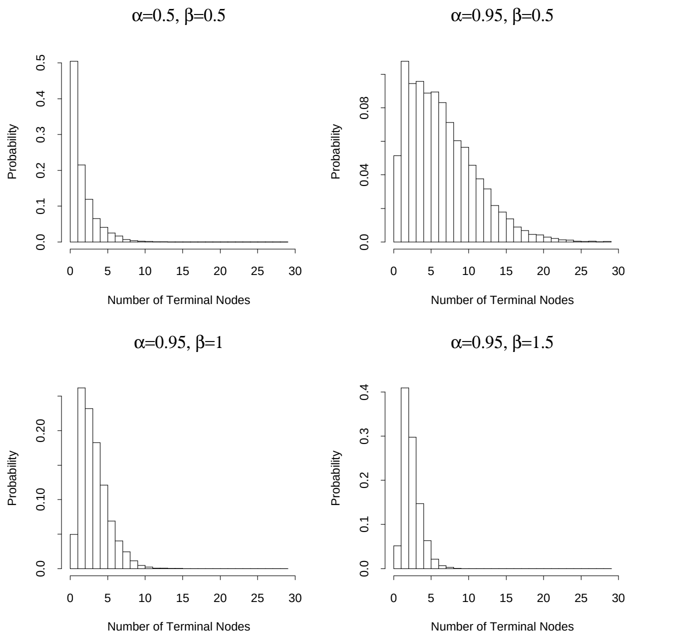

Figure 3: Prior distribution on number of terminal nodes. The prior means are 2.1, 7.0, 3.7, and 2.9 respectively.

allows for controlling the size and shape of the generated trees is

$$p_{SPLIT}(\eta, T) = \alpha(1 + d_\eta)^{-\beta} \quad (7)$$

where d_η is the depth of the node η (i. e. the number of splits above η), and $\beta \geq 0$. Under this specification, p_{SPLIT} is a decreasing function of d_η, more so for β large, making deeper nodes less likely to split. The resulting prior $p(T)$ puts higher probability on "bushy" trees, those whose terminal nodes do not vary too much in depth. Although more elaborate forms of p_{SPLIT} as a function of nodal ancestry can easily be constructed, (7) is flexible and simple.

In practice, we have found it convenient to choose α and β by exploring the consequent prior marginal distribution of some characteristic of interest such as the number of terminal nodes. Such a marginal can be readily simulated and graphed, facilitating the choice of α and β. For example, Figure 3 displays the prior distribution of the number of terminal nodes for $(\alpha, \beta) = (.5, .5), (.95, .5), (.95, 1.0), (.95, 1.5)$.

3.2 Specification of the Splitting Rule Assignments by p_{RULE}

As described in Section 2, a splitting rule ρ is determined by the choice of a predictor x_i and the choice of a split value s or a category subset C according to whether the predictor is quantitative or qualitative. Thus, it is convenient to describe $p_{RULE}(\rho \mid \eta, T)$ by a distribution on the set of available predictors x_i, and conditional on each predictor, a distribution on the available set of split values or category subsets. By available, we mean those predictors, split values and category subsets which would not lead to empty terminal nodes. For example, if a binary predictor was used in a splitting rule, it would no longer be available for splitting rules at nodes below it.

As a practical matter, we only consider prior specifications for which the overall set of possible split values is finite. Thus, each p_{RULE} will always be a discrete distribution. This is hardly a restriction since every data set is necessarily finite, and so can only be partitioned in a finite number of ways. As a consequence, the support of $p(T)$ will always be a finite set of trees. Note also that because the assignment of splitting rules will typically depend on X, the prior $p(T)$ will also depend on X. This dependence was noted by Buntine (1992) who discussed its potential advantages.

A simple choice of $p_{RULE}(\rho \mid \eta, T)$, which seems to work well in our examples, is the distribution obtained by choosing x_i uniformly from the available predictors, and then choosing s uniformly from the available observed values of x_i if x_i is quantitative, or choosing C uniformly from the set of available subsets if x_i is qualitative. This choice, which we refer to as the *uniform specification* of p_{RULE}, represents the prior information that at each node, available predictors are equally likely to be effective, and that for each predictor, available split values or category subsets are equally likely to be effective. An appealing feature of the uniform specification is that it is invariant to monotone transformations of the quantitative predictors. Note also that it is uniform on the observed quantiles of a quantitative predictor with no repeated values.

Although a natural default choice, the uniform specification for p_{RULE} assigns lower probability to splitting rules based on predictors with more potential split values or category subsets. This feature is necessary to maintain equal probability on predictor choices. The alternative uniform choice of p_{RULE} which is uniform across all available splitting rules, downweights predictors with fewer potential split values or category subsets. This seems unsatisfactory unless there is an a priori reason to expect such variables to have less predictive value.

Non-uniform choices for p_{RULE} may also be of interest. For example, in choosing a variable it may be preferable to place higher prior weight on predictors that are thought to be more important. We may instead prefer models that use only a few variables, i.e. variable selection. In this case, at node η, p_{RULE} would put greater mass on rules involving variables already used in ancestors of η, and less on rules involving unused variables. For the choice of split value, we might expect a region to split more towards its middle than near an edge and so might use a tapered distribution at the extremes. One might also consider the distribution of split values to be uniform on the available range of the predictor and so could weight the available observed values accordingly. For the choice of category subset, one might put extra weight on subsets thought to be more important.

4 Specification of the Parameter Prior: $p(\Theta \mid T)$

In choosing a prior for $\Theta \mid T$, it is important to be aware that using priors which allow for analytical simplification can substantially reduce the computational burden of posterior calculation and exploration. As will be seen in Section 5, this is especially true for prior forms $p(\Theta|T)$ for which it is possible to analytically margin out Θ to obtain:

$$p(Y \mid X, T) = \int p(Y \mid X, \Theta, T) p(\Theta \mid T) d\Theta. \tag{8}$$

In the following two subsections, we describe a variety of prior forms for which this marginalization is possible. These priors are obtained by putting conjugate priors on terminal node parameters and then assuming (conditional) independence of parameters across terminal nodes. Such priors are simple to describe and implement. Although we do not describe them here, it is also possible to construct analytically tractable hierarchical priors which use the tree structure to model parameter dependence across terminal nodes. Such priors are described and investigated in Chipman, George & McCulloch (1997).

4.1 Parameter Priors for Regression Trees

For regression trees with the mean-shift model normal model (3), perhaps the simplest prior specification for $\Theta \mid T$ is the standard conjugate form

$$\mu_1, \ldots, \mu_b \mid \sigma, T \ \text{ iid} \sim N(\bar{\mu}, \sigma^2/a) \tag{9}$$

and

$$\sigma^2 \mid T \sim \mathrm{IG}(\nu/2, \nu\lambda/2) \ \ (\Leftrightarrow \nu\lambda/\sigma^2 \sim \chi^2_\nu). \tag{10}$$

Under this prior, standard analytical simplification yields

$$p(Y \mid X, T) = \frac{c\ a^{b/2}}{\prod_{i=1}^b (n_i + a)^{1/2}} \left(\sum_{i=1}^b (s_i + t_i) + \nu\lambda \right)^{-(n+\nu)/2} \tag{11}$$

where c is a constant which does not depend on T, s_i is $(n_i - 1)$ times the sample variance of the Y_i values, $t_i = \frac{n_i a}{n_i + a}(\bar{y}_i - \bar{\mu})^2$, and $\bar{y}_i$ is the average value in Y_i.

In practice, the observed Y can be used to guide the choice of the prior parameters values for $(\nu, \lambda, \bar{\mu}, a)$. To begin with, because the mean-shift model attempts to explain the variation of Y, it is reasonable to expect that σ will be smaller than the sample standard deviation of Y, say s^*. Similarly, it is reasonable to expect that σ will be larger than a pooled standard deviation estimate obtained from a deliberate overfitting of the data by a greedy algorithm, say s_*. Using these values as guides, ν and λ would then be chosen so that the prior for σ assigns substantial probability to the interval (s_*, s^*). Once ν and λ have been chosen, $\bar{\mu}$ and a would be selected so that the prior for μ is spread out over the range of Y values.

For the more flexible mean-variance shift model (4) where σ_i can also vary across the terminal nodes, the conjugate form is easily extended to

$$\mu_i \mid \sigma_i \sim N(\bar{\mu}, \sigma_i^2/a) \tag{12}$$

and

$$\sigma_i^2 \sim \mathrm{IG}(\nu/2, \nu\lambda/2), \tag{13}$$

with the pairs $(\mu_1, \sigma_1), \ldots, (\mu_b, \sigma_b)$ independently distributed. Under this prior, analytical simplification is still straightforward, and yields

$$p(Y \mid X, T) = \prod_{i=1}^b \left[\pi^{-n_i/2} (\lambda\nu)^{\nu/2} \frac{\sqrt{a}}{\sqrt{n_i + a}} \times \frac{\Gamma((n_i + \nu)/2)}{\Gamma(\nu/2)} (s_i + t_i + \nu\lambda)^{-(n_i+\nu)/2} \right] \tag{14}$$

where s_i and t_i are as above.

As before, the observed Y can be used to guide the choice of the prior parameters values for $(\nu, \lambda, \bar{\mu}, a)$. The same ideas may be used with an additional consideration. In some cases, the mean-variance shift model may explain variance shifts much more so than mean shifts. To handle this possibility, it may be desirable to choose ν and λ so that s^* is more toward the center rather than the right tail of the prior for σ. We might also tighten up our prior for μ about the average y value. In any case, it may be appropriate to explore the consequences of several different prior choices.

Finally, note that it may be desirable to further extend the above priors to let the values of ν and λ depend on features of T. For example, because more complex trees (i.e. finer partitions of the predictor space) are likely to explain more variation, it might be reasonable to consider ν and λ as decreasing functions of tree complexity.

4.2 Parameter Priors for Classification Trees

For classification trees where y_{ij} belongs to one of K categories $C_1, \ldots, C_K$ under the generalized Bernoulli model (5), perhaps the simplest conjugate prior specification for $\Theta = (p_1, \ldots, p_b)$ is the standard Dirichlet distribution of dimension $K - 1$ with parameter $\alpha = (\alpha_1, \ldots, \alpha_K)$, $\alpha_k > 0$ namely

$$p_1, \ldots, p_b \mid T \ \text{ iid} \sim Dirichlet(p_i \mid \alpha) \propto p_{i1}^{\alpha_1 - 1} \cdots p_{iK}^{\alpha_K - 1}. \tag{15}$$

When $K = 2$ this reduces to the familiar Beta prior.

Under this prior, standard analytical simplification yields

$$P(Y|X, T) = \left(\frac{\Gamma\left(\sum_k \alpha_k\right)}{\prod_k \Gamma(\alpha_k)} \right)^b \prod_{i=1}^b \frac{\prod_k \Gamma(n_{ik} + \alpha_k)}{\Gamma\left(n_i + \sum_k \alpha_k\right)} \tag{16}$$

where $n_{ik} = \sum_j I(y_{ij} \in C_k)$, $n_i = \sum_k n_{ik}$ and $k = 1, ..., K$ over the sums and products above. For a given tree, $P(Y \mid X, T)$ will be larger when nodes are assigned more homogeneous values of y. To see this, note that assignments for which the y_{ij}'s at the same node are similar will lead to more disparate values of $n_{i1}, \ldots, n_{iK}$, which in turn leads to larger values of $P(Y \mid X, T)$.

The natural default choice for α is the vector $(1, \ldots, 1)$ for which the Dirichlet prior (15) is the uniform. However, by setting certain α_k to be larger for certain categories, $P(Y \mid X, T)$ will become more sensitive to misclassification at those categories. This might be desirable when classification into those categories is most important.

5 Stochastic Search of the Posterior

For each of the parameter prior forms $p(\Theta \mid T)$ proposed in Section 4, it is possible to analytically obtain $p(Y \mid X, T) = \int p(Y \mid X, \Theta, T) p(\Theta \mid T) d\Theta$ in (8), resulting in one of the closed forms (11), (14) or (16). Combining these with one of the CART tree priors $P(T)$ proposed in Section 3, allows us to quickly calculate the posterior of T

$$p(T \mid X, Y) \propto p(Y \mid X, T) p(T) \tag{17}$$

up to a norming constant.

Exhaustive evaluation of (17) over all T will not be feasible, except in trivially small problems, because of the sheer number of possible trees. (Recall that the number of possible trees is finite because we have restricted attention to a finite number of possible splitting functions). This not only prevents exact calculation of the norming constant, but also makes it nearly impossible to determine exactly which trees have largest posterior probability.

In spite of these limitations, Metropolis-Hastings algorithms can still be used to explore the posterior. Such algorithms simulate a Markov chain sequence of trees

$$T^0, T^1, T^2, \ldots \tag{18}$$

which are converging in distribution to the posterior $p(T \mid Y, X)$ in (17). Because such a simulated sequence will tend to gravitate towards regions of higher posterior probability, the simulation can be used to stochastically search for high posterior probability trees. We now proceed describe the details of such algorithms and their effective implementation.

5.1 Specification of the Metropolis-Hastings Search Algorithm

The Metropolis-Hastings (MH) algorithm for simulating the Markov chain $T^0, T^1, T^2, \ldots$ in (18) is defined as follows. Starting with an initial tree T^0, iteratively simulate the transitions from T^i to T^{i+1} by the two steps:

1. Generate a candidate value T^* with probability distribution $q(T^i, T^*)$.
2. Set $T^{i+1} = T^*$ with probability

$$\alpha(T^i, T^*) = \min \left\{ \frac{q(T^*, T^i)}{q(T^i, T^*)} \frac{p(Y \mid X, T^*) p(T^*)}{p(Y \mid X, T^i) p(T^i)}, 1 \right\}. \tag{19}$$

   Otherwise, set $T^{i+1} = T^i$.

Under weak conditions (see Tierney 1994), the sequence (18) obtained by this algorithm will be a Markov chain with limiting distribution $p(T \mid Y, X)$. Note that the norming constant for $p(T \mid Y, X)$ is not needed to compute (19).

To implement the algorithm, we need to specify the transition kernel q. We consider kernels $q(T, T^*)$ which generate T^* from T by randomly choosing among four steps:

- GROW: Randomly pick a terminal node. Split it into two new ones by randomly assigning it a splitting rule according to p_{RULE} used in the prior.
- PRUNE: Randomly pick a parent of two terminal nodes and turn it into a terminal node by collapsing the nodes below it.
- CHANGE: Randomly pick an internal node, and randomly reassign it a splitting rule according to p_{RULE} used in the prior.
- SWAP: Randomly pick a parent-child pair which are both internal nodes. Swap their splitting rules unless the other child has the identical rule. In that case, swap the splitting rule of the parent with that of both children.

In executing the GROW, CHANGE and SWAP steps, we restrict attention to available splitting rule assignments. By available, we mean splitting rule assignments which do not force the tree have an empty terminal node. We have also found it useful to further restrict attention to splitting rule assignments which yield trees with at least a small number (such as five) observations at every terminal node.

The proposal above has some appealing features. To begin with, it yields a reversible Markov chain because every step from T to T^* must have a counterpart that can move from T^* to T. Indeed, the GROW and PRUNE steps are counterparts of one another, and the CHANGE and SWAP steps are their own counterparts. Note that although other reversible moves can be considered, we have ruled them out because their counterparts are impractical to construct. For example, we have ruled out pruning off more than a pair of terminal nodes because the reverse step would be complicated and time consuming to generate.

Another appealing feature is that our MH algorithm is simple to compute. By using p_{RULE} to assign splitting rules in the GROW step, there is substantial cancelation between $p(T^*)$ and $q(T, T^*)$ in the calculation of $\alpha(T^i, T^*)$ in (19). In the PRUNE step similar cancelation occurs between $p(T)$ and $q(T^*, T)$. In the CHANGE and SWAP steps, calculation of the q values in (19) is avoided because their ratio is always 1.

5.2 Running the MH Algorithm for Stochastic Search

The MH algorithm described in the previous section can be used to search for desirable trees. To perform an effective search it is necessary to understand its behavior as it moves through the space of trees. By virtue of the fact that its limiting distribution is $P(T \mid Y, X)$, it will spend more time visiting tree regions where $P(T \mid Y, X)$ is large. However, our experience in assorted problems (see Sections 6 and 7) has been that the algorithm quickly gravitates towards such regions and then stabilizes, moving locally in that region for a long time. Evidently, this is a consequence of a proposal distribution which makes local moves over a sharply peaked multimodal posterior. Once a tree has reasonable fit, the chain is unlikely to move away from a sharp local mode by small steps. Because the algorithm is convergent, we know it will eventually move from mode to mode and traverse the entire space of trees. However, the long waiting times between such moves and the large size of the space of trees make it impractical to search effectively with long runs of the algorithm. Although different move types might be implemented, we believe that any MH algorithm for CART models will have difficulty moving between local modes.

To avoid wasting time waiting for mode to mode moves, our search strategy has been to repeatedly restart the algorithm. At each restart, the algorithm tends to move quickly in a direction of higher posterior probability and eventually stabilize around a local mode. At that point the algorithm ceases to provide new information, and so we intervene in order to find another local mode more quickly. Although the algorithm can be restarted from any particular tree, we have found it very productive to repeatedly restart at the trivial single node tree. Such restarts have led to a wide variety of different trees, apparently due to large initial variation of the algorithm. However, we have also found it productive to restart the algorithm at other trees such as previously visited intermediate trees or trees found by other heuristic methods. For example, in Section 7 we show that restarting our algorithm at trees found by bootstrap bumping (Tibshirani and Knight 1996) leads to further improvements over the start points.

5.3 Identifying the Good trees

As many trees are visited by each run of the algorithm, a method is needed to identify those trees which are of most interest. Because a long enough run of the chain is not feasible, frequencies of visits will not be useful. A natural choice is to evaluate $p(Y \mid X, T)p(T)$ for all visited trees and obtain their relative probabilities. One could then choose the trees with largest posterior probability. Alternatively, in the spirit of model averaging (see Breiman (1996) and Oliver and Hand (1995)), one could approximate the overall posterior mean by the average of the visited trees using weights proportional to $p(Y \mid X, T)p(T)$.

However, there is a subtle problem with using posterior probabilities to pick single trees in this context. Consider the following simple example. Suppose we were considering all possible trees with two terminal nodes and a single rule. Suppose further that we had only two possible predictors, one qualitative with two categories, the other quantitative with 100 possible split points. If the marginal likelihood $p(Y \mid X, T)$ was the same for all 101 rules, then the posterior would have a sharp mode on the qualitative rule. This is because the prior assigns small probability to each individual split value for the quantitative x, and much larger probability to the single rule on the qualitative x. The problem is that the relative sizes of the posterior modes does not capture the fact that the total posterior probability allocated to the quantitative predictor trees is the same as that allocated to the single qualitative tree.

It should be emphasized that the problem above is not a failure of the Bayesian prior. By using it, the posterior properly allocates high probability to tree neighborhoods which are collectively supported by the data. This serves to guide the algorithm towards such regions. The problem is that relative sizes of posterior modes do not capture the relative allocation of probability to such regions, and so can lead to misleading comparisons of single trees. Note that this would not be a limitation for model averaging.

A natural criterion for tree selection, which avoids the difficulties described above, is to use the marginal likelihood $p(Y \mid X, T)$. As illustrated in Section 7, a useful tool in this regard is a plot of the largest observed values of $p(Y \mid X, T)$ against the number of terminal nodes of T, an analogue of the C_p plot (Mallows 1973). This allows the user to directly gauge the value of adding additional nodes while removing the influence of $p(T)$. In the same spirit, we have also found it useful to consider other commonly used tree selection criteria such as residual sums of squares for regression trees and misclassification rates for classification trees. (The misclassification rate of a tree is the total number of observations different from the majority at each terminal node. When all misclassification costs are equal, it is an appropriate measure of model quality).

6 A Simulated Example

In this section, we illustrate the features of running our search algorithm by applying it to data simulated from the regression tree used for illustration in Figure 2. The model represented

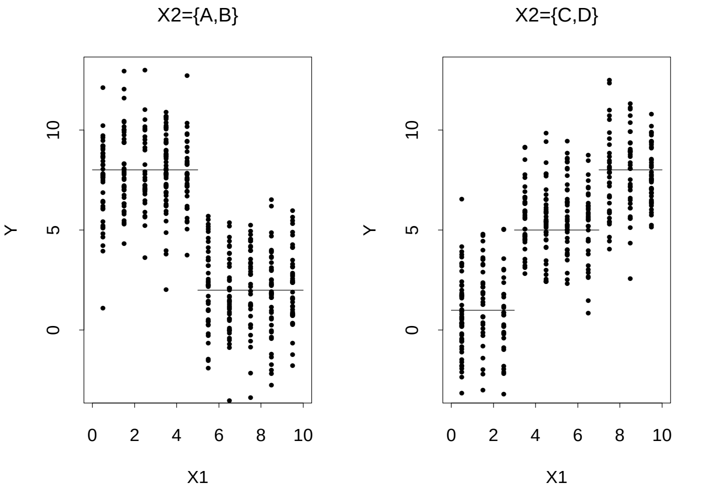

Figure 4: Simulated example, with 800 observations and true model overlaid.

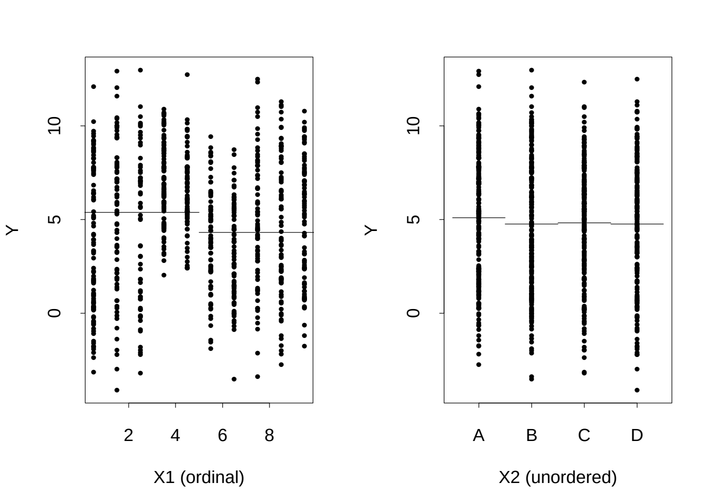

Figure 5: The response against the two predictors. The mean response for the best first split is given in X_1 plot, and mean levels for each X_2 in the other plot.

by that tree is:

$$Y = f(X_1, X_2) + 2\epsilon, \qquad \epsilon \sim N(0,1) \tag{20}$$

where

$$f(X_1,X_2) = \begin{cases} 8.0 & \text{if } X_1 \le 5.0 \text{ and } X_2 \in \{A,B\} \\ 2.0 & \text{if } X_1 > 5.0 \text{ and } X_2 \in \{A,B\} \\ 1.0 & \text{if } X_1 \le 3.0 \text{ and } X_2 \in \{C,D\} \\ 5.0 & \text{if } 3.0 < X_1 \le 7.0 \text{ and } X_2 \in \{C,D\} \\ 8.0 & \text{if } X_1 > 7.0 \text{ and } X_2 \in \{C,D\} \end{cases}.$$

Figure 4 displays 800 iid observations drawn from this model. One of the reasons we chose this particular model is that it tends to elude identification by the greedy algorithm which chooses splits to minimize residual sums of squares. To see why, consider Figure 5, which plots Y marginally against each X. The horizontal lines on the plots are the mean levels for the split chosen by the greedy algorithm. A first split on X_1 at 5 gives greater separation of means than any split on X_2. The greedy algorithm is unable to capture the correct model structure, instead making a first split on X_1.

We begin by applying our approach to a sample of 200 observations generated from the model (20). We used the tree prior $P(T)$ with $\alpha = 0.95$ and $\beta = 1$ for p_{SPLIT} in (7) (see Figure 3) and the uniform specification for p_{RULE}. Using the mean-variance shift model (4) for the terminal node distributions, we used the conjugate independence priors (12) and (13) with hyperparameter values based on the sample. The unconditional mean and variance of Y are $\hat{\mu}_Y = 4.85$ and $\hat{\sigma}^2_Y = 12$ respectively. The residual variance from an overfit greedy tree is roughly $\hat{\sigma}^2_{\min} = 4$. We consequently choose $\mu_i \mid \sigma_i \sim N(\hat{\mu}_Y, \sigma_i^2(\hat{\sigma}^2_Y/\hat{\sigma}^2_{\min})) = N(4.85, 3\sigma_i^2)$ and $\sigma_i \sim IG(\nu = 10, \lambda = \hat{\sigma}^2_{\min}) = IG(10,4)$ The prior on μ is informative but spread over the range of Y. The prior on σ_i^2 has the unconditional variance in the tail and the minimum variance near the mode.

To illustrate the behavior of the MH algorithm for this setup, we report characteristics of trees we found using ten runs of 1000 iterations, each time restarting from the single node tree. The top panel of Figure 6 displays the log posterior probabilities of the sequence of trees with a line drawn at the log posterior probability of the true tree. The figure clearly illustrates the typical behavior described in the previous section; the algorithm quickly gravitates towards regions of high posterior probability and then stabilizes, moving locally within that region. The middle panel of Figure 6 displays the (marginal) log likelihoods with a line is drawn at the log likelihood of the true tree. Note the variety of tree likelihoods found by the different restarts. One run, the third, seems to be trapped in a region of trees having relatively low posterior probabilities. The last two runs have led to trees with the "optimal" likelihood. Note that single long run would be much more likely to get stay trapped near a suboptimal tree. Finally, the lower panel of the Figure 6 gives the corresponding number of terminal nodes, with a line drawn at the size of the true tree. The likelihood makes little distinction between trees found in the last three runs, but the posterior downweights the eight run as it is too large. Interestingly, the dead end runs tend to find trees that are too large.

As Figure 6 shows, multiple restarts are a crucial part of tree searches based on our algorithm. To further illustrate the need for restarts and other features of our approach, we perform a Monte Carlo comparison of several strategies, based on 200 distinct samples of size 800 from (20). On each sample, we

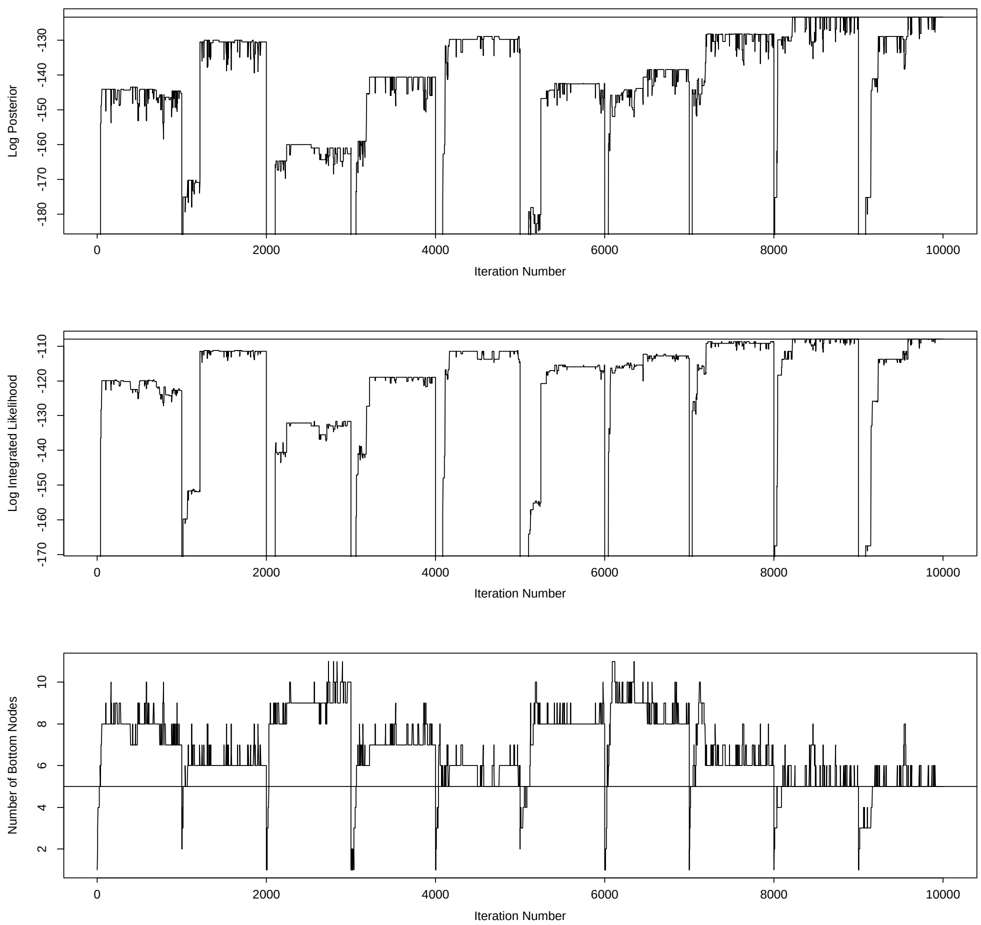

Figure 6: Log marginal likelihood and number of terminal nodes for trees visited by the Metropolis algorithm.

compare our recommended strategy of restarting versus the strategy of using one long run. We also show what happens to our strategy when the SWAP step is deleted, and when the tree prior is relaxed. More precisely, on each of the 200 samples, 40,000 iterations of the following four strategies are compared. Note that the last three strategies are identical to the "base", except as noted:

Base: All four move types (GROW, PRUNE, CHANGE, SWAP) with equal probabilities, 20 restarts, 2000 iterations after each restart, a tree prior p_{SPLIT} with $\alpha = .95, \beta = .5$ (giving 6.5 expected terminal nodes).

No Restart: One long chain of 40,000 iterations.

No SWAP: The SWAP move is not used. The remaining moves have equal probabilities.

Loose prior: The tree prior p_{SPLIT} is changed to $\alpha = .95, \beta = .25$, giving 11.8 expected terminal nodes.

Each strategy is based on identical priors for μ and σ described earlier in this section. Thus each strategy is using the same marginal likelihood $p(Y \mid X, T)$ as input.

The four approaches are compared in terms of the number of iterations required to correctly identify the initial splitting rule of the tree, namely $X_2 \in \{A, B\}$ vs. $X_2 \in \{C, D\}$. Finding this split is crucial in this problem because the true structure of the tree cannot be parsimoniously captured using a different initial split. Figure 7 displays boxplots of the number of iterations to correct identification on each of the 200 samples for the four strategies. In cases where the method failed to make a correct identification in 40,000 iterations, the value was recorded as 44,000.

Figure 7 shows the Base procedure performed best. The strategy of using one long run was clearly worst, over 75% of the chains failed to identify the correct split in 40,000 steps. Omission of the SWAP step inhibits mixing, and slows the chain considerably. The SWAP step is useful here because when correct rule is first found at a lower level, the swap allows that rule to be moved up. Lastly, in this particular

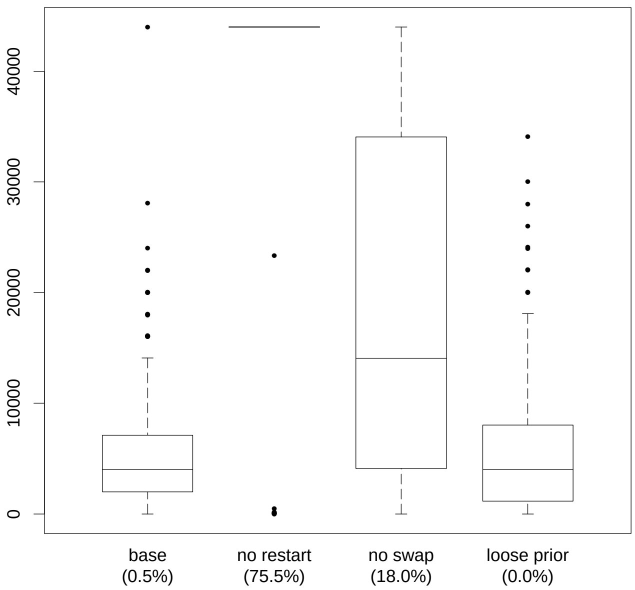

Figure 7: Number of iterations required to correctly identify the root node rule for four versions of the Metropolis chain. Runs for which the correct rule is not identified are coded 44,000, and the percent of these failing runs is given in brackets.

example, loosening up the prior has only a very slight effect.

Because of its similar stochastic nature, we were interested in the comparative performance of bootstrap bumping (Tibshirani and Knight (1995)) on this data set. Bootstrap bumping entails "varying the data" by bootstrap resampling, using a greedy algorithm to find a tree for each bootstrap sample, and then from these, picking the tree which performs best on the original data. By introducing variation to the data, it is hoped that the greedy method will get over local minima and find a better model. However, it turns out that this is not the case on this example. With 800 observations, the true tree was not located once with 1000 resamplings, probably because of the feature illustrated by Figure 5. Part of this difficulty has to do with the large amount of data; when 200 observations are used, the correct tree is usually identified within 100 resamplings. Interestingly, bootstrap bumping seems to work better when there is less data.

Finally, it should be mentioned that we are not trying to argue that our approach can defeat any competing algorithm. Indeed, modified greedy algorithms which look ahead by more than one step would probably perform very well on this example. Although it should be possible to create similarly challenging examples for such algorithms, our purpose here has been to simply to demonstrate how a stochastic search algorithm such as ours can avoid defeat by systematic structure.

Variable	Code
Clump Thickness	clump
Uniformity of Cell Size	size
Uniformity of Cell Shape	shape
Marginal Adhesion	adhes
Single Epithelial Cell Size	secs
Bare Nuclei	bare
Bland Chromatin	bland
Normal Nucleoli	normal
Mitoses	mitoses
Class:	class (2=benign, 4=malignant)

Table 1: Variable names, breast cancer data

7 The Breast Cancer Data

In this section, we illustrate the application of our procedure to the breast cancer data used by Breiman (1996) and Tibshirani and Knight (1995) to illustrate their bootstrap tree modeling procedures. The data was obtained from the University of California, Irvine repository of machine learning databases (`ftp://ftp.ics.uci.edu/pub/machine-learning-databases`). The data was given to the repository by William H. Wolberg, University of Wisconsin Hospitals, Madison (see Wolberg and Mangasarian, 1990). The data consists of nine cellular characteristics which might be useful to predict whether a tumor is benign or malignant (the binary variable `class`). All cellular characteristics are ordered numeric variables, each with levels $1, 2, \ldots, 10$. The variable names are given in Table 1. Of the original 699 observations, 683 complete observations were used.

Correlations between predictors range from from 0.34 to 0.91 in absolute value, so many different trees may describe the data well. Ten-fold cross-validation applied to greedy trees indicates that five to ten terminal nodes is a reasonable tree size. Based on this three of the tree prior settings in Figure 3 seemed reasonable. The values for p_{SPLIT} were $\alpha = .95$, and $\beta = 0.5, 1.0, 1.5$. The induced priors on the number of terminal nodes have means 7.0, 3.7, and 2.8, respectively. Roughly, these correspond to large, medium, and small trees. We also used the uniform specification for p_{RULE}. Using the Bernoulli model (5) for the terminal node distributions, we put a uniform prior on the Bernoulli parameter.

To search for promising trees over the induced posterior, we ran the MH algorithm 500 times with 5000 iterations per run. Each run was restarted at the single node tree. For all three priors, roughly the same size trees were visited. Although a large number of restarts were used, even the first chain finds trees that beat greedy trees of the same size. As mentioned in Section 5.3, we used both marginal likelihood and misclassification rates to identify promising trees. The marginal log likelihood of the data for identified trees is given in Figure 8, broken down by number of terminal nodes on the horizontal axis. Figure 9 plots misclassification rates in a similar fashion.

Although similar trees are identified by all three runs, the most likely trees are visited during the run with the medium prior. Misclassification rates of 11 observations out of 683 are attained with nine or more terminal nodes. This suggests that a nine node tree may be sufficient. The likelihood prefers a 10 node tree, and may provide more discrimination in this example (since misclassifications are of the order of 10 in 683). The trees found by our procedure had lower misclassification rates than greedy trees constructed using deviance pruning with Splus. The greedy trees were substantially worse, yielding misclassification rates of 30, 29, and 21 for trees with 5, 6, and 7-10 terminal nodes.

For each of the three tree priors, we proceeded to consider the best tree with 9, 10, and 11 terminal nodes, where best means the tree with the largest likelihood and the lowest misclassification rate. (Such best trees may not always exist). This group included a variety of tree structures, both in terms of topology and splitting rules. Several of these trees were "bushy" like the tree displayed in Figure 1, and similar to the trees selected by the greedy algorithm. In contrast, others, like the one given in Figure 10, had a very unbalanced appearance. It is interesting to note that this tree has a surprisingly simple interpretation which can be summarized as:

```
if (clump>8.5) or (normal>8.5) or (size>4.5) or
(bare>6.5) then malignant
else benign,  unless (2.5<bare<4.5) and
(clump>4.5)
```

This last condition seems at odds with the direction of all the other rules; its removal would further simplify the classification rule.

Although restarting from the single node tree produced very satisfactory results, we also considered restarting from trees found by bootstrap bumping. To construct these bumped trees, 500 bootstrap samples of the original data were generated. For each bootstrap sample, a greedy tree with 9 nodes (and at least 5 observations in each terminal node) was grown. The 10 best trees (in terms of log likelihood) were used as starting points for our algorithm. Starting at each of these 10 trees, 25 chains of length 1000 were run, amounting to one tenth of the computational effort ($10 \times 25 \times 1000 = 250,000$ steps) spent on starting from null trees. Shorter runs were used since the bumped trees already fit the data well.

Roughly 80% of the 250 chains visited trees with better log likelihoods than the bootstrap trees from which they were started. Trees with log likelihoods as high as -62.2 were identified by this procedure. This represents a substantial improvement over the best of the 10 bumped trees, which had a log likelihood of -67.3. Even shorter runs than those reported here produced improvements on the bumped trees. Restarting at bumped trees also yielded misclassification rate improvements. The best nine node trees identified by bumping had misclassification rates of 15. From the bumped start points, misclassification rates as low as 13 were attained with eight node trees. Although these improvements are not dramatic (probably because the rates are so low to begin with), it is interesting to note that even lower misclassification rates were found by restarting at the single node tree.

Finally, we also investigated the potential for model averaging in this example. Specifically, we construct estimates of malignancy probability for each individual by averaging across trees using posterior model probabilities as weights. All models with log posterior probability within 5.43 of the most probable tree were used. This set consisted of 1152 distinct models. Weights ranged from 10^{-4} to 0.0225. Trees averaged over had from four to eight terminal nodes, with the majority having six or seven. Misclassification rates of these trees ranged from 13 to 33. The posterior averages agree well with the data, having a misclassification rate of 15. In this example, there is little room for improvement, since the data is fit well by many trees. Even so, it is likely that the averaged tree would provide better out of sample prediction, being a more stable average of many trees. This is a promising area for future work.

8 Discussion

In this paper we put forward a Bayesian approach for finding CART models. Identifying each CART model by the tree T and the terminal node parameters Θ, prior specification is facilitated by considering $P(T)$ and $P(\Theta \mid T)$ separately. We have proposed specifying $P(T)$ by a tree generating process which randomly grows trees and assigns splitting rules. This process can be easily specified using simple defaults, or can be made elaborate to allow for special structure. For the specification of $P(\Theta \mid T)$, we have proposed a variety of conjugate forms which allow for analytical marginalization of Θ from the posterior. This marginalization substantially reduces the computational burden of our overall approach.

For stochastic search of the posterior, we have proposed a Metropolis-Hastings algorithm which moves around the posterior by gravitating towards regions of high probability. The key to our algorithm is a proposal distribution which randomly selects one of four moves in tree space: GROW, PRUNE, CHANGE or SWAP. With these moves our algorithm is easy to compute and moves quickly to posterior modes. Although the vast size of the CART model space and the multimodal nature of the posterior prevent the simulated Markov chain from converging, our experience indicates that the algorithm does succeed in finding at least some of

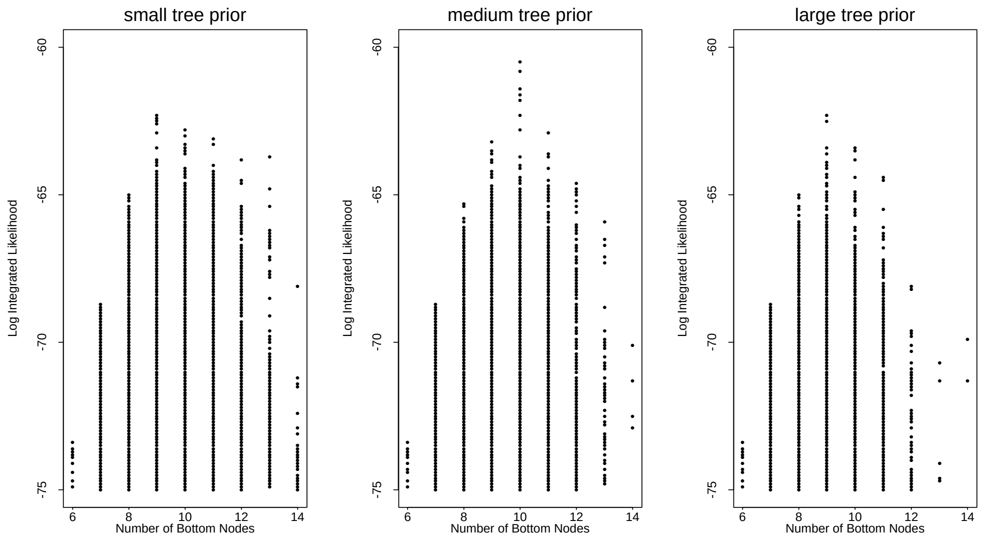

Figure 8: Log marginal likelihood against tree size.

the high posterior regions. To take advantage of this behavior, we have found it fruitful to repeatedly restart the search after a local mode has been found. The examples in Sections 6 and 7 illustrate that our approach is capable of finding a set of (possibly quite different) trees that fit the data well, and can outperform those trees of similar size found by alternative methods.

Finally, it may be of interest to compare our approach with the independent development of Denison et.al. (1997). Although the basic thrusts are similar, there are key differences. To begin with, we use a tree generating process prior which can depend on both tree size and shape, whereas they use a simple truncated Poisson prior which puts equal weight on equal sized trees. For splitting value assignment for quantitative predictors, we recommend a discrete uniform prior on the observed predictor values, an invariant prior, whereas they recommend a continuous uniform on the predictor range, which is not invariant. For regression tree parameters, we use proper conjugate priors and then margin out analytically, whereas they use improper priors, plug-in parameter estimates, and do not completely margin.

For stochastic search, we both use Metropolis-Hastings algorithms with the same GROW, PRUNE and CHANGE steps. However, we include an additional SWAP step. (In an early version of this paper, our CHANGE step was more limited. However, we enhanced it after seeing an early version of their paper). To cope with their continuous splitting values priors, they recommend an elaborate reversible jump algorithm (Green 1995) which requires an additional rejection step. Finally, they recommend stochastic search using a single long run preceded by a burn-in period, "after which posterior probabilities have been settled for some time". In sharp contrast, we have strongly cautioned against such a strategy, because even with our additional SWAP step, the algorithm quickly gets trapped in a local posterior mode after which it only moves locally, (see Section 6). Our recommended strategy of continual restarts is deliberately designed to avoid this problem.

References

Breiman, L., Friedman, J. Olshen, R. and Stone, C. (1984), *Classification and Regression Trees*, Wadsworth.

Breiman, L (1996), "Bagging Predictors", *Machine Learning*, 24, 123–140.

Buntine, W. (1992), "Learning Classification Trees", *Statistics and Computing*, 2, 63–73.

Clark, L., and Pregibon, D. (1992), "Tree-Based Models" in *Statistical models in S*, J. Chambers and T. Hastie, Eds., Wadsworth.

Chipman, H., George, E.I. & McCulloch, R.E. (1997) "Hierarchical Bayesian CART", Technical Report, Department of MSIS, University of Texas at Austin.

Denison, D., Mallick, B. and Smith, A.F.M. (1997) "A Bayesian CART Algorithm", Technical Report, Department of Mathematics, Imperial College, London.

George, E.I. (1998), "Bayesian Model Selection", *Encyclopedia of Statistical Sciences Update*, Volume 3, Wiley, New York.

Green, P. (1995), "Reversible Jump Markov chain Monte Carlo Computation and Bayesian Model Determination",*Biometrika*, 82, 711-32.

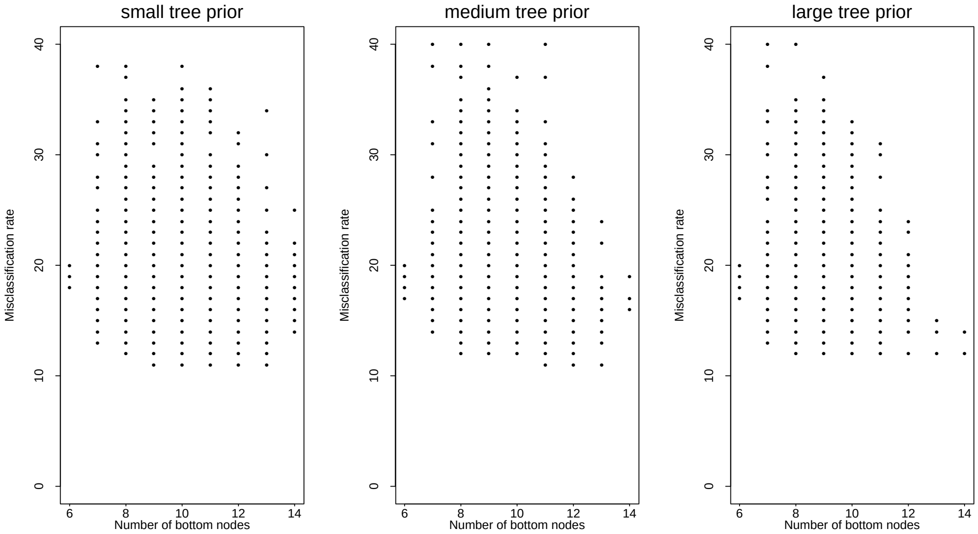

Figure 9: Misclassification rate against tree size.

Jordan, M.I. and Jacobs, R.A. (1994), " Mixtures of Experts and the EM Algorithm", *Neural Computation*, 6, 181-214.

Mallows, C. L. (1973), "Some Comments on C_p." *Technometrics*, 15, 661–676.

Oliver, J.J. and Hand, D.J. (1995). "On Pruning and Averaging Decision Trees", Proceedings of the International Machine Learning Conference, 430-437.

Paass, G. and Kindermann, J. (1997). "Describing the Uncertainty of Bayesian Predictions by Using Ensembles of Models and its Application", *1997 Real World Computing Symposium*, Real World Computing Partnership, Tsukuba Research Center, Tsukuba, Japan, p. 118–125.

Quinlan, J.R. and Rivest, R.L. (1989). "Inferring decision trees using the minimum description length principle", *Information and Computation*, 80:227-248.

Sutton, C. (1991), "Improving Classification Trees with Simulated Annealing", *Proceedings of the 23rd Symposium on the Interface*, E. Keramidas, Ed., Interface Foundation of North America.

Tibshirani, R., and Knight, K. (1995), "Model Search and Inference by Bootstrap 'Bumping' ", University of Toronto technical report.

Tierney, L. (1994), "Markov Chains for Exploring Posterior Distributions," *Annals of Statistics*, 22, 1701–1762.

Wallace, C.C. and Patrick, J.D. (1993). "Coding decision trees", *Machine Learning*, 11, 7-22.

Wolberg, W. H. and Mangasarian, O. L. (1990), "Multisurface method of pattern separation for medical diagnosis applied to breast cytology", Proceedings of the National Academy of Sciences, 87, 9193-9196.

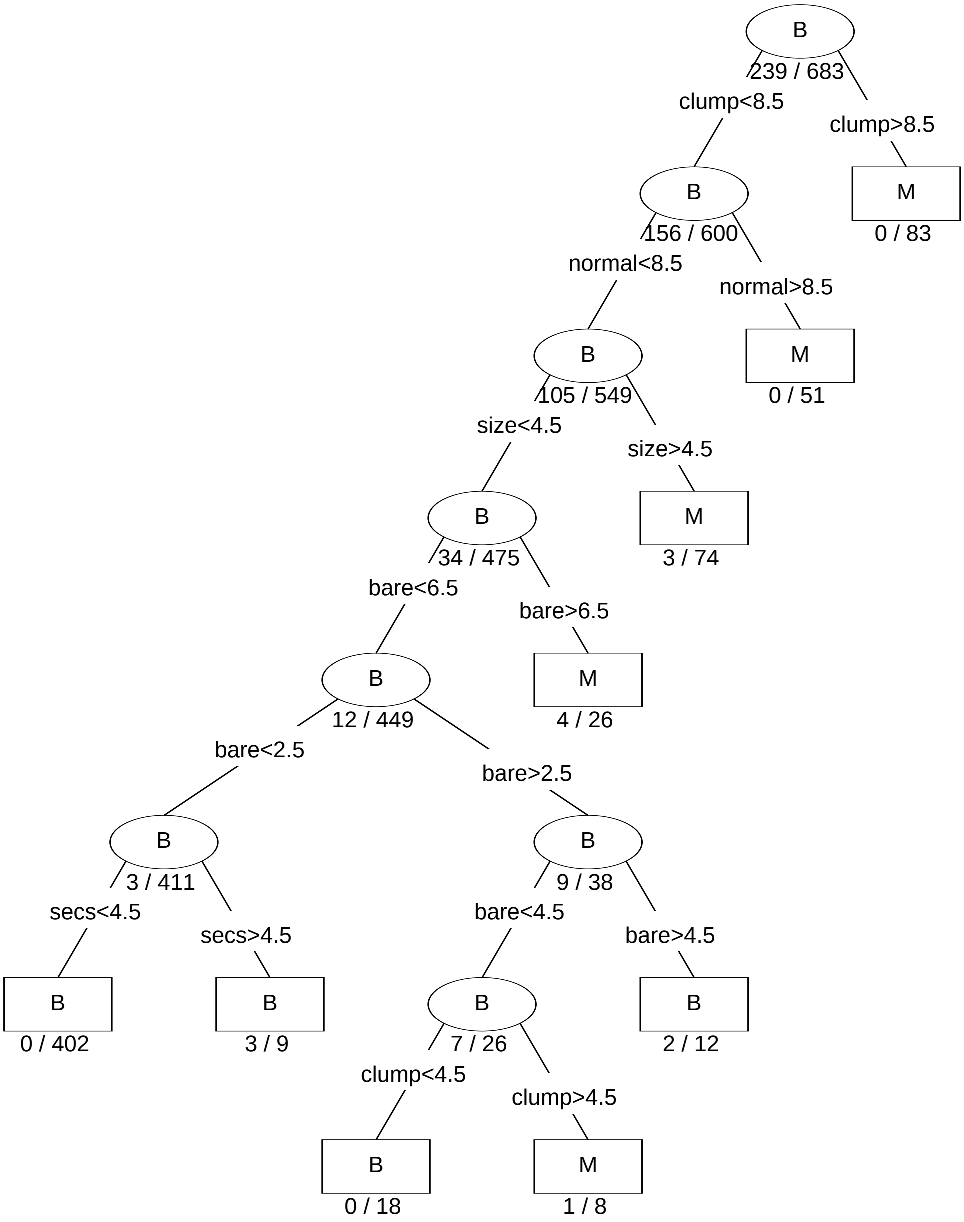

Figure 10: A nine node tree found by by stochastic search. The overall misclassification rate is 13, and log marginal likelihood -63.4.