



A Bayesian CART Algorithm

David G. T. Denison; Bani K. Mallick; Adrian F. M. Smith

Biometrika, Vol. 85, No. 2 (Jun., 1998), 363-377.

Stable URL:

<http://links.jstor.org/sici?sici=0006-3444%28199806%2985%3A2%3C363%3AABCA%3E2.0.CO%3B2-M>

Biometrika is currently published by Biometrika Trust.

Your use of the JSTOR archive indicates your acceptance of JSTOR's Terms and Conditions of Use, available at <http://www.jstor.org/about/terms.html>. JSTOR's Terms and Conditions of Use provides, in part, that unless you have obtained prior permission, you may not download an entire issue of a journal or multiple copies of articles, and you may use content in the JSTOR archive only for your personal, non-commercial use.

Please contact the publisher regarding any further use of this work. Publisher contact information may be obtained at <http://www.jstor.org/journals/bio.html>.

Each copy of any part of a JSTOR transmission must contain the same copyright notice that appears on the screen or printed page of such transmission.

JSTOR is an independent not-for-profit organization dedicated to creating and preserving a digital archive of scholarly journals. For more information regarding JSTOR, please contact jstor-info@umich.edu.

A Bayesian CART algorithm

BY DAVID G. T. DENISON, BANI K. MALLICK
AND ADRIAN F. M. SMITH

Department of Mathematics, Imperial College, London SW7 2BZ, U.K.

d.denison@ic.ac.uk b.mallick@ic.ac.uk a.smith@ic.ac.uk

SUMMARY

A stochastic search form of classification and regression tree (CART) analysis (Breiman et al., 1984) is proposed, motivated by a Bayesian model. An approximation to a probability distribution over the space of possible trees is explored using reversible jump Markov chain Monte Carlo methods (Green, 1995).

Some key words: Bayesian method; Classification tree; Regression tree; Reversible jump Markov chain Monte Carlo.

1. INTRODUCTION

The CART method of Breiman et al. (1984) addresses the classification and regression problem by building a binary decision tree according to some splitting rule based on the predictor variables. In this way, the space of predictor variables is partitioned recursively in a binary fashion. The partition is intended to increase within-node homogeneity, where homogeneity is determined by the response variable in the problem. The partitioning is repeated until a node is reached for which no split improves the homogeneity, whereupon the splitting is stopped and this node becomes a terminal node. Prediction is determined by terminal nodes, and takes the form either of a class level in classification problems, or the average of the response variable in least squares regression problems (Breiman et al., 1984, Ch. 2, 8).

A tree T has a root node whose descendant nodes, also known as daughters, can be divided into terminal nodes and split nodes. Throughout the paper terminal nodes will be denoted by rectangles and be identified with a t , split nodes by ellipses and identified with an s .

Suppose the data consist of a response variable Y and vector of predictor variables $X = (X_1, X_2, \dots, X_m)$, with fixed dimensionality m , where X_i can be ordered, i.e. continuous or discrete ordinal, or categorical, i.e. nominal. The tree itself is grown as follows (Breiman et al., 1984, Ch. 2).

At each node do the following.

Step 1. Examine every allowable split on each predictor variable. Usually the binary splits are generated by binary questions.

Step 2. Select and execute the ‘best’ of these splits.

Step 3. Stop splitting on a node when some stopping rule is satisfied.

For ordered variables X_i the questions in Step 1 are of the form $\{Is X_i > c?\}$ for all c in the range of X_i . If X_i is categorical, taking a finite number of values $\{b_1, b_2, \dots, b_l\}$,

say, the questions are of the form $\{Is X_m \in C?\}$ as C ranges over all subsets of $\{b_1, b_2, \dots, b_l\}$.

Those cases in t answering 'yes' go to the left descendant node and those answering 'no' to the right descendant node.

The 'best' in Step 2 is assessed in terms of some choice of goodness-of-split criterion (Breiman et al., 1984, Ch. 4). Two popular criteria are 'least squares' and 'least absolute deviations'. Both afford a comparison based on a subadditive 'between/within' decomposition, where 'between' alludes to the homogeneity or loss measure applied to the parent node.

The rules for Step 3 that have been considered seem not to work well in practice (Breiman et al., 1984, pp. 59–62). The tree tends to grow too big and have too few data points in each terminal node to make the study worthwhile. To overcome this problem, trees are grown which are very large with few data points in each terminal node and are then recursively pruned. Different ways of doing this are examined in Breiman et al. (1984, Ch. 3), with the most common method being the minimal cost complexity procedure (Breiman et al., 1984, pp. 66–71). Frequently, the pruned subtree is constrained to have more than some minimum number of data points in each terminal node.

At each node, the tree algorithm searches through the variables one by one, beginning with X_1 and continuing up to X_m . For each variable it finds the best split. Then it compares the m best single variable splits and selects the best of these.

Steps 1 and 2 are then reapplied to each of the daughter nodes, and so on, thus arriving at the full tree.

When we have only continuous variables, another way of looking at the tree structured procedure is as a recursive partitioning of X into hyper-rectangles such that the population within each rectangle becomes more and more class homogeneous. With a continuous response variable, this is equivalent to fitting a response surface consisting of constant heights within rectangular blocks.

The aim of this paper is to provide a Bayesian algorithm which mimics the CART procedure by regarding the number of splitting nodes, their positions and the questions used at the nodes as unknowns. We treat these as additional parameters in the problem and make inference about them using the data.

Simulation from the true posterior distribution cannot yet be achieved for such a complex model without a prohibitive number of iterations. Instead, the form of Markov chain Monte Carlo algorithm we are proposing achieves a 'restricted walk' over the posterior distribution. However, we shall illustrate, with examples, that the restricted walk visits a variety of 'good' trees. It therefore performs as an effective Bayesian-motivated stochastic search algorithm, providing a richer and improved output when compared with conventional CART. Instead of producing just one tree which is, in a sense, a 'point' estimate of the 'true' structure, as in the classical CART method, we produce a variety of tree structures together with relative weights. The Bayesian analogue of 'pruning the tree' (Breiman et al., 1984, Ch. 3) is achieved by putting a suitably chosen prior distribution over the number of terminal nodes in the structure, and, in the case of classification trees, a distribution over the classification of data points within each node. Details are given in § 2.2.

The problem of routine calculation of the posterior distribution of the tree structure is addressed by designing a Markov chain Monte Carlo reversible jump simulation algorithm.

2. THE BAYESIAN CART METHOD

2.1. Basic ideas

We propose a model which can be used to set up a probability distribution over the space of possible trees.

Any binary tree-based model can be uniquely defined by the splitting nodes present, the variables on which the nodes are split and the rules with which these splits are made. We define these variables respectively as s_i , s_i^{var} and s_i^{rule} ($i = 1, \dots, s_{\text{max}}$).

We use a simple labelling scheme to define uniquely the positions s_i of the split nodes. The root node, which is always in the model, is chosen to be the first split node and its position is labelled as position 1, so that $s_1 = 1$. Any descendant splitting node's position, s_i , say, is then uniquely defined given its parent's position, s_i^{parent} , say. This can be done by putting $s_i = 2s_i^{\text{parent}}$ if the node contains the data points for which the question at the parent node is true and $s_i = 2s_i^{\text{parent}} + 1$ otherwise. The positions of the terminal nodes are similarly defined but as they are completely determined by the tree structure given by the split nodes their positions do not need to be included in the model.

To illustrate the use of s_i^{var} and s_i^{rule} we give an example. Suppose the question at split node i is, 'Is $x_3 < 4.2$?'. Then the split variable $s_i^{\text{var}} = 3$ and the split rule variable $s_i^{\text{rule}} = 4.2$. An orientation for the split node is not required as any of the data points for which $x_3 < 4.2 = s_i^{\text{rule}}$ unambiguously go to the left descendant node with the rest going to the right.

We note that the number of terminal nodes present in the model is $k = (s_{\text{max}} + 1)$ so with the training data we can, depending on the type of problem, classify or assign a level to the data in each of the terminal nodes. We change the dimension of the model when we change k and so, for Bayesian computation, we use a reversible jump Markov chain Monte Carlo approach (Green, 1995) when we are considering changes in the number of terminal nodes in the tree structure.

Inference is carried out assuming that the true model is unknown but comes from the class of models M_1, M_2, \dots , where M_k denotes the model with exactly k terminal nodes, which implies $(k - 1)$ splitting nodes. The complexity of a tree structure is considered as just a function of the number of terminal nodes in Breiman et al. (1984), which is why we consider trees with the same number of terminal nodes as coming from the same 'class'. The topology of the tree structure is unimportant even though it would be possible to set up priors which penalise/encourage splits at specific locations to represent prior beliefs.

The overall parameter space Θ can then be written as a countable union of subspaces $\Theta = \cup_1^\infty \Theta_k$, where Θ_k is a subspace of the Euclidean space $R^{n(k)}$, where $R^{n(k)}$ denotes the $n(k) = 3(k - 1)$ dimensional parameter space corresponding to model M_k . Note that s_i and s_i^{var} are discrete. Here

$$\theta^{(k)} = (s_1, s_1^{\text{var}}, s_1^{\text{rule}}, \dots, s_{k-1}, s_{k-1}^{\text{var}}, s_{k-1}^{\text{rule}}).$$

There is a natural hierarchical structure to this set-up, which, denoting a generic element of Θ_k by $\theta^{(k)}$ and the data vector by y , we formalise by modelling the joint distribution of $(k, \theta^{(k)}, y)$ as

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

that is, as the product of model probability, parameter prior and likelihood.

Bayesian inference about k and $\theta^{(k)}$ will be based on the joint posterior $p(k, \theta^{(k)}|y)$, which we shall explore and summarise by regarding it as the target distribution for tailored Markov chain Monte Carlo computations. It will often be useful to consider this in the

factorised form

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

We will generate samples from the joint posterior of $(k, \theta^{(k)})$ by using a class of reversible jump Metropolis–Hastings algorithms (Green, 1995). Full details of the method can be found in the reference cited. Here, we focus on the essence of the methodology and the particular forms of the algorithms in our current context.

2.2. The Bayesian models

We set up Bayesian models for the distributions of the data points in the terminal nodes, using different forms for the regression tree and classification tree problems. For regression tree models let $\mathcal{T}_i = \{j: y_j \in t_i\}$ ($i = 1, \dots, k$), so that $\cup_{i=1}^k \mathcal{T}_i = \{1, \dots, n\}$. To control the parsimony of the trees we shall constrain our stochastic search to the class of trees for which each terminal node is assigned a minimal number of data points. We assume that the distribution of the data points at terminal node t_i is normal with an unknown mean, α_i , and unknown variance, σ_i^2 . The parameter vector, θ , is then extended to include the variables $\alpha_1, \sigma_1^2, \dots, \alpha_k, \sigma_k^2$.

It easily follows that the likelihood function, $p(y | \theta^{(k)})$ is given by

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

where $I(\cdot)$ is the usual 0–1 indicator function and $|\mathcal{T}_i|$ is the number of data points in terminal node i . Here T_{\min} is the chosen minimum number of data points assigned to each terminal node. This is commonly taken to be 5, as in the `tree.control` function in S-Plus (Becker, Chambers & Wilks, 1988). For illustration, in what follows we shall use this value for T_{\min} .

We use the limiting uniform prior for the α_i 's and, to avoid any problems with possible improper posteriors, a proper but vague prior, independent of the priors for the α_i 's, for the σ_i^2 's,

$$\pi(\alpha_i) = \text{const}, \quad \pi(\sigma_i^{-2}) = \text{Ga}(10^{-2}, 10^{-2}).$$

For classification trees, we assume that each data point y_i ($i = 1, \dots, n$) comes from a multinomial distribution (Clark & Pregibon, 1992). This leads to the likelihood

$$p(y | k, \theta^{(k)}) \propto \prod_{i=1}^k \left\{ I(|\mathcal{T}_i| > T_{\min}) \prod_{j=1}^N (p_{ij})^{m_{ij}} \right\}, \quad (2)$$

where N is the number of distinct classes, m_{ij} is the number of data points at terminal node t_i which are classified in category j and p_{ij} is the corresponding probability. Again we use a uniform prior for (p_{i1}, \dots, p_{iN}) ($i = 1, \dots, k$) so that

$$\pi(p_{i1}, \dots, p_{iN}) = \text{Dir}_{N-1}(p_{i1}, \dots, p_{iN} | 1, \dots, 1) \quad (i = 1, \dots, k).$$

In exploring the posterior distribution, our main focus is on moving around the interesting tree configurations. To accelerate this process and save on total computation time, we work with the means of the sampling distributions for the α_i 's and p_{ij} 's in (1) and (2) rather than drawing them. That is $\alpha_i = n_i^{-1} \sum_{j \in \mathcal{T}_i} y_j$ and $p_{ij} = m_{ij}/n_i$, where $n_i = |\mathcal{T}_i|$.

For the remainder of the prior specification we assume uniform distributions over the set of possible values for the splitting nodes present in the model and the splitting variable

chosen at each of these nodes. The splitting rule is assumed to be uniform over the set of all possible categories or, if the splitting variable is numeric, over the range of the variable we are splitting.

A Poisson distribution with parameter λ restricted to be greater than zero, that is a minimum of one terminal node, is used to specify the prior probabilities for each of the models, giving

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

In practice we restrict the tree to some large size so that k follows a truncated Poisson distribution where $k \leq 2^K$, where K is the maximum number of splits that can be made before a terminal node is reached.

2.3. Computational strategy

Our aim is to simulate samples from the joint posterior distribution of $p(\theta^{(k)}, k|y)$, since analytical or numerical analyses are totally intractable in this situation. For this purpose we design a reversible jump algorithm of the general type discussed by Green (1995), to which the reader is referred for details.

In the context of our problem, with multiple parameter subspaces of different dimensionality, it will be necessary to devise different types of move between the subspaces. These will be combined to form what Tierney (1994) calls a hybrid sampler, making random choice between available moves at each transition, in order to traverse freely around the combined parameter space.

We use the following rather natural transitions: (a) a change of the splitting variable at some node, (b) a change in the splitting rule at some node, (c) the addition of terminal node and (d) the deletion of a terminal node. Note that in steps (c) and (d) we are changing the dimension of the model.

Given the splitting nodes present in the model, steps (a) and (b) are quite straightforward. In both steps, we first uniformly choose a splitting node, s_j ($1 \leq j \leq k-1$), say. For step (a) we then choose a new variable, x_i , say, to split on at node s_j and a new splitting rule from those possible on variable x_i . For step (b) we need only to find a new splitting rule on the current splitting variable. For a numeric variable, the splitting rule is of the form ' $x_i > c$?', where $c \in (\min_i x_{i,l}, \max_i x_{i,l})$, whereas for a categorical variable it is of the form ' $x_i \in C$?', where $C \subset \{\text{possible values of } x_i\}$. In practice for a numeric variable we split up the range of each predictor into a number of grid points so there is a finite number of possible tree structures. Hence choosing a new splitting rule is just a matter of choosing a c or C from their possible values. These changes are easily undertaken using a Metropolis step (Metropolis et al., 1953; Hastings, 1970) to accept or reject the proposed new state.

Step (b) is, in fact, only a special case of step (a) but we have found that including both steps increases the mixing in the sampler and this improves the overall results. Not including step (b), which only proposes 'small' local moves, hinders the sampler from finding local modes which give good tree structures. However, including step (b) by itself leads to the sampler getting stuck in these local modes which is why the 'large' moves proposed by step (a) are needed.

The addition of a terminal node, step (c), is carried out by uniformly choosing one of the terminal nodes present in the model, t_j , say, $1 \leq j \leq k$. We then put a splitting

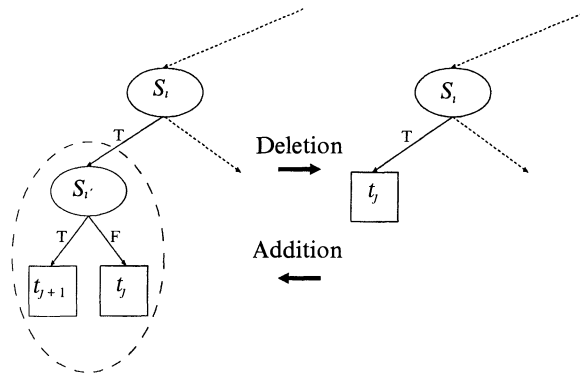


Fig. 1. Addition and deletion of terminal nodes which are allowed in reversible Markov chain Monte Carlo algorithm; T, true; F, false.

node in place of the chosen terminal node with splitting variable and rule determined as in step (a) with both branches ending in terminal nodes. We then have a tree with one more terminal node than was previously present; see Fig. 1.

The only allowable deletions, step (d), are at positions of the type shown by the dotted ellipse in Fig. 1. This step is chosen to ensure that the chain is irreducible. We choose such a site uniformly among those present in the current tree structure. It is easy to show that there is always at least one such site. Consequently the tree cannot move too quickly about the space of possible trees as we cannot delete more than one terminal node at a time. Bolder moves which may encourage better mixing in the posterior sample are not used. Allowing large branches to be pruned would require a converse step which would grow a similar branch in one move. Neither move type would be accepted often because of the large change in the tree structure and so we do not include them.

If we assume that maximum likelihood estimates are used throughout then there is nothing more to sample from in the classification tree case but in the regression tree case we need to update the σ_i 's. This is done straightforwardly using a Gibbs step (Gelfand & Smith, 1990) as we know the full conditionals of the σ_i 's which are

$$(\sigma_i^{-2} | y, \theta, \sigma_{-i}^2) \sim \text{Ga} \left(10^{-2} + \frac{n_i}{2}, 10^{-2} + \frac{1}{2} \sum_{j \in \mathcal{I}_i} (y_j - \alpha_i)^2 \right),$$

where the $-i$ subscript denotes all those elements which do not have index i . Thus, at each full step of the algorithm we obtain a sample of $(k, s_{(k)}, s_{(k)}^{\text{var}}, s_{(k)}^{\text{rule}}, \sigma_1^2, \dots, \sigma_k^2)$ for the regression case and just $(k, s_{(k)}, s_{(k)}^{\text{var}}, s_{(k)}^{\text{rule}})$ in the classification case.

2.4. Algorithm

In the reversible jump algorithm we use the four move types described above so that we can write the set of moves as $m = \{V, R, 1, 2, \dots\}$. Here V refers to changing a splitting variable, R refers to changing a splitting rule and $m = 1, 2, \dots$ refers to increasing the number of terminal nodes from m to $m + 1$ or decreasing it from $m + 1$ to m . Independent move types are randomly chosen with probabilities v_k for $m = V$, ρ_k for $m = R$, b_k for $m = k$ and d_k for $m = k - 1$ which satisfy $v_k + \rho_k + b_k + d_k = 1$ for all k . In this problem we took

$$v_k = \rho_k, \quad b_k = 2c \min \{1, p(k+1)/p(k)\}, \quad d_{k+1} = c \min \{1, p(k)/p(k+1)\},$$

with the constant c as large as possible subject to $b_k + d_k \leq 0.75$ for all $k = 2, 3, \dots$. For $k = 1$ we put $b_1 = 1$, $d_1 = v_1 = \rho_1 = 0$. We choose b_k so that it is twice as big as d_k when $\lambda = k + 1$ to compensate for the fact that the birth step often proposes a tree which has fewer data points in one of the terminal nodes than we allow and we want acceptable births and deaths to be proposed at a similar rate.

If we use the notation of Green (1995), the acceptance probability of a birth step is just

$$\alpha = \min \{1, (\text{likelihood ratio}) \times (\text{proposal ratio}) \times (\text{prior ratio})\}. \quad (3)$$

For the steps which involve a change in a splitting variable or a splitting rule the acceptance probability, α , is very simple as the last two ratios in (3) cancel each other out, but for the moves which change dimension it is not so straightforward. After simple calculations we find that for the birth step the proposal multiplied by the prior ratio equals $(k_{\text{die}} + 1)/k$, where k_{die} is the number of possible locations for a death in the current model. For a death step this fraction is inverted.

The algorithm we use is very simple and works quickly.

ALGORITHM

1. Start with a tree with no splitting node present.
2. Set k equal to the number of terminal nodes in the present tree.
3. Generate u uniformly on $[0, 1]$.
4. Goto move type determined by u .
 If $(u \leq b_k)$ then goto Birth step;
 else if $(b_k < u \leq b_k + d_k)$ then goto Death step;
 else if $(b_k + d_k < u \leq b_k + d_k + v_k)$ then goto Variable step;
 else goto Rule step.
5. If performing a regression problem draw the error variances σ_i^2 ($i = 1, \dots, k$).
6. Repeat 2–5 until there is little change in the posterior probability of the tree structures.

3. EXAMPLES

3.1. Regression tree

We first illustrate our methodology to produce a regression tree. We use data from a study by Bruntz et al. (1974) of the dependence of ozone on some meteorological variables on 111 days from May to September 1973 at sites in the New York metropolitan area. As in Cleveland & Devlin (1988) we work with the cube root of ozone. This dataset is known as ‘air’ and is available in S-Plus (Becker et al., 1988). For illustration we focus on wind speed and temperature. We initially remove a smooth effect for radiation and fit the tree to the partial residuals as a function of temperature and wind. A similar approach was adopted in Hastie & Tibshirani (1990, pp. 271–4). The data were obtained from S-Plus as follows.

```
y <- gam(ozone ~ lo(radiation) + lo(wind, temperature), data=air)
y1 <- (y$y - y$smooth[, 1])
data <- cbind(y1, air[, 3], air[, 4])
```

We put a Poisson prior over k with $\lambda = 10$ and put $\sigma = \sigma_1 = \dots = \sigma_k$ so the within-node variances are assumed to be equal. Initially we start with no splitting nodes, as described in the algorithm. We take our results from the final 40 000 iterations of the process after an initial burn-in period of 10 000 iterations, after which the posterior probabilities of the

tree structures have been settled for some time. We used 92 split points for the wind predictor and 40 for temperature. These values were chosen so that there was a possible split point between each value of the predictor as wind was given to one decimal place and was on the range [2.3, 20.7], and temperature was integer valued on [57, 97].

At the beginning of this burn-in period we restrict the tree from having more than 6 terminal nodes to assist it in finding good initial splits to the data. This is required because otherwise the tree grows quickly and then proposed changes in split variables or rules near the root node are extremely unlikely to be accepted. This is the case because the Poisson prior for the number of terminal nodes does not often allow the tree to become very small and only when the tree is small are good changes in split rules or variables possibly accepted because the rest of the tree was grown conditionally on the current structure. Throughout the process we do not allow the tree to have terminal nodes with fewer than 5 members.

Table 1. *Posterior probabilities of models for regression tree*

Number of terminal nodes	Posterior probability	RSS of mode	RSS of CART
5	0.004	25.02	—
6	0.040	24.70	26.71
7	0.108	23.34	25.56
8	0.209	23.13	—
9	0.213	20.18	—
10	0.170	22.90	23.42
11	0.112	17.82	23.06
12	0.107	16.90	22.80
13	0.032	17.32	25.56
14	0.005	16.30	22.35

RSS, residual sum of squares; CART, classification and regression tree method.

The posterior probabilities of the number of terminal nodes are given in Table 1 and we display the posterior modal tree structure, which had 12 terminal nodes, in Fig. 2. The residual sum of squares, RSS, of the modal trees found by the Bayesian method and the standard CART method are also given in Table 1. These were calculated using the equation

$$\text{RSS} = \sum_{i=1}^k \sum_{j \in \mathcal{T}_i} (y_j - \bar{y}_{t_i})^2,$$

where \bar{y}_{t_i} is the response mean in terminal node t_i . We used the S-Plus functions `tree` and `prune.tree` to find the standard CART structure (Clark & Pregibon, 1992) and we display the result with the same number of terminal nodes as our modal structure in Fig. 3. Note that the stepwise backward deletion algorithm using `prune.tree` does not find trees with every number of nodes, which is why there are some missing values in Table 1.

In Fig. 4 we present a boxplot of the sums of squares of the trees in our sample, and in Fig. 5 we show the densities of these sums of squares for the nodes that had a posterior probability greater than 0.1. As can be seen our modal estimate is usually below the lower quartile of the trees for each number of terminal nodes and the densities appear to be multimodal. The modes coincide with similar tree structures with the same number of

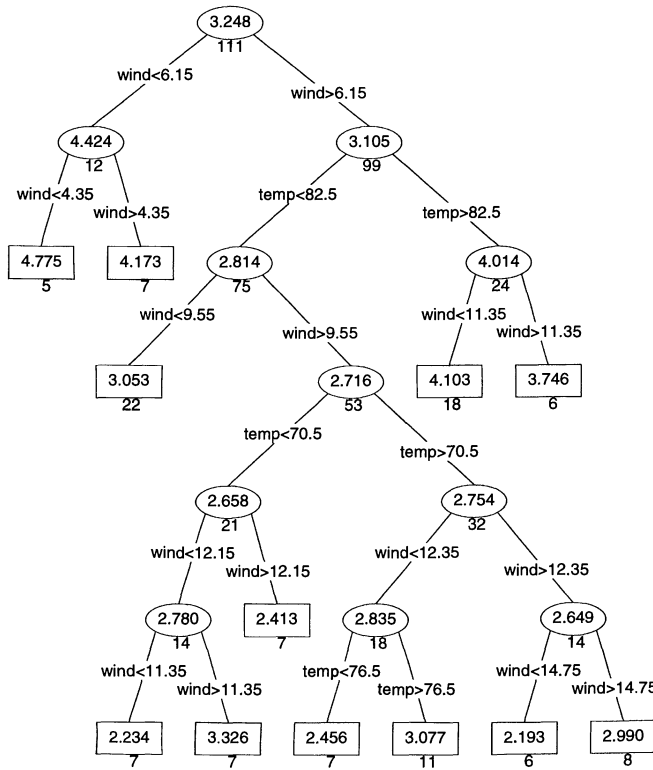


Fig. 2. Posterior modal tree structure obtained from ozone data. The mean response is inscribed and the number of occupants is given below each terminal node.

terminal nodes. It would appear that although the sampler does not mix well enough to give us a full posterior distribution it does not get stuck in one local mode but visits others which have very different residual sums of squares so that sufficient mixing does take place for plenty of ‘good’ trees to be visited.

3.2. Classification tree

We illustrate our method of producing a classification tree using the ‘kyphosis’ dataset which is also available in S-Plus (Becker et al., 1988). It is a binary dataset which consists of measurements on 81 children after corrective spinal surgery. The response indicates the presence or absence of a postoperative deformity known as kyphosis and the predictor variables are the age of the child in months (Age), the number of vertebrae involved in the operation (Number) and the beginning of the range of the vertebrae involved (Start). Out of the 81 data points 17 had kyphosis present after the operation.

Again we took $\lambda = 10$ for this example but took our results from only 20 000 iterations of the sampler because of the simpler nature of this problem compared to the previous regression tree example. As before we restrict the tree from having fewer than 5 members and prevent it from growing too large for a short time at the beginning of the burn-in period. The number of split points for each predictor was chosen in the same way as before, so we used 205 for Age, 8 for Number and 17 for Start.

The posterior probabilities for the number of nodes are given in Table 2. Also in Table 2

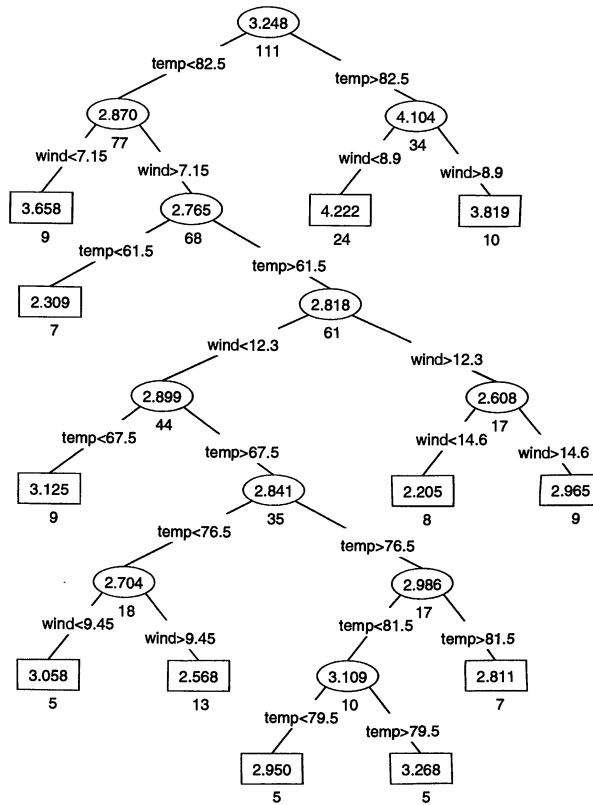


Fig. 3. Tree grown for ozone data using the standard CART method. The mean response is inscribed and the number of occupants is given below each terminal node.

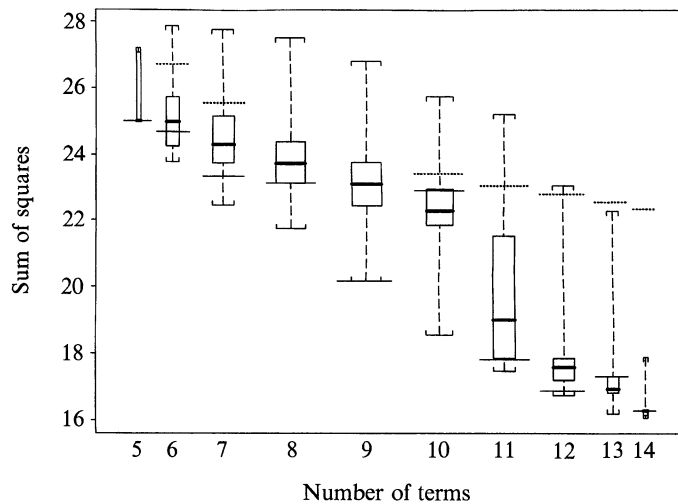


Fig. 4. Boxplot giving the sums of squares of the generated sample of tree structures. The solid and dotted horizontal lines give the sums of squares of the Bayesian CART modes and the standard CART trees for each class respectively. The widths of the boxes are related to the number of samples generated in each class.

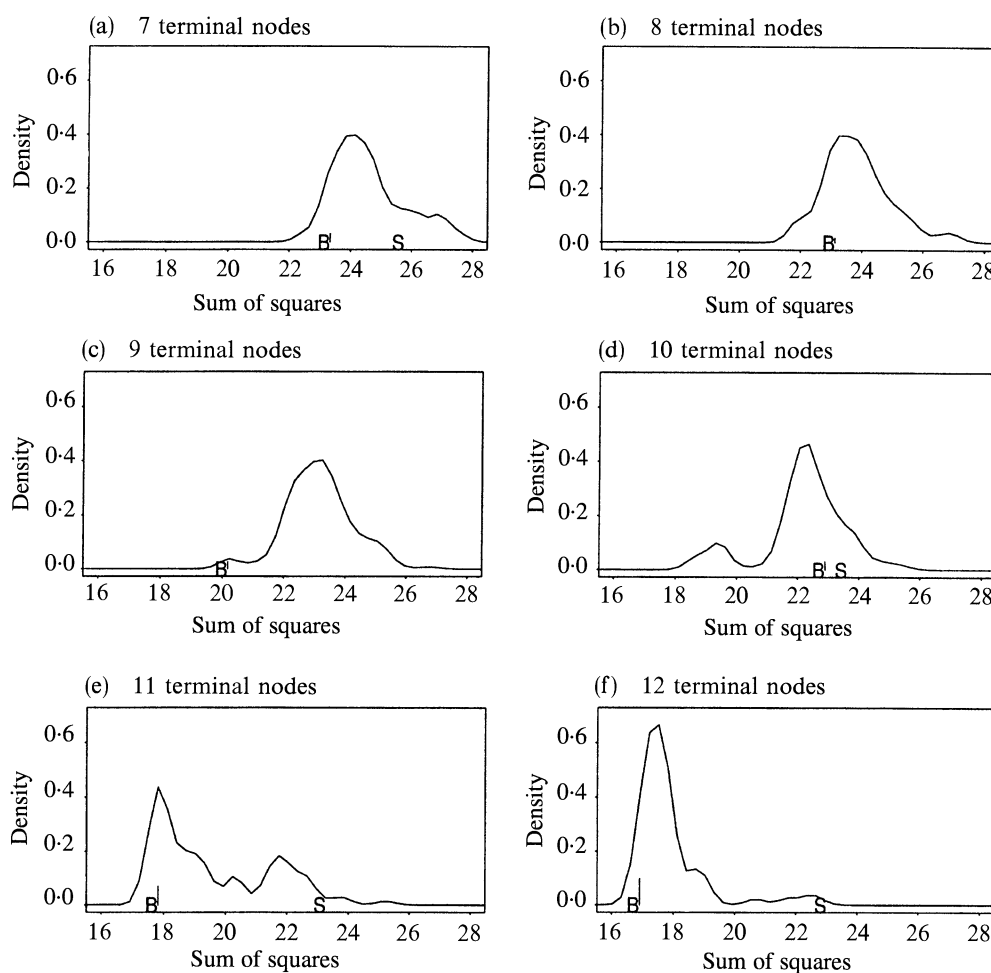


Fig. 5. Densities of the sums of squares of the trees in each class. The 'B' and 'S' give the sum of squares of our modal tree and the standard CART respectively, and the vertical solid lines give the posterior probabilities of the modal trees conditioned on the number of nodes.

we display the deviances of the modal structures obtained for each number of terminal nodes using the Bayesian algorithm and those found using standard CART. The deviances for each tree are easily found as they are just minus twice the loglikelihood given in equation (4).

Table 2. Posterior probabilities of models for classification tree

Terminal nodes	Posterior probability	Deviance of mode	Misclassified by mode	Deviance of CART	Misclassified by CART
4	0.001	54.39	17	53.11	14
5	0.024	56.07	13	—	—
6	0.111	50.63	13	47.88	11
7	0.300	31.80	8	—	—
8	0.344	31.80	8	43.94	11
9	0.169	33.54	8	42.44	11
10	0.051	45.35	11	41.24	11

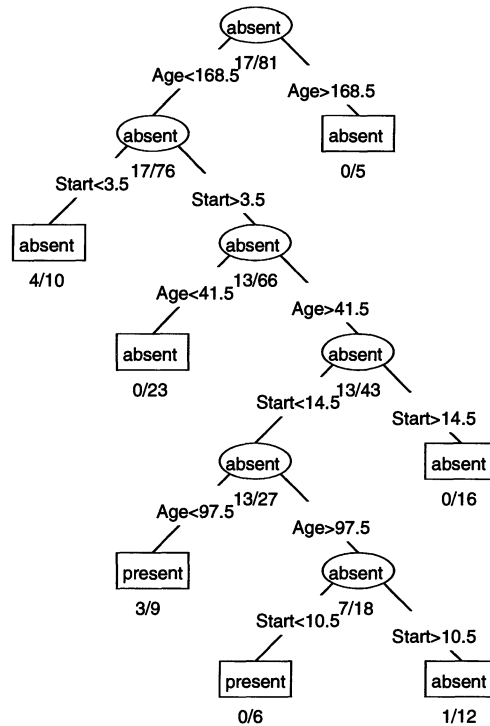


Fig. 6. Posterior modal tree structure obtained from kyphosis data. The classification and fraction misclassified are shown inside and below each node respectively.

In Fig. 6 we show the Bayesian CART modal tree structure, which had 7 terminal nodes, and in Fig. 7, for comparison, we display the standard CART with 8 terminal nodes, as the stepwise backward deletion process does not yield a tree with 7 nodes.

From Table 2 we can see that, as before, the modal Bayesian CART trees outperform standard CART in all the cases with significant posterior weight except for the structure with 6 terminal nodes. However, the modal structure with 6 terminal nodes is hardly supported in the posterior and is only the 16th most probable tree structure.

During our investigations we noted that the densities of the deviances, conditional on the number of terminal nodes, were multimodal. This is to be expected as the deviance of a tree is greatly influenced by the number of data points it classifies correctly. This suggests that for each number of terminal nodes the algorithm visits trees that misclassify differing numbers of points and does not get stuck in local modes. For example, in the 9 terminal node case, trees with significant posterior weight were found that misclassified 8, 9, 10 and 11 points, but, as we can see from Table 2, the mode misclassified only 8 points.

4. DISCUSSION

We have presented a stochastic search method motivated by Bayesian models for finding classification and regression trees for various datasets. The method provides output which give insight into a range of plausible 'good' tree structures rather than delivering a 'point estimate'.

A fully Bayesian CART, sampling from the entire posterior, seems currently infeasible,

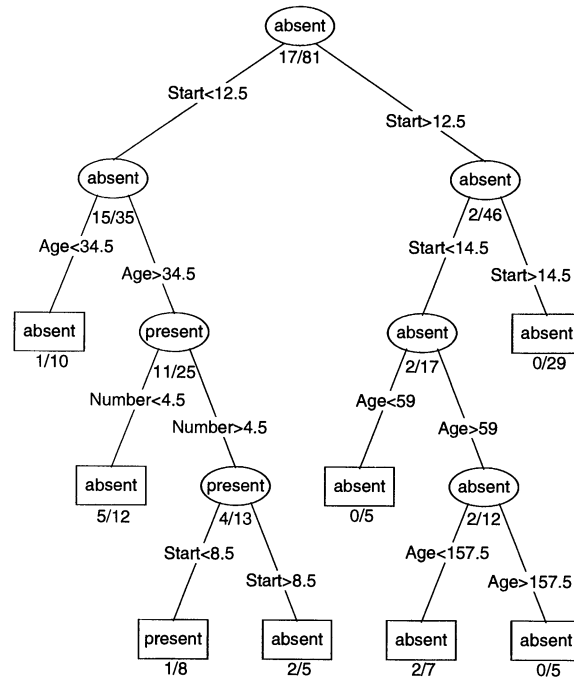


Fig. 7. Tree grown for kyphosis data using the standard CART method. The classification and fraction misclassified are shown inside and below each node respectively.

given such a large parameter space with rigid hierarchical structure and since the move types we employ must satisfy the reversibility condition.

We restrict the tree from growing indiscriminantly at the beginning of the burn-in period to find reasonable initial splits as these are unlikely to be changed later on in the run of the chain. We can see this by noting how rarely the sampler visits trees with very few terminal nodes; see Tables 1 and 2. However, the output from the sampler, even though it varies from run to run because of the different initial splits, produces trees which are as good as can be achieved with the given initial splits. Thus, to increase the efficiency of the algorithm we do not allow it to find these initial splits blindly but allow it some time to locate them by restricting the tree's growth at the beginning of the burn-in period.

The posterior modal structures are usually below the lower quartile of the tree structures for each number of terminal nodes. In the cases where the modal structure is not below the lower quartile the tree structure it proposes has little weight in the posterior and no good structure for that number of nodes is found, but the modal structure is still, invariably, in the bottom half of all the structures. Another reason for using the modes is that they are easy to find amongst the large number of tree structures that are in the output of the algorithm.

Posterior averaging over the regression surfaces formed by the trees is possible but if surface estimates are required there are many better methods for doing this. This type of averaging obviously does not produce a tree structure so the strengths of the CART method, in particular its 'rule-based' appeal to many non-statisticians, are immediately lost. We can think of prediction with tree structures as similar to prediction of an integer-valued random variable. Finding the mean is not useful as it is typically not in the event space

of the variable and we do not want to concentrate on the median as we do not want to find average structures but good ones, which is why the mode is used.

We could have chosen the tree with the lowest sum of squares or deviance, depending on the context, but the posterior might assign little weight to such trees and so it would be difficult to judge which would be the 'best' estimate among those with different numbers of nodes.

Another 'Bayesian' CART approach can be found in Chipman, George & McCulloch (1998). However, their approach concentrates more on the prior specification of the tree structure which can be used to encourage trees to grow with specific topologies. Furthermore, they allow changes in splitting rules only at split nodes which have two terminal nodes below them and terminal nodes, when they are grown, are classified as 'splittable' or 'not splittable' with some probability which depends on the depth of the node and some preset parameters. Although this type of prior specification of tree structures may be beneficial in a few examples, we rarely know a priori what type of tree structure we expect, so that we prefer to place no constraint on the structure and let the data speak for themselves. Chipman et al. (1998) concede that the lack of flexibility in the way that split moves and changes in splitting rules are made can cause the tree to become stuck in local modes. Our algorithm alleviates these problems by always allowing terminal nodes to be split again and not just allowing changes in splitting rules to occur at split nodes with two terminal nodes as direct descendants. However, even though we believe our move structure produces a great deal more mixing than the Chipman et al. approach, both algorithms have a stochastic search flavour rather than providing a full exploration of the posterior.

One important advantage we believe our algorithm to have is that the greater flexibility of the move steps leads to useful results every time we run it. In contrast to this Chipman et al. (1998) need many restarts of their algorithm resulting in a very much greater computational burden: for instance, the results of the example in Chipman et al. required running the chain for 3000 iterations, 30 000 times. The examples in this paper took between 15 and 20 minutes to produce on a Sun Sparc 3.

ACKNOWLEDGEMENT

The work of D. G. T. Denison was supported by an Engineering and Physical Sciences Research Council research studentship. We acknowledge the helpful comments given by the referee and editor.

REFERENCES

- BECKER, R., CHAMBERS, J. M. & WILKS, A. (1988). *The New S Language*. Belmont, CA: Wadsworth.
- BREIMAN, L., FRIEDMAN, J. H., OLSHEN, R. & STONE, C. J. (1984). *Classification and Regression Trees*. Belmont, CA: Wadsworth.
- BRUNTZ, S. M., CLEVELAND, W. S., KLEINER, B. & WARNER, J. L. (1974). The dependence of ambient ozone on solar radiation, temperature and mixing height. In *Symposium on Atmospheric Diffusion and Air Pollution*, pp. 125–8. Boston, MA: American Meteorological Society.
- CHIPMAN, H., GEORGE, E. I. & MCCULLOCH, R. E. (1998). Bayesian CART model search (with Discussion). *J. Am. Statist. Assoc.* **93**. To appear.
- CLARK, L. A. & PREGIBON, D. (1992). Tree based models. In *Statistical Models in S*, Ed. J. M. Chambers and T. J. Hastie, pp. 377–420. Pacific Grove, CA: Wadsworth.
- CLEVELAND, W. S. & DEVLIN, S. J. (1988). Locally-weighted regression: an approach to regression analysis by local fitting. *J. Am. Statist. Assoc.* **83**, 597–610.

- GELFAND, A. E. & SMITH, A. F. M. (1990). Sampling-based approaches to calculating marginal densities. *J. Am. Statist. Assoc.* **85**, 398–409.
- GREEN, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* **82**, 711–32.
- HASTIE, T. J. & TIBSHIRANI, R. J. (1990). *Generalized Additive Models*. London: Chapman and Hall.
- HASTINGS, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika* **57**, 97–109.
- METROPOLIS, N., ROSENBLUTH, A. W., ROSENBLUTH, M. N., TELLER, A. H. & TELLER, E. (1953). Equations of state calculations by fast computing machines. *J. Chem. Phys.* **21**, 1087–91.
- TIERNEY, L. (1994). Markov chains for exploring posterior distributions (with Discussion). *Ann. Statist.* **22**, 1701–62.

[Received March 1996. Revised September 1997]