Università Bocconi

MSc in Data Science and Business Analytics
Major in Data Science

Tempered Stochastic Search of Bayesian CART Models

Master Dissertation

Gattiglio, Guglielmo
Supervisor: Zanella, Giacomo

2021

Contents

Chapter 1

Introduction

In this work we focus on the Bayesian extension of classification and regression trees (CART) models as introduced by Chipman, George, et al. (1998), using tempering to address the computational challenges involved.

CART are flexible, easy to interpret models that have seen wide popularity after their introduction by Breiman et al. (1984). Their tree structure allows for simple graphical representation and they are easily summarized by a sequence of logical questions 'Is x smaller than y?'. Although able to achieve good prediction accuracy, they are most often used for inference tasks where more accurate but less transparent models cannot be employed. Trees are generally fit using a greedy procedure to decide the best split at each node given the current structure of the tree, which tends to overfit the training data. To account for that, Breiman et al. (1984) proposed a tree pruning procedure that seeks to find the optimal subtree of the grown tree, according to some metric that will be made precise in the following chapter. Hence, in the original formulation, the process is greedy and deterministic, which makes it impossible for the algorithm to undo poor splitting choices at some point in the tree. Moreover, in established fields characterized by considerable existing knowledge, these models do not easily allow to incorporate insights and further information. To account for these shortcomings, Chipman, George, et al. (1998) and Denison, Mallick, et al. (1998) presented a Bayesian extension of CART models. The statistical structure of CART is enhanced to include a prior distribution over the space of trees, which can be designed to incorporate prior information at the model specification phase, used to guide the fitting procedure. Together with the model likelihood, it induces a posterior distribution over the space of trees, consisting of the combination of prior information as coded in the prior, and evidence coming from the data. Such statistical object reveals to be difficult to handle as it cannot be obtained in closed form and hence we must resort to sampling in order to explore the posterior, by relying on Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953, Hastings, 1970). This algorithm explores the space using a stochastic approach, starting from an initial tree and attempting to change parts of it every time step, accepting the change with a

certain probability that depends in part on the posterior distribution of the original and modified tree. By relying on an informed stochastic search rather than a deterministic methodology, Bayesian tree models have the potential to escape adverse situations, such as the simulated example of Section 3.5. Moreover, the posterior distribution can be used to inspect differences between competing trees in terms of posterior probability and to provide confidence intervals for tree (not for tree's predictions), something which is not easily done for CART where just a single tree (point estimate) is produced as a result. However, all these benefits come with a major drawback: due to the multimodality of the posterior distribution, computational issues arise during sampling since MH takes an infeasible amount of time to move between peaks, failing to reach convergence in all practical applications. Although theoretically beautiful, the usefulness of Bayesian CART would sensibly decrease without a proper way to sample from the posterior distribution. Existing literature has addressed this issue either by focusing on a stochastic search for the best model, by randomly restarting the algorithm several times and comparing trees across modes, or by developing more efficient MH proposal distributions that improve space exploration (see Chapter 3 introduction for a more detailed treatment with references). In this work we use Metropolis-coupled Markov chain Monte Carlo ($(MC)^3$, Geyer, 1991, Gilks and Roberts, 1996), also known as 'tempering', to address the same issue. The underlying idea behind this technique is simple: enrich the MH algorithm by adding a number of chains targeting a modified, flatter version of the original posterior distribution where space exploration is easier (peaks are flattened out). Then attempt to exchange states between chains, so that the one acting on the original posterior is better able to escape modes. Leveraging insights from the model, we introduce a novel, ad-hoc tempering procedure for Bayesian CART that exploits the structure of the model to produce a more computationally efficient sampler, thereby improving mixing. Preliminary results suggest the speed-up is obtained by avoiding sampling trees in low-probability valleys, which is instead done by $(MC)^3$'s modified chains in order to move between peaks. Even though both samplers exhibit improved mixing compared to the single-chain MH procedure, they fail to converge to the posterior distribution for the simulated example considered, showing that further work is needed.

This work is organized as follows: in Chapter 2 we review CART models, detailing their structure and components, fitting and pruning procedures, setting the notation that will be used throughout. Chapter 3 introduces Bayesian CART, specifies the prior distribution and the algorithm used for exploring the posterior distribution, concluding with a simulated example revealing the computational issues previously mentioned. Chapter 4 describes $(MC)^3$ from the theoretical point of view and presents the ad-hoc tempering procedure designed for Bayesian trees. Preliminary results for the simulated example are commented in Chapter 5. Chapter 6 provides a recap and suggests further research directions.

Chapter 2

Classification and Regression Trees

We devote this chapter to the presentation of classification and regression trees as first introduced by Breiman et al. (1984). We start by recalling the statistical learning framework within which such models are used. Focusing on the regression case at first, we describe the model structure, the main components of trees and the intuition behind. Sections 2.2.1 and 2.2.2 will detail the fitting procedure and pruning process respectively. We then conclude the chapter by considering their extension to the classification case.

2.1 Statistical Learning Framework

Suppose we observe a quantitative response $y \in \mathbb{R}$ and p predictors $x_1, x_2, ..., x_p \in \mathbb{R}$ and assume that there is some dependence between y and $\boldsymbol{x} = (x_1, x_2, ..., x_p) \in \mathbb{R}^p$:

$$y = f(\boldsymbol{x}) + \epsilon$$

where ϵ is a random error term with zero mean. Knowledge of $f(\cdot)$ would allow to:

- Predict y when $\boldsymbol{x}$ is readily available but y is not, in such cases where the object of interest is y itself;
- Infer the relationship between y and $\boldsymbol{x}$, for example to understand causal effects of predictors on the response y and be able to manipulate the former to achieve a certain desired effect on the latter.

CART models can be used to address both questions, however they are usually preferred for the latter one, since their structure makes them easy to interpret compared to other machine learning models, even by people not familiar with this discipline.

Irrespective of the objective, the task is the same: obtain $\hat{f}(\cdot)$, an estimate of $f(\cdot)$. This can be accomplished by finding the function f which minimizes the expected loss:

$$E\{L[y, f(\boldsymbol{x})]\} = \int L[y, f(\boldsymbol{x})]\, p(y, \boldsymbol{x})\, dyd\boldsymbol{x}$$

for some loss function $L[y, f(\boldsymbol{x})]$ which evaluates the quality of the predictions. A couple of notable cases:

- If y quantitative, under squared error loss, $L[y, f(\boldsymbol{x})] = [y - f(\boldsymbol{x})]^2$, the solution is (Hastie et al., 2001, pp. 32)
$$f(\boldsymbol{x}) = E[y|\boldsymbol{x}]$$
- If y qualitative, setting $L[y, f(\boldsymbol{x})] = 1$ if y is mis-classified, and 0 otherwise, we obtain (Hastie et al., 2001, pp. 34-35):
$$f(\boldsymbol{x}) = argmax_l \; p(y = l|\boldsymbol{x})$$

2.2 Regression Trees

Let us assume for now that $y \in \mathbb{R}$, so that we are in the quantitative scenario previously mentioned. Our goal is then to build a model for the conditional expectation of y given $\boldsymbol{x}$, $E[y|\boldsymbol{x}]$. Tree-based methods approach this problem by relying on partitions of the predictor space into a set of smaller, non-overlapping regions. The intuition is that, globally, the regression function $f(\cdot)$ may be arbitrarily complex due to interactions and non-linearities in the predictors, but that upon segmenting the feature space, these become more manageable and simpler models can be used to approximate its value in each region. Formally, a regression tree T is made of two components:

- A partition of the sample space into b regions $R_1, R_2, ..., R_b$. These will also be referred to as terminal regions, terminal nodes, leaves or leaf nodes.
- A predictive rule $\hat{y}_{R_i}$ for each region R_i, $i = 1, ..., b$.

For regression trees, we shall consider $\hat{y}_{R_i}$ to be constant and equal to the average value of the response in that region, $\hat{y}_{R_i} = \bar{y}_i$. Note that in principle other forms for $\hat{y}_{R_i}$ could be chosen, such as a linear model or a more flexible one; here we choose to follow the approach of Breiman et al. (1984) throughout. Similarly, the regions could take on any shape, however we restrict ourself to considering only high-dimensional rectangles, or boxes, which are easy to interpret and simplify drastically the tree building procedure.

2.2.1 Tree Fitting Procedure

Let

$$\begin{aligned} (\boldsymbol{x}, y)_{ij} \quad & i = 1, ..., b \qquad y \in \mathbb{R} \\ & j = 1, ..., n_i \quad \boldsymbol{x} \in \mathbb{R}^{\mathrm{p}} \end{aligned}$$

be a collection of observations, n be the total number of observations $n = \sum_{i=1}^{b} n_i$, where n_i is the number of observations falling into region R_i, so that $y_{ij} \in R_i$, and assume a squared error loss function. As remarked before, the regression tree building process consists of two main steps:

- Divide the sample space into b distinct, non-overlapping, p-dimensional rectangles
- Given the terminal regions, compute the mean of the response in each region and set $\hat{y}_{R_i} = \bar{y}_i$.

Therefore, it boils down to finding a set of boxes $R_1, ..., R_b$ that minimize:

$$RSS = \sum_{i=1}^{b} \sum_{j=1}^{n_i} (y_{ij} - \hat{y}_{R_i}(\boldsymbol{x}_{ij}))^2$$

This is in general not computationally feasible, due to the sheer number of potential partitions. Note that the problem is still infeasible even if one decides to consider a discretization of the predictors to their observed values, so that the number of possible partitions becomes finite. The solution proposed by Breiman et al. (1984) is a top-down greedy approach known as recursive binary splitting, which is detailed in Algorithm 1. As it can be seen, the procedure start from a tree consisting of only one terminal node and recursively applies binary splits of the data at the previous node to identify smaller regions where the RSS is smaller, until a stopping criterion is met. The choice for the best split at each step is made in a greedy way, considering only local information (parent and children nodes' RSS), rather than selecting splits that will lead to a lower RSS in some future step. We define a split as a pair formed by a predictor x_l and corresponding cutpoint s which divides a given region, or node, into two regions R_L and R_R, defined as follows:

$$R_L = \{\boldsymbol{x} : x_l < s\} \quad \text{and} \quad R_R = \{\boldsymbol{x} : x_l \geq s\}$$

A few remarks are in order:

- Potential stopping criteria can include limits on the maximum depth of the tree, on the maximum number of terminal regions, or set a minimum number of observations that must fall in each regions, after which that node cannot be split further. Additionally, another possibility is to grow the tree as long as the decrease in the RSS exceeds some predetermined threshold.

Algorithm 1: Recursive Binary Splitting Procedure

1 Start with a tree consisting of only the root node
2 Pick a node η from the set of terminal nodes
3 Select split (x_l, s) which minimizes

$$\sum_{ij:x_{ij}\in R_L} (y_{ij} - \hat{y}_{R_L})^2 + \sum_{ij:x_{ij}\in R_R} (y_{ij} - \hat{y}_{R_R})^2$$

where R_L and R_R denote respectively the left and right regions generated by the split, and $\hat{y}_{R_L}$, $\hat{y}_{R_R}$ the predictions
4 Split node η by applying split (x_l, s) and generate two new terminal regions
5 Repeat steps 2-4 until a stopping condition is met.

- Regression trees as defined until now do not allow categorical predictor variables to be used in the model fitting. Including them is straightforward as it requires no modification to the tree itself, just the definition R_L and R_R for the categorical split:

$$R_L = \{\boldsymbol{x} : x_l \in A\} \quad \text{and} \quad R_R = \{\boldsymbol{x} : x_l \notin A\}$$

where A is any subset of observed values for the predictor x_l.

- The RSS at a parent node is always greater than or equal to the sum of its children's RSS:

$$\sum_{ij:x_{ij}\in R} (y_{ij} - \hat{y}_R)^2 \geq \sum_{ij:x_{ij}\in R_L} (y_{ij} - \hat{y}_{R_L})^2 + \sum_{ij:x_{ij}\in R_R} (y_{ij} - \hat{y}_{R_R})^2$$

where R denotes the region of the feature space identified by the parent node, and R_L, R_R those of the left and right children respectively. Hence it is clear that a procedure purely based on minimizing RSS always prefers a bigger tree (Breiman et al., 1984, p. 232, see also pp. 95-96 of the same for a proof in the classification trees case).

The last remark highlights an important issue: by giving preference to bigger trees, Algorithm 1 will grow trees that overfit the data. This usually causes high prediction accuracy on the training set but poor performance out of sample, because the tree is likely to end up with several terminal nodes with only a handful of observations in them. This results in very small terminal regions, yielding in principle a more precise estimate of the regression function (lower bias), at the expense of greater variability due to the low number of observations the estimate and the fitted tree structure is based upon Berk (2008, p. 117). A first possibility to limit this phenomenon has already been considered among the stopping criteria, namely to prevent splitting leaves with too few observations. However this strategy alone is not enough: it is useful to remove specific branches by

combining nodes that do not reduce the RSS sufficiently for the extra complexity added. This procedure is called cost complexity pruning, or weakest link pruning, and is detailed in the coming section.

2.2.2 Pruning

Let us start by extending our notation. So far we have mostly referred to regions, however it is important to remark the relationship between regions and nodes: every node η of a tree T, whether internal or terminal, represents a region of the sample space. Hence saying that an observation $\boldsymbol{x}$ belongs to the terminal region R, $\boldsymbol{x} \in R$, is equivalent to stating that $\boldsymbol{x}$ belongs to the node η, $\boldsymbol{x} \in \eta$. With this relation in mind, we consider the following simple definitions:

- A node $\eta' \in T$ is a descendant of a node $\eta \in T$ if there is a connected path down the tree leading from η to η'.
- A branch T_η of T with root node $\eta \in T$ consists of the node η and all descendants of η in T. Note that a branch is itself a tree.
- Pruning a branch T_η from a tree T consists of deleting from T all descendants of η, that is, cutting off T_η except its root node.
- If T' is obtained from T by sequentially pruning off branches, then T' is called a pruned subtree of T and denoted by $T' < T$.
- T_{max} is an overgrown tree obtained by overfitting the training data.
- $\tilde{T}$ is the set of terminal nodes of tree T.
- $RSS(\eta)$ is the residual sum of squares at a node η:

$$RSS(\eta) = \sum_{ij:x_{ij} \in \eta} (y_{ij} - \bar{y}_\eta)^2$$

- $RSS(T_\eta)$ is the residual sum of squares for the branch T_η:

$$RSS(T_\eta) = \sum_{\eta' \in \tilde{T}_\eta} R(\eta')$$

We seek to find a subtree of T_{max} which is optimal in some sense, without the need to enumerate all its possible subtrees, to retain computational feasibility. The approach taken by Breiman et al. (1984) is to modify the RSS to include a complexity penalty for trees that are too big, and find the subtree which minimizes such a loss function.

Formally, define the complexity of a subtree $T < T_{max}$ as $|\tilde{T}|$, the number of terminal nodes in T, and let $\alpha > 0$ be the complexity parameter. Then, the cost-complexity measure $RSS_\alpha(T)$ is:

$$RSS_\alpha(T) = RSS(T) + \alpha|\tilde{T}|$$

This measure introduces a trade-off between reducing the RSS via splitting a terminal node, and increasing the cost incurred through the complexity parameter α.

Let T_α be the smallest subtree minimizing $RSS_\alpha(T)$ for a fixed value of α, defined by the following conditions:

- $RSS_\alpha(T_\alpha) = min_{T \leq T_{max}} RSS_\alpha(T)$
- If $RSS_\alpha(T_\alpha) = RSS_\alpha(T)$ then $T_\alpha < T$. In other words, if another tree T achieves the minimum at the same α, then it is guaranteed to have more terminal nodes than T_α.

Breiman et al. (1984) has proved that for every non-negative α, there exists a smallest subtree $T_\alpha < T_{max}$, obtained through pruning, which minimizes RSS_α. Moreover, as we shall see shortly, the procedure to compute the minimizer returns a sequence of nested trees which are optimal for different values of the complexity parameter.

2.2.2.1 Minimal Cost-Complexity Pruning

Fix $\alpha = 0$ and let T_1 be the smallest subtree of T_{max} satisfying $RSS(T_1) = RSS(T_{max})$, which can be obtained as follows:

1. Identify a node η such that $RSS(\eta) = RSS(\eta_L) + RSS(\eta_R)$, and prune it.
2. Repeat the previous step recursively until no other node can be found.

In particular, we have that:

$$RSS_\alpha(\eta) > RSS_\alpha(T_\eta), \quad \forall\, \eta \in T_1 \tag{2.1}$$

where $RSS_\alpha(\eta) = RSS(\eta) + \alpha$. That is, pruning any other branch of T_1 will strictly increase the residual sum of squares. Hence, T_1 is the smallest subtree of T_{max} for $\alpha = 0$. It turns out that as α is increased past a certain value, say α^*, there exist at least one node for which Equation 2.1 no longer holds. Such node may not be unique and they are collectively referred to as weakest links. By pruning the weakest links from T_1, one can obtain the smallest subtree for α^*, which is advantageous because optimal results for smaller values of α can be re-used in the computation, rather than starting from T_{max}, saving computational time.
To identify the weakest link, solve for α in the inequality $RSS_\alpha(T_\eta) > RSS_\alpha(\eta)$, yielding:

$$\alpha < \frac{RSS(\eta) - RSS(T_\eta)}{|\tilde{T}_\eta| - 1}$$

The right hand side is the first value of α for which Eq. 2.1 does not hold, and T_1 is no longer optimal. Define a function $g_1(\eta),\ \eta \in T_1$ as:

$$g_1(\eta) = \begin{cases} \frac{RSS(\eta) - RSS(T_\eta)}{|\tilde{T}_\eta| - 1}, & \eta \notin \tilde{T}_1 \\ +\infty, & \eta \in \tilde{T}_1 \end{cases}$$

Then the weakest link $\bar{t}_1$ in T_1 is the node achieving the minimum of $g_1(\eta)$. Simply prune all weakest links from T_1 and denote the resulting subtree by T_2, and let $\alpha_2 = g_1(\bar{t}_1)$. Repeating the previous step, using T_2 instead of T_1 as starting tree, yields a new pair of complexity parameter and smallest subtree, (α_3, T_3). The process continues until only one terminal node is left. The result is a decreasing sequence of nested trees

$$T_1 > T_2 > ... > \eta_1$$

where η_1 is the root node of T_{max}, and a corresponding increasing sequence of α values

$$0 = \alpha_1 < \alpha_2 < ...$$

such that for $\alpha_k \leq \alpha < \alpha_{k+1}$, T_k is the smallest subtree of T_{max} minimizing $RSS_\alpha(T)$.

2.2.2.2 Best Pruned Subtree

The last step is to select the best pruned subtree from the sequence $T_1 > T_2 > ....$ For this, one can follow two natural options: test set evaluation and cross-validation.
The former is the simplest, however it requires more data since some observations must be taken out of the training data and placed in a dedicated test set, used to compare the performance of each tree in the sequence. Any suitable metric for the task at hand can be used for comparison, the simplest being the RSS itself. If data is scarce, k-fold cross-validation can be used to reduce the variance of the test set estimates, by averaging them across folds. The complete procedure is detailed in Algorithm 2.

2.3 Classification Trees

Classification trees are conceptually very similar to regression trees, since the growing and pruning phases follow the same logic, however they account for those cases where the dependent variable y is categorical in nature. Hence, some steps need to be modified to take this into account, namely adaptations in the loss function to be optimized, and the predictive rule $\hat{y}_{R_i}$ for each region $R_i,\ i = 1, ..., b$.

Let us tackle the predictive rule first. Recall that, for regression trees, $\hat{y}_{R_i} = \bar{y}_i$, so a natural alternative is to predict with the most commonly occurring class at each terminal region R_i. This is equivalent to choosing the label having highest relative frequency:

$$\hat{y}_{R_i} = \text{argmax}_l(\hat{p}_{il}), \quad \text{with} \quad \hat{p}_{il} = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbb{1}(y_{ij} = l),\ l = 1, ..., L$$

Algorithm 2: Pruning by Cross-Validation

1 Use recursive binary splitting to grow a large tree T_{max} on the training data, stopping when each terminal node has fewer than some minimum number of observations.
2 Apply cost complexity pruning to T_{max}, obtaining a sequence $(T_i, \alpha_i)_{i \geq 1}$
3 Divide the training observations into K random subsets.
4 **foreach** $k = 1, ..., K$ **do**
5 | Repeat steps 1 and 2 on all but the kth subset.
6 | Evaluate the target metric (e.g. MSE) on the left-out kth subset, for each tree in the sequence of step 5.
7 **end**
8 Average the results for each value of α, and pick α that minimizes the average metric.
9 Return the subtree from step 2 corresponding to the chosen α.

where $\hat{p}_{il}$ is the proportion of observations of class l in region R_i, $\mathbb{1}(\cdot)$ denotes the indicator function, and L the total number of classes for the qualitative variable y. Given such predictive rule, a natural way to replace the RSS is to use the mis-classification rate:

$$E_{R_i} = \frac{1}{n_i} \sum_{j=1}^{n_i} \mathbb{1}(y_{ij} \neq \hat{y}_{R_i}) = 1 - \max_l(\hat{p}_{il})$$

With these two changes it is possible to utilize the machinery detailed in Section 2.2. However, as discussed in detail in Section 4.1 of Breiman et al. (1984), direct minimization of mis-classification error is not desirable as there can be several splits at a given node that minimize it, along a more subtle problem: it has a tendency to produce trees that are more difficult to interpret. For these reasons, other measures of node impurity are preferred, such as Gini index

$$G_i = \sum_{l=1}^{L} \hat{p}_{il} \, (1 - \hat{p}_{il})$$

or cross-entropy

$$D_i = -\sum_{l=1}^{L} \hat{p}_{il} \, log(\hat{p}_{il})$$

They are both measures of the heterogeneity of a node because they take values close to zero if the region R_i is close to being pure, i.e. it contains mostly observations of the same class. In case of more than two classes, Gini index is more likely to split so that there is one relatively homogeneous child with relatively few cases, and the other relatively heterogeneous with relatively more cases. On the other hand, cross-entropy tends to

partition the data so that all the nodes for a given split are about equal in homogeneity and size (Berk, 2008, pp. 114-115).

2.4 CART Statistical Structure

In this section we make precise the statistical structure of CART models, needed to tackle their Bayesian extension in Chapter 3. As before, let

$$\begin{array}{lll} (\boldsymbol{x}, y)_{ij} & i = 1, ..., b & y \in \mathbb{R} \\ & j = 1, ..., n_i & \boldsymbol{x} \in \mathbb{R}^{\mathrm{p}} \end{array}$$

where indexes (ij) indicate the j^{th} observation in the i^{th} terminal node. Additionally, consider the parameter $\Theta = (\theta_1, ..., \theta_b)$ which associates the parameter value θ_i with the i^{th} terminal region. If $\boldsymbol{x}$ falls into region i, then we assume that $y|x$ has distribution $f(y|\theta_i)$, with f a parametric family indexed by θ_i. Practically, we can think of θ_i as the average value of y in region i, for regression trees, or in a two-class prediction problem, as the probability of y_{ij} of having class 1. Define

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

and X and X_i similarly. We further assume, as we have done implicitly in the previous sections, that y values within a terminal node are iid, and that across terminal nodes are independent. Then, the model distribution for the data can be written as:

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

For regression trees, we assume that $f(y_{ij}|\theta_i)$ is normally distributed:

$$y_{i1}, ..., y_{in_i} \,|\, \theta_i \quad \text{iid} \sim N(\mu_i, \sigma_i^2), \quad \theta_i = (\mu_i, \sigma_i^2) \tag{2.2}$$

For classification trees, where y belongs to one of K categories, say $C_1, ..., C_K$, we assume $f(y_{ij}|\theta_i)$ to be a generalized Bernoulli:

$$f(y_{i1}, ..., y_{in_i}) \,|\, \theta_i = \prod_{j=1}^{n_i} \prod_{k=1}^{K} p_{ik}^{\mathbb{1}(y_{ij} \in C_k)} \quad \theta_i = p_i = (p_{i1}, ..., p_{ik}) \tag{2.3}$$

where p_{ik} is the probability that observations in leaf i belong to class k; therefore $p_{ik} \geq 0$ and $\sum_k p_{ik} = 1$.

Chapter 3

Bayesian CART

This chapter focuses on the Bayesian extension of CART models, their statistical structure, stochastic search and computational challenges involved.

Probably the earliest work in this direction is the PhD thesis by Buntine (1992b), which introduces Bayesian ideas to the construction and selection of tree models; key ideas have been summarized in Buntine (1992a). Subsequently, two main papers on Bayesian tree modeling have been published, Denison, Mallick, et al. (1998) and Chipman, George, et al. (1998). Although they share the same underlying ideas and procedures, there are key differences. Denison, Mallick, et al. (1998) set the prior over trees to be a truncated Poisson for the number of terminal regions, putting equal weights on equal sized trees. Additionally, they use maximum likelihood estimates in their model rather than updating model parameters following the Bayesian approach. For a more through comparison of the two, and a generic treatment of Bayesian CART, see (Denison, Holmes, et al., 2002, ch. 6). The description of Bayesian trees given here follows closely that introduced by Chipman, George, et al. (1998). As presented in Section 3.5, these models fail to explore the posterior distribution in reasonable time due to multi-modality. A number of model adaptations have been considered to alleviate this, either improving prior specification or including more efficient proposals for the MH algorithm. Wu et al. (2007) provides an explicit specification of the prior distribution over trees, to control for both tree size and shape (known as 'pinball prior'), where Chipman, George, et al. (1998) do not directly control the number of leaves. Additionally, it introduces the "tree-restructure" move for MH, which makes large changes in the tree, leaving untouched the underlying partition of observations into terminal nodes, improving mixing. Similarly, Pratola et al. (2016) developed a novel tree rotational proposal that efficiently traverses different regions of the model space by local changes of the tree structure. It also proposed an improved version of the change move. Finally, Angelopoulos and Cussens (2005a) leverages *stochastic logic programming* (Muggleton et al., 1996) to design grow/prune moves that are able to prune off entire branches rather than just collapsing the parent of two terminal nodes, as in Chipman, George, et al. (1998).

Another line of research focused on improving prediction accuracy for Bayesian trees. Chipman and McCulloch (2000) introduced hierarchical priors that induce trees to shrink predictions from adjacent terminal nodes towards each other, incorporating local smoothness constraints for the model surface through the prior. Chipman, George, et al. (2002) extended Bayesian CART models to incorporate more sophisticated models at each terminal node, focusing particularly on linear regression. Finally, Chipman, George, et al. (2010) presented Bayesian Additive Regression Trees (BART), an ensemble learning model where each tree is constrained to be a weak learner by the (regularization) prior over trees. We do not explore BART and its improvements any further, instead we focus in great detail on its building blocks.

Recall that a CART model is fully identified by (Θ, T), where T encodes sample space partitioning information, and Θ the parameters of the parametric model assumed for y at each leaf. Hence, a Bayesian extension of the model requires the specification of a prior probability distribution $p(\Theta, T)$, usually defined following the relation:

$$p(\Theta, T) = p(\Theta|T)p(T)$$

An advantage of this strategy is that $p(T)$ doesn't depend on the specific parametric model chosen; moreover, it is usually easier to define $p(\Theta|T)$ conditional on the partition of the feature space.

This chapter is organized as follows: first we discuss the prior specification for the model, then we look at how to explore the posterior space, and conclude with a simulated example that evidences some of the computational challenges behind Bayesian CART models. A potential solution will be introduced later in Chapter 4.

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

Instead of defining a closed-form distribution over the space of trees, we specify the probability of observing a specific tree following a stochastic tree generating process, listed in Algorithm 3. Note that each tree produced following such procedure can be considered as a random draw from $p(T)$.

Algorithm 3: Tree Prior Generating Process

1. Set T to be the tree consisting of a single root node η.
2. Split the terminal node η with probability $p_{SPLIT}(\eta, T)$.
3. If η splits, assign it a splitting rule ρ sampled from the distribution $p_{RULE}(\rho|\eta, T)$ and create left and right children. Let T denote the newly created tree.
4. Apply steps 2 and 3 setting η equal to the children from previous step.
5. Stop when all terminal nodes have been exhausted.

3.1.1 $p_{SPLIT}(\eta, T)$

We can interpret $p_{SPLIT}(\eta, T)$ as the probability that node η splits generating two children nodes, defined as:

$$p_{SPLIT}(\eta, T) = \alpha(1 + d_\eta)^{-\beta} \tag{3.1}$$

Where $\alpha < 1$, $\beta \geq 0$ and d_η is the depth of node η. This component of $p(T)$ controls the shape and size of the tree, for instance higher values of β make deeper nodes less likely to split, while lower values of α reduce the probability of splitting at any depth. In this sense, $(1 + d_\eta)^{-\beta}$ can be interpreted as a sort of penalization for bigger trees, based on the value of the parameter β.
It is usually easier to decide values for α and β by considering the marginal distribution of some quantities of interest, such as tree depth or number of terminal regions. A simulation for the latter is reported in Figure 3.1 for different values of α and β.

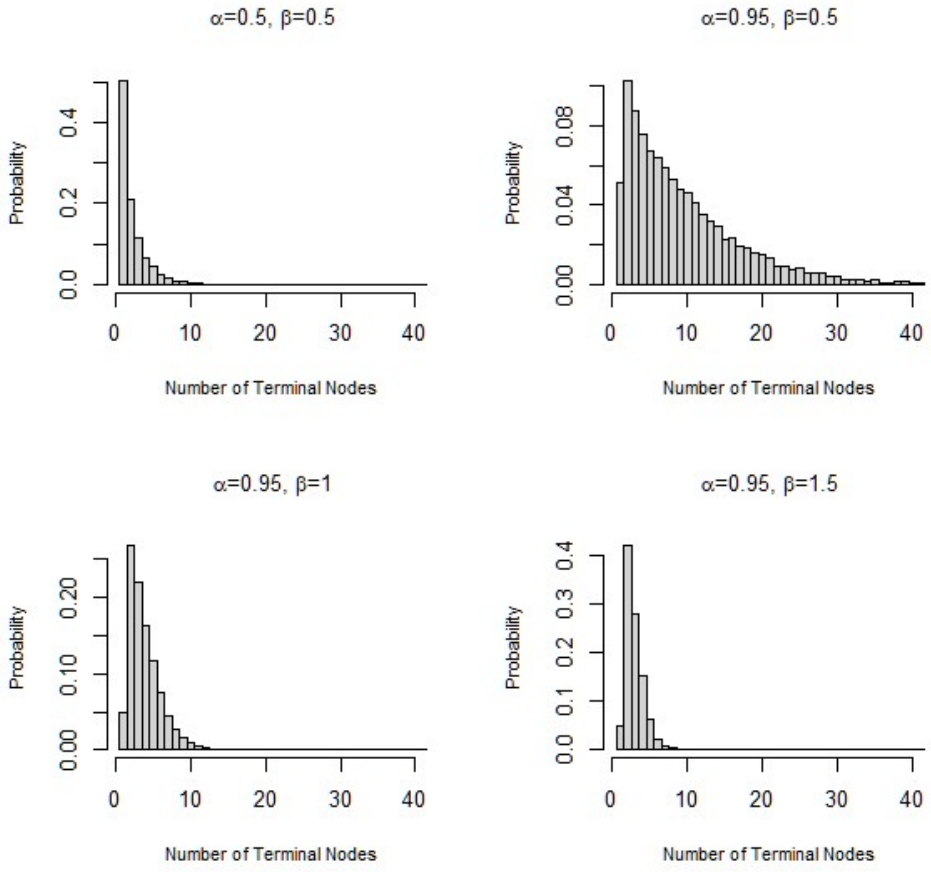


Figure 3.1: Marginal distribution of the number of terminal nodes for p_{SPLIT}, as defined in Eq. 3.1

3.1.2 $p_{RULE}(\rho|\eta, T)$

On the other hand, $p_{RULE}(\rho|\eta, T)$ is the probability of assigning the splitting rule ρ to η, provided η splits. Recall that a split is identified by a predictor x_l and a corresponding cutpoint s or subset A, based on the nature of x_l (ref. Section 2.2.1). Thus, we can describe $p_{RULE}(\rho|\eta, T)$ as a distribution on the set of available predictors x_l, and a distribution on the set of available splits s or A, conditional on x_l. Before continuing with the specification, a few remarks are in order:

- The denomination *available* indicates only those split variables and values that would not lead to empty terminal regions, producing an invalid tree.
- For a practical matter, we consider only specifications of p_{RULE} for which the set of split values is finite; this follows naturally from the fact that every dataset is necessarily finite.
- In general, p_{RULE} depends on X as splitting values and category subsets are usually chosen among the observed values of X for the corresponding predictor. Hence, $p(T)$ also depends on X.

We let $p_{RULE}(\rho|\eta, T)$ be the probability distribution obtained by sampling x_l at random among the available features at η, and consequently sampling at random s or A among the available split values or category subsets for x_l at η. This two-step process ensures uniformity across splits for the same predictor x_l, but not between different predictor splits. This is usually desirable as not to down-weight those having several observed values, unless there is prior information suggesting otherwise. The rationale for this choice is that, a priori, every variable is equally likely to be effective at a given node, and given such feature, every split is equally likely to be effective.
More refined (non-uniform) choices for p_{RULE} could be considered (Chipman, George, et al., 1998), however we will rely solely on the one just described.

3.2 Specification of the Parameter Prior $p(\Theta|T)$

The last component needed to obtain a prior over the space of trees is the prior parameter $p(\Theta|T)$, which relies on knowledge of the tree-induces sample space partitions to define a parametric model in each region. Among the different alternatives, it is useful to choose specifications for $p(\Theta|T)$ that allow analytical simplification when computing the posterior distribution. In particular, following Chipman, George, et al. (1998), we prefer prior forms for which Θ can be analytically marginalized to obtain the integrated likelihood:

$$P(Y|X, T) = \int P(Y|X, \Theta, T)P(\Theta|T)\mathrm{d}\Theta \tag{3.2}$$

This is particularly useful to explore the posterior distribution, as we will see in Section 3.3. We are going to consider two priors that make this marginalization possible, one for regression and the other for classification scenarios. In both cases, they are obtained by putting conjugate priors on leaf parameters and assuming conditional independence across terminal regions. Note that more flexible (hierarchical) priors can also be implemented, which model parameters dependence across nodes. These will not be discussed here, see Chipman and McCulloch (2000) for more details.

3.2.1 $p(\Theta|T)$ for Regression Trees

For regression trees under the mean-shift normal model (Eq. 2.2), we take the prior specification for $\Theta|T$ to be the standard conjugate form:

$$\begin{aligned} \mu_i \,|\, \sigma_i, T &\sim N(\bar{\mu}, \sigma_i^2/a) \\ \sigma_i^2 \,|\, T &\sim \text{IG}(\nu/2, \nu\lambda/2) \end{aligned}$$

with $i = 1, ..., b$ and the pairs $(\mu_1, \sigma_1), ..., (\mu_b, \sigma_b)$ identically distributed. After a bit of math (see Section 3.3.4.1), we get:

$$\begin{aligned} P(Y|X,T) = \prod_{i=1}^{b} \pi^{-n_i/2} (\nu\lambda)^{\nu/2} \frac{\sqrt{a}}{\sqrt{n_i + a}} \frac{\Gamma((n_i+\nu)/2)}{\Gamma(\nu/2)} \\ \left(\nu\lambda + S_i + \frac{n_i a}{n_i + a} (\bar{y}_i - \bar{\mu})^2 \right)^{(n_i+\nu)/2} \end{aligned} \tag{3.3}$$

Where π denotes the mathematical constant, $\Gamma(\cdot)$ is the Gamma function, $S_i = \sum_{j=1}^{n_i} (y_{ij} - \bar{y}_i)^2$ and $\bar{y}_i = \frac{1}{n_i} \sum_{j=1}^{n_i} y_{ij}$. Some guidelines of data-induced plausible values for priori parameters $(\nu, \lambda, \bar{\mu}, a)$ is given in Chipman, George, et al. (1998).

3.2.2 $p(\Theta|T)$ for Classification Trees

For regression trees under the generalized Bernoulli model (Eq. 2.3), we take the prior specification for $\Theta|T$ to be the standard Dirichlet distribution of dimension $K-1$ with parameter $\alpha = (\alpha_1, ..., \alpha_K)$, $\alpha_k > 0$:

$$p_1, ..., p_b \,|\, T \quad \text{iid} \sim \text{Dir}(p_i|\alpha) \propto p_{i1}^{\alpha_1 - 1} \cdots p_{iK}^{\alpha_K - 1}$$

After a bit of math (see Section 3.4), we get:

$$P(Y|X,T) = \left(\frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \right)^b \prod_{i=1}^{b} \frac{\prod_k \Gamma(n_{ik} + \alpha_k)}{\Gamma(n_i + \sum_k \alpha_k)} \tag{3.4}$$

where $n_{ik} = \sum_{j=1}^{n_i} I(y_{ij} \in C_k)$ and $n_i = \sum_{k=1}^{K} n_{ik}$. A natural choice for α is the vector $\alpha = (1, ..., 1)$ for which the Dirichlet prior becomes uniform.

3.3 Posterior Exploration

Let $\boldsymbol{\mu} = (\mu_1, ..., \mu_b)$ denote region means, and $\boldsymbol{\sigma^2} = (\sigma_1^2, ..., \sigma_b^2)$ be region variance, so that $\Theta = (\boldsymbol{\mu}, \boldsymbol{\sigma})$. Then, the posterior distribution over (Θ, T) becomes:

$$\begin{aligned} P(\Theta, T|X, Y) &\propto P(\Theta|T, X, Y)P(T|X, Y) \\ &\propto P(\boldsymbol{\mu}|X, Y, T, \boldsymbol{\sigma})P(\boldsymbol{\sigma}|X, Y, T) \\ &\quad P(Y|X, T)P(T|X) \end{aligned}$$

Since $P(T|X)$ is described using a stochastic generation process, it is known only up to a constant. Therefore, even if $P(\Theta|T, X, Y)$ can be computed analytically, $P(T|X, Y)$ cannot. Lacking a closed-form expression to analyze (for example, to compute moments or other quantities of interest for inference), we must resort to sampling in order to reconstruct the probability distribution. For that we follow a sequential approach, exploiting the statistical structure and dependencies of the model:

1. Sample T from $P(T|X, Y)$.
2. Given the information on feature space partitioning contained in T, sample $\boldsymbol{\sigma}$ from $P(\boldsymbol{\sigma}|X, Y, T, \boldsymbol{\mu})$.
3. Given T and $\boldsymbol{\sigma}$, sample $\boldsymbol{\mu}$ from $P(\boldsymbol{\mu}|X, Y, T, \boldsymbol{\sigma})$.

The formulas detailed in steps 2 and 3 can be computed analytically, and under the conjugate priors selected in Section 3.1, routine sampling methods can be applied, so that no additional effort is required. On the other hand, step 1 involves evaluation of $P(T|X)$, that, as mentioned above, is known up to a constant. Hence, we rely on the same methodology used by Chipman, George, et al. (1998): Metropolis-Hastings (MH) algorithm (Metropolis et al., 1953, Hastings, 1970). We use this algorithm to simulate a Markov chain sequence of trees

$$T^0, T^1, T^2, ...$$

that are converging in distribution to the posterior $p(T|Y, X)$ under mild conditions (Tierney, 1994). The complete sampling procedure is detailed in Algorithm 4 below.

While the results derived so far are valid regardless of the nature of y, whether qualitative or quantitative, for the remainder of this section we assume that $y \in \mathbb{R}$. Extensions to the classification case will be dealt with later on. We now carefully analyze each component of Algorithm 4.

Algorithm 4: Posterior Sampling for Bayesian CART Models
Regression Trees under Mean-Variance Shift Model

Data: Target variable y (length n), feature matrix X (n rows, p columns)
Result: Posterior list of trees

Initialization
Hyper-parameter values of α, β, a, $\bar{\mu}$, ν, λ
Optional parameters: μ_0, σ_0
Move probabilities: $P(move)$, $move \in \{grow, prune, change, swap\}$
Number of iterations: N
Thinning amount: *thinning*
If not given, set $\sigma_0^2 \sim \text{IG}(\nu/2, \nu\lambda/2)$
If not given, set $\mu_0 \sim N(\bar{\mu}, \sigma_0^2/a)$
Set initial tree T to stump, with parameters μ_0, σ_0

for *loop* $\leftarrow 1$ **to** N **do**
  $move \leftarrow$ sample a move from $\{grow, prune, change, swap\}$
  $T^* \leftarrow$ generate candidate tree from T, based on *move*
  if T^* *is valid* **then** `// no empty terminal region`
    $\alpha(T, T^*) \leftarrow$ compute MH acceptance probability, Section 3.3.4
    Generate $u \sim U(0, 1)$
    if $u \leq \alpha(T, T^*)$ **then**
      $T \leftarrow T^*$
    end
  end
  Get prediction $\hat{y}$ from T
  Simulate $\boldsymbol{\sigma}$ values using Equation 3.5
  Simulate $\boldsymbol{\mu}$ values using Equation 3.6
  if *loop % thinning == 0* **then**
    Store tree data.
  end
end

3.3.1 Full Conditional for $\boldsymbol{\sigma}$

In this section we derive a closed-form expression for $P(\boldsymbol{\sigma}|X, Y, T, \boldsymbol{\mu})$. Using Bayes theorem, we obtain:

$$P(\boldsymbol{\sigma}|X, Y, T, \boldsymbol{\mu}) \propto P(Y|X, T, \boldsymbol{\mu}, \boldsymbol{\sigma})P(\boldsymbol{\sigma}|T)$$

Let $\tau_i = \sigma_i^2$

$$\begin{aligned}
&\propto \prod_{i=1}^{b}\prod_{j=1}^{n_i} \tau_i^{-1/2}\exp\left[-\frac{(y_{ij}-\mu_i)^2}{2\tau_i}\right]\prod_{i=1}^{b}\tau_i^{(-\nu/2-1)}\exp\left[-\frac{\nu\lambda}{2\tau_i}\right] \\
&= \prod_{i=1}^{b}\tau_i^{-(n_i+v)/2-1}\exp\left[-\frac{\sum_{j=1}^{n_i}(y_{ij}-\mu_i)^2+\nu\lambda}{2\tau_i}\right]
\end{aligned}$$

By inspection, it can be observed that

$$\sigma_i^2|X, Y, T \quad \text{iid} \sim \text{IG}\left(\frac{n_i+\nu}{2},\ \frac{1}{2}\left(\sum_{j=1}^{n_i}(y_{ij}-\mu_i)^2+\nu\lambda\right)\right) \tag{3.5}$$

using the parametrization of the Inverse-Gamma distribution, having density

$$f(x) = \frac{\beta^\alpha}{\Gamma(\alpha)}x^{-\alpha-1}e^{-\beta/x}$$

3.3.2 Full Conditional for $\boldsymbol{\mu}$

Similarly to before, we apply Bayes theorem to obtain:

$$P(\boldsymbol{\mu}|X, Y, T, \boldsymbol{\sigma}) \propto P(Y|X, T, \boldsymbol{\mu}, \boldsymbol{\sigma})P(\boldsymbol{\mu}|T, \boldsymbol{\sigma})$$

Let $\tau_i = \sigma_i^2$

$$\propto \prod_{i=1}^{b}\prod_{j=1}^{n_i} \tau_i^{-1/2}\exp\left[-\frac{(y_{ij}-\mu_i)^2}{2\tau_i}\right]\prod_{i=1}^{b}\sqrt{a}\,\tau_i^{-1/2}\exp\left[-\frac{a(\mu_i-\bar{\mu})^2}{2\tau_i}\right]$$

Following computation similar to those done for $P(Y|X, T)$ in Section 3.3.4.1, we get

$$\propto \prod_{i=1}^{b}\exp\left[-\frac{n_i+a}{2\tau}\left(\mu_i-\frac{n_i\bar{y}_i+a\bar{\mu}}{n_i+a}\right)^2\right]$$

By inspection, it can be observed that

$$\mu_i|X, Y, T, \boldsymbol{\sigma} \quad \text{iid} \sim N\left(\frac{n_i\bar{y}_i+a\bar{\mu}}{n_i+a},\ \frac{\sigma_i^2}{n_i+a}\right) \tag{3.6}$$

3.3.3 Metropolis-Hastings Specification

Compared to the other steps, sampling from $P(T|Y,X)$ is by far the most computationally difficult part, which we will attempt to improve in Chapter 4. However, before getting there, we need to look at the basic mechanism and implementation details of the MH algorithm. This is used to generate a Markov chain of trees starting from an initial tree T^0 and iteratively simulating a transition from T^i to T^{i+1} as follows:

1. Sample a candidate value T^* from the transition kernel $q(T^i, T^*)$.

2. Set $T^{i+1} = T^*$ with probability

$$\alpha(T^i, T^*) = \min\left\{\frac{q(T^*, T^i)}{q(T^i, T^*)}\frac{P(Y|X,T^*)P(T^*)}{P(Y|X,T^i)P(T^i)}, 1\right\} \tag{3.7}$$

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

We refer to Eq. 3.7 as the acceptance probability, discussion about its computation is postponed to Section 3.3.4. Notice how the norming constant needed to compute $p(T|Y,X)$ simplifies in the acceptance ratio, which makes MH particularly suited for these kind of situations.

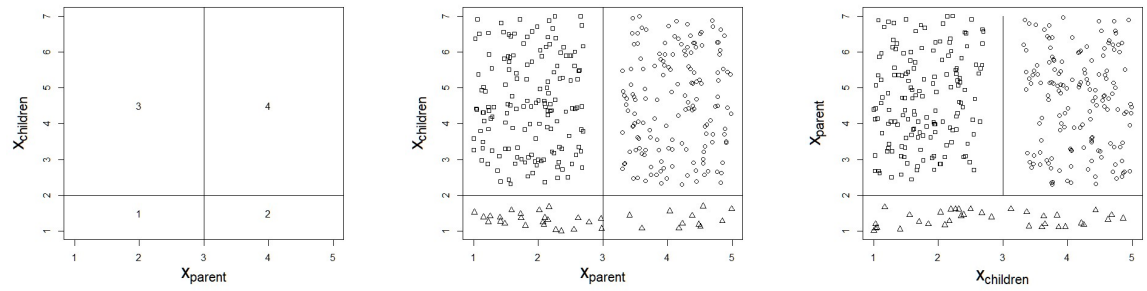
The transition kernel $q(T^i, T^*)$ is defined so that it chooses at random among four possible moves:

- *grow*: randomly pick a terminal node, split it into two children nodes and assign a splitting rule each according to p_{RULE} (ref. Section 3.1.2)

- *prune*: randomly pick a parent of two terminal nodes and turn it into a terminal node by removing its children.

- *change*: randomly pick an internal node and replace its splitting rule with another sampled from p_{RULE}.

- *swap*: randomly pick a parent-child pair of internal nodes and swap their splitting rules. If the two children have the same splitting rule, then swap that of the parent with those of both children. The rationale is that if after the parent split (x_p, s_p) both children come up with the same pair (x_l, s), then the ordering of the splits is wrong and model complexity can be reduced by moving the child's rule up one level, and pruning off one of the children at a later step. See Figure 3.2 for an example.



(a) (b) (c)

Figure 3.2: Illustration of *swap* move when both children have same splitting rule, as shown in (a). (b) displays the same split with potential datapoints overlaid, note how regions 1 and 2 contain observations of the same class. In (c) the rules have been swapped and one of the children pruned off, yielding same accuracy with a smaller tree.

The proposal distribution defined above yields a reversible Markov chain since every evolution from T^i to T^* has a counterpart that can move the chain back to T^i. In fact, note that *grow* and *prune* are each other's counterpart, while *change* and *swap* are their own counterpart. Moreover, this specification allows significant simplification in computing the acceptance probability, which is tackled in the following section.

3.3.4 Metropolis-Hastings Acceptance Probability

To ease notation, let T be the current tree, and T^* be the proposed tree after move $move \in \{grow, prune, change, swap\}$, respectively. Denote by $q(T, T^*)$ the probability distribution of generating a candidate T^* from the current tree T. The acceptance probability for the MH algorithm is computed as

$$\alpha(T, T^*) = \min\left\{\frac{q(T^*, T)}{q(T, T^*)}\frac{P(Y|X, T^*)P(T^*)}{P(Y|X, T)P(T)}, 1\right\}$$

Computations for the acceptance probability depend on the move. In particular, it is useful to divide it into smaller chunks: the likelihood ratio, the transition ratio $q(T^*, T)/q(T, T^*)$, and the tree structure ratio $P(T^*)/P(T)$. Note that computation for the latter two changes based on the type of move being carried out, as we will see in the coming sections. As for the likelihood ratio, it is always computed in full regardless of the move type, in log terms.

3.3.4.1 Integrated Likelihood Computation

We focus first on the task of computing $P(Y|X,T)$; this can be obtained in closed form as follows:

$$P(Y|X,T) = \int P(Y|X,\Theta,T)P(\Theta|T)\mathrm{d}\Theta \tag{3.8}$$

$$\begin{aligned} &= \int \prod_{i=1}^{b}\prod_{j=1}^{n_i}(2\pi\sigma_i^2)^{-1/2}\exp\left[-\frac{(y_{ij}-\mu_i)^2}{2\sigma_i^2}\right] \\ &\quad \prod_{i=1}^{b}\sqrt{a}(2\pi\sigma_i^2)^{-1/2}\exp\left[-\frac{a(\mu_i-\bar{\mu})^2}{2\sigma_i^2}\right] \\ &\quad \prod_{i=1}^{b}\frac{(\nu\lambda)^{\nu/2}}{\Gamma(\nu/2)}\sigma_i^{2(-\nu/2-1)}\exp\left[-\frac{\nu\lambda}{2\sigma_i^2}\right]\mathrm{d}\boldsymbol{\mu}\,\mathrm{d}\boldsymbol{\sigma} \end{aligned} \tag{3.9}$$

Let $\tau_i = \sigma_i^2$ and $\boldsymbol{\tau} = (\tau_1, ..., \tau_b)$

$$\begin{aligned} &= \int \prod_{i=1}^{b}\sqrt{a}\,\frac{(\nu\lambda)^{\nu/2}}{\Gamma(\nu/2)}\,(2\pi)^{-(n_i+1)/2}\,\tau_i^{-(n_i+3+\nu)/2}\,e^{(\nu\lambda)/(2\tau_i)} \\ &\quad \left(\int \prod_{i=1}^{b}\exp\left[\frac{1}{2\tau_i}\left(\sum_{j=1}^{n_i}(y_{ij}-\mu_i)^2 + a(\mu_i-\bar{\mu})^2\right)\right]\mathrm{d}\boldsymbol{\mu}\right)\mathrm{d}\boldsymbol{\tau} \end{aligned} \tag{3.10}$$

Let $S_i = \sum_{j=1}^{n_i}(y_{ij}-\bar{y_i})^2$ and $\bar{y_i} = \frac{1}{n_i}\sum_{j=1}^{n_i} y_{ij}$. Using the following fact $\sum_{j=1}^{n_i}(y_{ij}-\mu_i)^2 = S_i + n_i(\mu_i-\bar{y_i})^2$ and recognizing that the terms inside the second integral constitute a normal distribution (up to a normalizing constant; see steps below), we get

$$\begin{aligned} &= \int \prod_{i=1}^{b}\sqrt{a}\,\frac{(\nu\lambda)^{\nu/2}}{\Gamma(\nu/2)}\,(2\pi)^{-(n_i+1)/2}\,\tau_i^{-(n_i+3+\nu)/2}\,e^{(\nu\lambda)/(2\tau_i)} \\ &\quad \left(\prod_{i=1}^{b}\exp\left[\frac{1}{2\tau_i}\left(S_i + \frac{n_i a}{n_i+a}(\bar{y_i}-\bar{\mu})^2\right)\right]\left(\frac{n_i+a}{\tau_i}\right)^{-1/2}\right)\mathrm{d}\boldsymbol{\tau} \end{aligned} \tag{3.11}$$

$$\begin{aligned} &= \int \prod_{i=1}^{b}\tau_i^{-\frac{n_i+\nu}{2}-1}\exp\left[\frac{1}{2\tau_i}\left(\nu\lambda + S_i + \frac{n_i a}{n_i+a}(\bar{y_i}-\bar{\mu})^2\right)\right] \\ &\quad \sqrt{a}\,\frac{(\nu\lambda)^{\nu/2}}{\Gamma(\nu/2)}\,(2\pi)^{-n_i/2}(n_i+a)^{-1/2}\,\mathrm{d}\boldsymbol{\tau} \end{aligned} \tag{3.12}$$

Similarly, the integral can be solved recognizing an Inverse-Gamma distribution, up to a constant of proportionality

$$= \prod_{i=1}^{b} \pi^{-n_i/2} (\nu\lambda)^{\nu/2} \frac{\sqrt{a}}{\sqrt{n_i + a}} \frac{\Gamma((n_i + \nu)/2)}{\Gamma(\nu/2)} \left(\nu\lambda + S_i + \frac{n_i a}{n_i + a} (\bar{y}_i - \bar{\mu})^2 \right)^{(n_i+\nu)/2} \tag{3.13}$$

To obtain Eq. 3.11, we relied on the following decomposition, displayed here with simplified notation

$$\begin{aligned} \sum_{j=1}^{n} (y_j - \mu)^2 + a(\mu - \bar{\mu})^2 &= S + n(\mu - \bar{y})^2 + a\mu^2 + a\bar{\mu}^2 - 2a\mu\bar{\mu} \\ &= (S + n\bar{y}^2 + a\bar{\mu}^2) + ((a+n)\mu^2 - 2n\mu\bar{y} - 2a\mu\bar{\mu}) \\ &= (S + n\bar{y}^2 + a\bar{\mu}^2) + (n + a) \\ &\quad \left[\left(\mu - \frac{n\bar{y} + a\bar{\mu}}{n + a} \right)^2 - \left(\frac{n\bar{y} + a\bar{\mu}}{n + a} \right)^2 \right] \end{aligned}$$

Note the separation between terms containing μ, which is what we seek to isolate to solve the integral, and constants, which can go outside the second integral, in Eq. 3.10. To get Eq. 3.11, just replace in the formula and recognize a normal distribution.

3.3.4.2 Grow

The computation for *grow* and *prune* follow loosely those of Kapelner and Bleich (2013), albeit with different notation. Those for *change* differ in that Kapelner and Bleich (2013) uses a simplified notion of this move type to reduce bookkeeping, while they don't implement *swap* altogether.
Before jumping to the computation we need to introduce some notation to ease formula burden later on:

- Let η denote the target node for the move, namely the node to split for *grow*, the one to prune for *prune*, the node at which change splitting rule for *change*, and the parent node (in the parent-child pair) for *swap*. In the latter, we additionally use $c1$ and $c2$ to indicate the two children.

- Let $*$ denote an attribute of the proposed tree T^*. For example, η^* during *swap* refers to the parent node in the proposed tree T^*.

- Let $\#v_r$ denote the number of available variables to split on at a generic node r, in the original tree. The corresponding value in the proposal tree would be $\#v_r^*$.

- Let $\#s|w_r$ be the number of available splits for variable w at node r, in the original tree. Similarly, $\#s|w_r^*$ corresponds to the proposed tree T^*.

Transition Ratio
Transitioning from the current tree T to T^* involves splitting a terminal node η into two children nodes, by assigning it a splitting rule. Assume w is the corresponding splitting variable, selected at random among the available ones. The split value for w is also chosen at random, among the available ones, as specified in the definition of p_{RULE} (Sec. 3.1.2).

$$\begin{aligned} P(T \to T^*) =& P(grow)P(\text{selecting } \eta \text{ to grow from})\times \\ & P(\text{selecting } w \text{ to split on})\times \\ & P(\text{selecting a split value for } w) \\ =& P(grow)\ \frac{1}{b}\ \frac{1}{\#v_\eta}\ \frac{1}{\#s|w_\eta} \end{aligned}$$

We believe the notation introduced above is useful in easing formulas, however can be a bit obscure at the beginning. To avoid interpretation errors, the above formula reads: *the probability of growing T times one over the number of terminal regions for T, times one over the number of available variables to split on at η in the original tree T, times one over the number of available values for splitting along feature w, at η, in T.*

Transitioning from T^* to T requires pruning η:

$$P(T^* \to T) = P(prune)P(\text{selecting } \eta \text{ to prune from}) = P(prune)\frac{1}{cand^*}$$

where $cand^*$ denotes the number of second generation internal nodes (those with two terminal children nodes), in T^*. All together, the transition ratio is:

$$\frac{P(T^* \to T)}{P(T \to T^*)} = \frac{P(prune)}{P(grow)}\ \frac{b\ \#v_\eta\ \#s|w_\eta}{cand^*}$$

Note: it is implicitly assumed that there is at least one available split at η, which is necessary for the resulting tree to be a valid one, otherwise it would lead to an empty terminal region. If such split were not available, the transition probability would be taken to be zero, as the tree is automatically rejected.

Tree Structure Ratio
Recall that the tree structure can be written, in terms of p_{SPLIT} and p_{RULE}, as:

$$P(T) = \prod_{\eta \in \tilde{T}} (1 - p_{SPLIT}(\eta)) \prod_{\eta \in T - \tilde{T}} p_{SPLIT}(\eta) \prod_{\eta \in T - \tilde{T}} p_{RULE}(\eta)$$

where $\tilde{T}$ denotes the set of terminal nodes, and $T - \tilde{T}$ that of internal nodes. Since *grow* leaves unchanged the part of the tree above η, T and T^* only differ at η and at the two children nodes η_1 and η_2. The ratio is:

$$\begin{aligned}
\frac{P(T^*)}{P(T)} &= \frac{p_{SPLIT}(\eta)\, p_{RULE}(\eta)\, (1 - p_{SPLIT}(\eta_1))\, (1 - p_{SPLIT}(\eta_2))}{(1 - p_{SPLIT}(\eta))} \\
&= \frac{\frac{\alpha}{(1+d_\eta)^\beta}\ \frac{1}{\#v_\eta}\ \frac{1}{\#s|w_\eta}\ \left(1 - \frac{\alpha}{(2+d_\eta)^\beta}\right)^2}{\left(1 - \frac{\alpha}{(1+d_\eta)^\beta}\right)} \\
&= \alpha \left(1 - \frac{\alpha}{(2+d_\eta)^\beta}\right)^2 \left[\left((1+d_\eta)^\beta - \alpha\right)\ \#v_\eta\, \#s|w_\eta\right]^{-1}
\end{aligned}$$

using the fact that the depth of the children nodes is one more that of the parent ($d_{\eta_1} = d_{\eta_2} = d_\eta + 1$).

Transition + Tree Structure Ratio
Combining the two ratios together usually provides fruitful, as some simplification can be done. For *grow* it becomes:

$$\frac{P(T^* \to T)P(T^*)}{P(T \to T^*)P(T)} = \frac{P(prune)}{P(grow)} \frac{b\,\alpha}{cand*} \left(1 - \frac{\alpha}{(2+d_\eta)^\beta}\right)^2 \left((1+d_\eta)^\beta - \alpha\right)^{-1}$$

3.3.4.3 Prune

Recall that *prune* and *grow* are one the reverse of the other; computation for the ratios is the inverse of those for *grow*, therefore we report only the final result.

Transition Ratio

$$\frac{P(T^* \to T)}{P(T \to T^*)} = \frac{P(grow)}{P(prune)}\ \frac{cand}{b^*\ \#v_\eta^*\ \#s|w_\eta^*}$$

Tree Structure Ratio

$$\frac{P(T^*)}{P(T)} = \frac{1}{\alpha} \left(1 - \frac{\alpha}{(2+d_\eta)^\beta}\right)^{-2} \left[\left((1+d_\eta)^\beta - \alpha\right)\ \#v_\eta^*\, \#s|w_\eta^*\right]$$

Transition + Tree Structure Ratio

$$\frac{P(T^* \to T)P(T^*)}{P(T \to T^*)P(T)} = \frac{P(grow)}{P(prune)}\ \frac{cand}{\alpha\, b^*} \left(1 - \frac{\alpha}{(2+d_\eta)^\beta}\right)^{-2} \left((1+d_\eta)^\beta - \alpha\right)$$

3.3.4.4 Change

The *change* move picks an internal node η uniformly and replaces its splitting rule according to p_{RULE}. Note that *change* is its own counterpart. Assume that node η splits along feature w in T, and along w^* in T^*.

Transition Ratio
The probability of transitioning from T to T^* is very similar to *grow*:

$$\begin{aligned} P(T \to T^*) = & P(change)P(\text{selecting } \eta \text{ to change}) \times \\ & P(\text{selecting } w^* \text{ to split on}) \times \\ & P(\text{selecting a split value for } w^*) \end{aligned}$$

However, the *change* move does not affect the graphical structure of the tree, intended as the number and relationship between nodes, only the induced partition of the feature space. For this reason, the transition ratio simplifies somewhat:

$$\frac{P(T^* \to T)}{P(T \to T^*)} = \frac{\#v_\eta \, \#s|w_\eta}{\#v_\eta^* \, \#s|w_\eta^*} = \frac{\#s|w_\eta}{\#s|w_\eta^*}$$

Where we used the fact that the number of available variables at η is the same both in T ($\#v_\eta$) and in T^* ($\#v_\eta^*$), since the structure of the tree is unchanged, therefore also the part of the tree above η is unchanged, so that η and η^* look at the same data subset.

Tree Structure Ratio
The original and proposed tree differ only for the splitting rule at η, considering the remark above we obtain:

$$\frac{P(T^*)}{P(T)} = \frac{\#s|w_\eta^*}{\#s|w_\eta}$$

Transition + Tree Structure Ratio
It is easy to see that the two ratios simplify, so that overall for *change* move type, the acceptance probability is determined by the likelihood ratio.

$$\frac{P(T^* \to T)P(T^*)}{P(T \to T^*)P(T)} = 1$$

3.3.4.5 Swap

The *swap* move is slightly more complex to tackle, due to the possibility that both children's splitting rule will be swapped with that of their parent, a conditional event which needs to be taken into account when computing transition probabilities. Here we consider only this latter case, the simpler one where the parent rule is swapped with that of one child is similar and yields the same final result.
Formally, let $(\eta, c1)$ be a parent-child pair of internal nodes, $c2$ be the other child of η, and w_η, w_{c1}, w_{c2} be the splitting variable in T at node η, $c1$ and $c2$ respectively. Let also s_η, s_{c1}, s_{c2} be the split values for w_η, w_{c1}, w_{c2} at η, $c1$ and $c2$ respectively. Note that, in general, w_{c2} (and therefore s_{c2}) could be null as $c2$ is not guaranteed to be internal, hence we assume that it is internal and that the splitting rule of $c1$ and $c2$ are equal, namely $w_{c1} = w_{c2}$ and $s_{c1} = s_{c2}$. Denote with $\eta*$ the node in the same position of η in T^* (the *swap* move, like *change*, does not affect the graphical structure of the tree), and with $c1^*$ and $c2^*$ its two children. By definition of *swap*, (w_{c1}, s_{c1}) will be the splitting rule of η^*, and (w_η, s_η) that of $c1^*$ and $c2^*$.

Transition Ratio
Transitioning from T to T^*:

$$\begin{aligned} P(T \to T^*) =& P(swap)P(\text{selecting a triplet } (\eta, c1, c2) \text{ to swap})\times \\ & P(\text{selecting split } (w_{c1}, s_{c1}) \text{ at } \eta)\times \\ & P(\text{selecting split } (w_\eta, s_\eta) \text{ at } c1)\times \\ & P(\text{selecting split } (w_\eta, s_\eta) \text{ at } c2) \\ =& P(S)P(\text{selecting a triplet } (\eta, c1, c2) \text{ to swap})\times \\ & \frac{1}{\#v^*_\eta \, \#s|w^*_{c1}} \; \frac{1}{\#v^*_{c1} \, \#s|w^*_\eta} \; \frac{1}{\#v^*_{c2} \, \#s|w^*_\eta} \end{aligned}$$

Transitioning from T^* to T:

$$\begin{aligned} P(T^* \to T) =& P(swap)P(\text{selecting a triplet } (\eta^*, c1^*, c2^*) \text{ to swap})\times \\ & P(\text{selecting split } (w_\eta, s_\eta) \text{ at } \eta^*)\times \\ & P(\text{selecting split } (w_{c1}, s_{c1}) \text{ at } c1^*)\times \\ & P(\text{selecting split } (w_{c1}, s_{c1}) \text{ at } c2^*) \\ =& P(S)P(\text{selecting a triplet } (\eta^*, c1^*, c2^*) \text{ to swap})\times \\ & \frac{1}{\#v_\eta \, \#s|w_\eta} \; \frac{1}{\#v_{c1} \, \#s|w_{c1}} \; \frac{1}{\#v_{c2} \, \#s|w_{c2}} \end{aligned}$$

Combining the two yields:

$$\frac{P(T^* \to T)}{P(T \to T^*)} = \frac{\#s|w^*_{c1} \; \#v^*_{c1} \, \#s|w^*_\eta \; \#v^*_{c2} \, \#s|w^*_\eta}{\#s|w_\eta \; \#v_{c1} \, \#s|w_{c1} \; \#v_{c2} \, \#s|w_{c2}}$$

A number of remarks are in order:

- P(selecting a triplet to swap) is the same in both trees T and T^* since the structure does not change.
- The number of available variables to split on at η^* ($\#v_{\eta^*}$) and at η ($\#v_\eta$) is equal for the same reason as above, since the two nodes share the same data subset.
- The number of available variables to split on at $c1$ ($\#v^*_{c1}$) and at $c1^*$ ($\#v_{c1}$) is potentially different since the parent split is different and they may be looking at a different data subset. Same for $c2$ and $c2^*$.
- The number of available split values at η ($\#s|w^*_{c1}$) should be computed on the proposal tree T^*, as suggested by the notation, since it must take into account the new splitting variable assigned to η, taken from the children. Similar reasoning for the splits at $c1$ and $c2$.
- The number of available split values at η ($\#s|w^*_{c1}$) is potentially different from those at η^* ($\#s|w_\eta$) since the two split variables are different (as suggested by the notation employed).

Tree Structure Ratio
The tree structure ratio for *swap* is the inverse of its transition ratio, as it was for *change*, and it follows from the same reason: the tree structure is identical between T and T^*.

$$\frac{P(T^*)}{P(T)} = \frac{\#s|w_\eta \; \#v_{c1} \; \#s|w_{c1} \; \#v_{c2} \; \#s|w_{c2}}{\#s|w^*_{c1} \; \#v^*_{c1} \; \#s|w^*_\eta \; \#v^*_{c2} \; \#s|w^*_\eta}$$

Transition + Tree Structure Ratio
As for *change*, the two ratios simplify, so that the acceptance probability is determined by the likelihood ratio.

$$\frac{P(T^* \to T)P(T^*)}{P(T \to T^*)P(T)} = 1$$

3.4 Extension to Classification Trees

Extension to the classification case is straightforward. Recall the statistical model as defined in Eq. 2.3 and the parameter prior treated in Section 3.2.2:

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

$$p_1, ..., p_b \,|\, T \quad \text{iid} \sim \text{Dir}(p_i|\alpha)$$

Where K is the number of classes, each denoted by $C_1, ..., C_K$, and p_{ik} is probability that observations in region i belong to class k. Let

$$B(\alpha) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{K} \alpha_k)} \qquad n_{ik} = \sum_{j=1}^{n_i} I(y_{ij} \in C_k)$$

$$n_i = \sum_{k=1}^{K} n_{ik} \qquad \boldsymbol{p} = (p_1, ..., p_b)$$

Then, integrated likelihood computation follows the same logic as in the regression case:

$$P(Y|X,T) = \int P(y|X,T,\boldsymbol{p})\, P(\boldsymbol{p}|T)\, \mathrm{d}\boldsymbol{p} \tag{3.14}$$

$$= \int \prod_{i=1}^{b} \prod_{k=1}^{K} \prod_{j=1}^{n_i} p_{ik}^{I(y_{ij} \in C_k)} \prod_{i=1}^{b} \frac{1}{B(\alpha)} \prod_{k=1}^{K} p_{ik}^{\alpha_k - 1}\, \mathrm{d}\boldsymbol{p} \tag{3.15}$$

$$= \int \prod_{i=1}^{b} \frac{1}{B(\alpha)} \prod_{k=1}^{K} p_{ik}^{n_{ik} + \alpha_k - 1}\, \mathrm{d}\boldsymbol{p} \tag{3.16}$$

The integral can be solved as for the regression case, recognizing by inspection an (updated) Dirichlet distribution, up to a normalizing constant.

$$= \left(\frac{\Gamma(\sum_k \alpha_k)}{\prod_k \Gamma(\alpha_k)} \right)^b \prod_{i=1}^{b} \frac{\prod_k \Gamma(n_{ik} + \alpha_k)}{\Gamma(n_i + \sum_k \alpha_k)} \tag{3.17}$$

The final ingredient needed is the full conditional for $\boldsymbol{p}$:

$$\begin{aligned} P(\boldsymbol{p}|T,X,Y) &\propto P(Y|X,T,\boldsymbol{p})\, P(\boldsymbol{p}|T) \\ &= \prod_{i=1}^{b} \prod_{k=1}^{K} \prod_{j=1}^{n_i} p_{ik}^{I(y_{ij} \in C_k)} \prod_{i=1}^{b} \frac{1}{B(\alpha)} \prod_{k=1}^{K} p_{ik}^{\alpha_k - 1} \\ &\propto \prod_{i=1}^{b} \prod_{k=1}^{K} p_{ik}^{n_{ik} + \alpha_k - 1} \end{aligned}$$

By inspection, it can be observed that

$$p_i|X, Y, T \quad \text{iid} \sim \text{Dir}(\alpha_1 + n_{i1}, ..., \alpha_k + n_{ik})$$

Overall, the sampling procedure in the classification case is detailed in Algorithm 5. In computing the acceptance probability, replace the integrated likelihood with Eq. 3.4.

Algorithm 5: Posterior Sampling for Bayesian CART Models
Classification Trees

Data: Target variable y (length n), feature matrix X (n rows, p columns)
Result: Posterior list of trees

Initialization
Hyper-parameter values of α, β, $(\alpha_1, ..., \alpha_K)$, $\bar{\mu}$, ν, λ
Optional parameters: p_0
Move probabilities: $P(move)$, $move \in \{grow, prune, change, swap\}$
Number of iterations: N
Thinning amount: *thinning*
If not given, set $p_0 \sim \text{Dir}(\alpha_1, ..., \alpha_K)$
Set initial tree T to stump, with parameters p_0

for *loop* $\leftarrow 1$ **to** N **do**
    $move \leftarrow$ sample a move from $\{grow, prune, change, swap\}$
    $T^* \leftarrow$ generate candidate tree from T, based on *move*
    if T^* *is valid* **then** `// no empty terminal region`
        $\alpha(T, T^*) \leftarrow$ compute MH acceptance probability, Section 3.3.4
        Generate $u \sim U(0, 1)$
        if $u \leq \alpha(T, T^*)$ **then**
            $T \leftarrow T^*$
        end
    end
    Get prediction $\hat{y}$ from T
    Simulate $\boldsymbol{p}$ values using Equation 3.4
    if *loop % thinning == 0* **then**
        Store tree data.
    end
end

3.5 A Simulated Example

It is useful to illustrate the features of Algorithm 4, used to explore the posterior distribution, on a simulated example. To this end, consider the following setting, taken from Chipman, George, et al. (1998):

$$y = f(x_1, x_2) + 2\epsilon, \quad \epsilon \sim N(0, 1) \tag{3.18}$$

with

$$f(x_1, x_2) = \begin{cases} 8.0 & \text{if } x_1 \leq 5.0 \text{ and } x_2 \in \{A, B\} \\ 2.0 & \text{if } x_1 > 5.0 \text{ and } x_2 \in \{A, B\} \\ 1.0 & \text{if } x_1 \leq 3.0 \text{ and } x_2 \in \{C, D\} \\ 5.0 & \text{if } 3.0 < x_1 \leq 7.0 \text{ and } x_2 \in \{C, D\} \\ 8.0 & \text{if } x_1 > 7.0 \text{ and } x_2 \in \{C, D\} \end{cases}$$

Figure 3.3 displays 800 iid observations drawn from this model. As the authors argue in their paper, this particular example tends to elude identification by greedy algorithms which minimize the residual sum of squares. The reason can be best seen in Figure 3.4. Note how a greedy procedure minimizing the RSS will prefer splitting on x_1 at the root node, while the true tree splits along x_2 first.

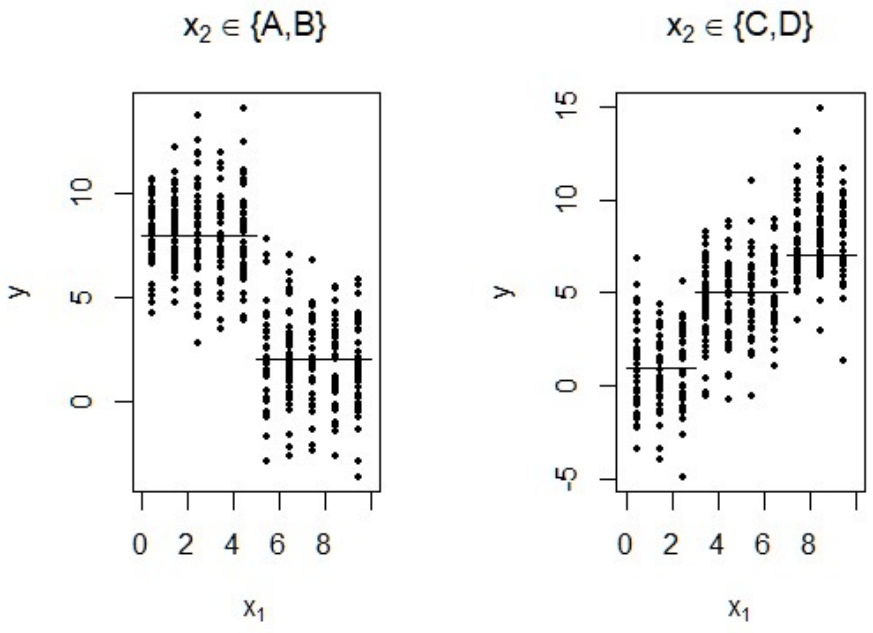


Figure 3.3: Simulated example with 800 iid datapoints and true model overlaid.

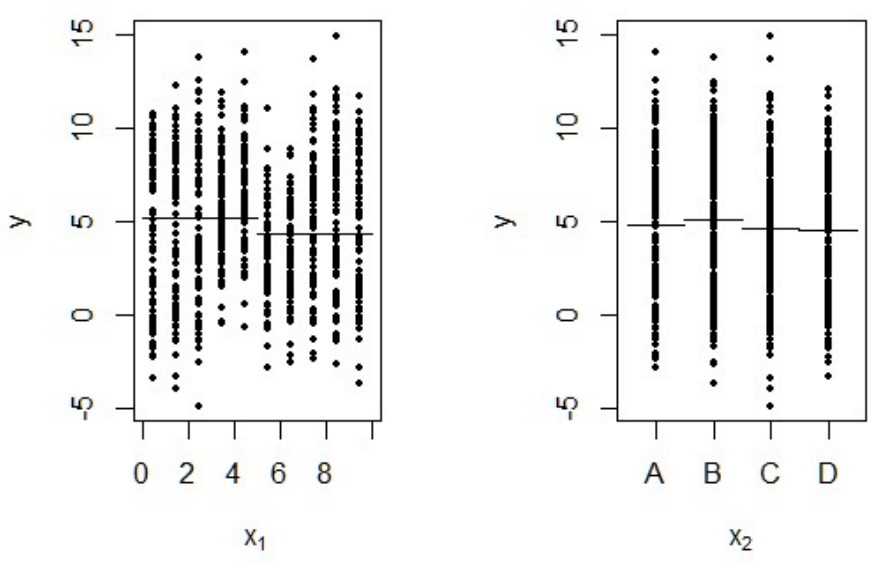


Figure 3.4: The response variable against the two predictors. Horizontal bars represent the mean response for the best split in x_1 plot, and mean levels for each value of x_2 in the other plot.

We proceed to replicate Chipman, George, et al. (1998) analysis using code written by the author of this manuscript to point out the features of the posterior space over trees, along with failures and difficulties of the sampling strategy. For this we run 10 different Markov chains of 1000 iterations long each started from the single node tree (i.e. a stump). Results are shown in Figure 3.5. We point out a number of connected remarks:

- The algorithm displays considerable stochasticity in the first few iterations. In fact each chain has been restarted from a stump, and has always converged to a different tree in all runs, notwithstanding the simple regression function here considered. This can be seen by noticing that the average number of terminal nodes is different for each run, without the need to analyze trees. In general, an high degree of initial stochasticity is good as it helps the algorithm to quickly forget the starting point.

- The algorithm quickly gravitates towards regions of high posterior probability and tends to stay there for the remaining iterations. In other words, it converges quickly onto a mode and is unable to escape. Multi-modality is confirmed by running multiple chains and observing the same behaviour with different tree characteristics (number of terminal nodes) and posterior probabilities, both indicating convergence

to different modes.

- The integrated likelihood tends to make little distinction between different modes, contrarily to the posterior probability, which down-weights trees that are too big.
- Overall, the posterior space is high-dimensional and exhibiting multi-modality, which make it difficult for the algorithm to escape local modes and to explore the surrounding space, hindering mixing of the chain.

The solutions proposed by Chipman, George, et al. (1998) is to exploit these features and run several short chains which will converge onto different modes, then compare most frequent trees across runs using the integrated log-likelihood (we refer the interested reader to their paper for an argument of why it is not recommended to compare using the posterior probability in such a scenario). Hence, although they built the model under a Bayesian framework, they then rely on a stochastic fitting procedure using the integrated log-likelihood for doing inference, rather than a rigorous exploration of the posterior space as is common in Bayesian analysis. This should not surprise, the difficulties mentioned above make the task of sampling from the posterior a particularly complex one. In the coming section we are going to leverage tempering to improve mixing and speed up convergence.

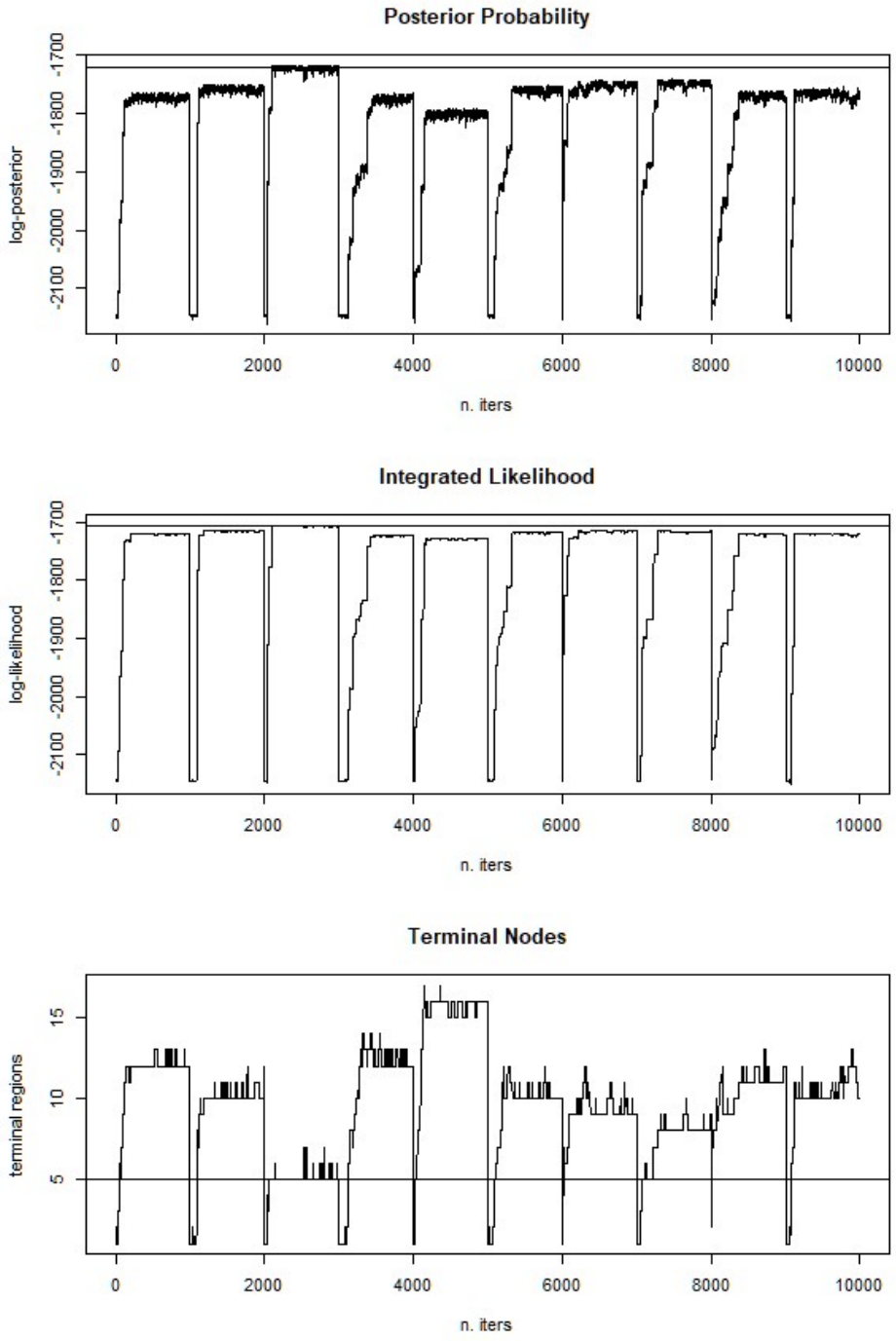


Figure 3.5: Simulation results. Every 1000th iterations the chain is restarted from a stump with a different seed. Horizontal lines represent corresponding values for the true tree. The posterior probability and integrated likelihood are shown in log terms.

Chapter 4

Tempering

As evidenced by the simulated example of Section 3.5, existence of multiple peaks on the posterior distribution separated by low-probability valleys can cause problems for MH algorithms, especially in terms of mixing and speed of convergence to the target distribution. In such instances, the Markov chain will fail to move rapidly throughout its support, remaining trapped for lengthy periods of time in local modes, unable to jump to other parts of the posterior surface. If properly constructed, the Markov chain will eventually cross even very deep valleys to explore other peaks (Tierney, 1994), however this can take a prohibitive amount of time. Consequently, several high-posterior trees may go unexplored by relying solely on Algorithm 4.

A number of methods have been developed over the years to improve mixing for MCMC application, see Gilks and Roberts (1996) for a review. In this section we are going to focus on a specific technique first introduced by Geyer (1991) under the name "Metropolis-coupled MCMC" (MCMCMC, or $(MC)^3$), also referred to as parallel tempering. It improves space exploration by running $m-1$ additional 'heated' chains in parallel to the target one (or 'cold' chain), and attempting to swap the state of the chains. Each heated chain explores a flattened version of the posterior landscape, allowing them to cross valleys with greater ease and propagate this information up to the cold chain through a sequence of swaps, eventually improving mixing for the cold chain itself. For a thorough explanation of $(MC)^3$, refer to Yang (2014, pp. 245-247) or later in this chapter. An idea closely related to parallel tempering is simulated tempering (Marinari and Parisi, 1992, Geyer and Thompson, 1995). Instead of running m chains in parallel, each targeting a different, but related, stationary distribution, simulated tempering runs them in series producing one long chain. Along the run, chains are randomly interchanged in a similar way to that of parallel tempering. Contrarily to $(MC)^3$, it requires estimation of normalizing constants (precise estimation is not necessary for a valid analysis), and it is claimed to perform better (Geyer, 2011). In particular, Geyer and Thompson (1995) give an example in which serial tempering seems to work for a hard problem, but its parallel counterpart fails. A technique similar to serial tempering was developed by Torrie and

Valleau (1977), albeit presented as a way to obtain stable estimates of expectations with respect to a wide range of distributions. Denoted as "umbrella sampling", it has been used to sample from all the m chains simultaneously and efficiently, rather than focusing only on the cold one, as in serial tempering. For a review of different tempering techniques, see Geyer (2011).

The application of tempering to Bayesian CART models has already been explored by Angelopoulos and Cussens (2005b), building on top of the methodology put forth in Angelopoulos and Cussens (2005a), namely the use of stochastic logic programming to design grow/prune moves that are able to prune off entire branches, thereby reducing mixing problems. Comparing their work with our approach, they combine both tempering and an improved proposal move which allows them to reach convergence to the posterior distribution. However, it requires knowledge of Prolog proof system and logic programs in order to be successfully implemented. Also, they constrain the proposal distribution to closely resemble the prior for the acceptance probability to simplify, which, as they point out, restricts the number of proposal/prior specifications that can be considered. For instance, they fail to implement the *change* and *swap* moves for MH algorithm. In this work we focus on the application of tempering to the original formulation of Bayesian CART and corresponding stochastic search procedure given by Chipman, George, et al. (1998). This gives us enough flexibility to introduce an ad-hoc tempering procedure, as well as the possibility to modify the prior distribution to provide a better pairing with this procedure, as suggested in Chapter 6. We conclude this review by mentioning the R package *MrBayes* (Altekar et al., 2004) which implements parallel tempering for phylogenetic inference; the simulations and analysis displayed later in this section rely on code fully implemented by the author of this manuscript.

We now give a formal treatment of $(\text{MC})^3$ in its generic formulation, following Geyer (1991) and Gilks and Roberts (1996); we shall consider extensions of this later on. Let $\pi(\boldsymbol{x})$ be the unnormalized density of some distribution of interest from which we wish to sample, and consider m MCMC chains each with different stationary distribution $\pi_i(\boldsymbol{x})$, $i = 1, ..., m$, where $\pi_1(\boldsymbol{x}) = \pi(\boldsymbol{x})$ and $\{\pi_i(\boldsymbol{x});\ i > 1\}$ are chosen to improve mixing. For simplicity, we further assume that each Markov chain takes values in the p-dimensional Euclidean space, so that $\boldsymbol{x} = (x_1, ..., x_p) \in \mathbb{R}^p$. We refer to the first chain as the 'cold' chain, and the others as the 'heated' chain, with higher values of i indicating 'hotter' chains, or chains having 'higher temperature'. The motivation behind such naming conventions is historical and based on an analogy between $(\text{MC})^3$ and simulated annealing (Kirkpatrick et al., 1983). The two algorithms are quite different, the latter being an optimization algorithm rather than a Monte Carlo, but it provides the useful metaphor of starting with 'heated' versions of the problem and slowly cooling down to the problem of interest. Moreover, they share the same insight of flattening the landscape to ease multimodality, where the flattening effect is controlled by the temperature parameter in simulated annealing (there is a physical intuition for this but we will not enter into further

details). Going back to $(MC)^3$, the m chains are updated in parallel and after one full iteration an attempt is made to swap the state of two of the chains using a Metropolis-Hastings step. For this, let $\boldsymbol{x}_t^i$ denote the state of chain i at iteration t, and suppose that a swap between chains i and j is proposed. Then the MH acceptance probability for the swap is:

$$\min\left\{1, \frac{\pi_i(\boldsymbol{x}_t^j)\,\pi_j(\boldsymbol{x}_t^i)}{\pi_i(\boldsymbol{x}_t^i)\,\pi_j(\boldsymbol{x}_t^j)}\right\} \tag{4.1}$$

The proposal distribution is implicitly assumed to be uniform over swaps and therefore excluded from the formula. At the end of the run, only the states from the cold chain are retained, those of the heated chains can be discarded.

For an effective result, modified chains should be chosen so that the MCMC sampler mixes well, in order to improve mixing for the cold chain by swapping states. For instance, at higher temperatures the chain should move freely around the space, allowing it to propagate states placed at different location of the posterior surface which would be impossible for the cold chain to reach. Moreover, proposed swaps will be rarely accepted if $\pi_i(\boldsymbol{x})/\pi_j(\boldsymbol{x})$ is very unstable, for example when few chains are used which differ drastically with i. To help select the correct spacing for i, it is recommended to space them out so that the acceptance rate of the chains is between 0.2 an 0.4, to optimize the speed of convergence for the cold chain (Kone and Kofke, 2005, Lingenheil et al., 2009, Atchadé et al., 2011). Automated alternatives for tuning chain spacing are also possible (Miasojedow et al., 2013).

The coming sections describe specific forms for $\pi_i(\boldsymbol{x})$ which we are going to focus on, together with alternative swapping strategies.

4.1 Geometric Tempering

Let us start by considering an example taken from Yang (2014) to foster intuition. Let $\pi(\boldsymbol{x})$ be a mixture of three Gaussian with modes at -2, 0 and 1.5, and standard deviation set to 0.1. The result is a distribution with three sharp peaks, as shown in Figure 4.1. As already mentioned, direct application of MH on such example would not yield the expected result, since it would fail to escape such sharp peaks and move to the others. However, by taking powers of the distribution we obtain a flattening effect that improves mixing. Formally, we consider incremental heating of the form:

$$\pi_i(\boldsymbol{x}) = \pi(\boldsymbol{x})^{1/t_i}, \quad 1 = t_1 < t_2 < \ldots < t_m$$

or, alternatively

$$\pi_i(\boldsymbol{x}) = \pi(\boldsymbol{x})^{\beta_i}, \quad \beta_i = 1/t_i \tag{4.2}$$

We employ this latter notation, even though the former makes a clear parallel with the heating analogy mentioned before, and we refer to Eq. 4.2 as geometric tempering. Note

that alternative formulations do exist, and may sometimes perform better; see Geyer and Thompson (1995).

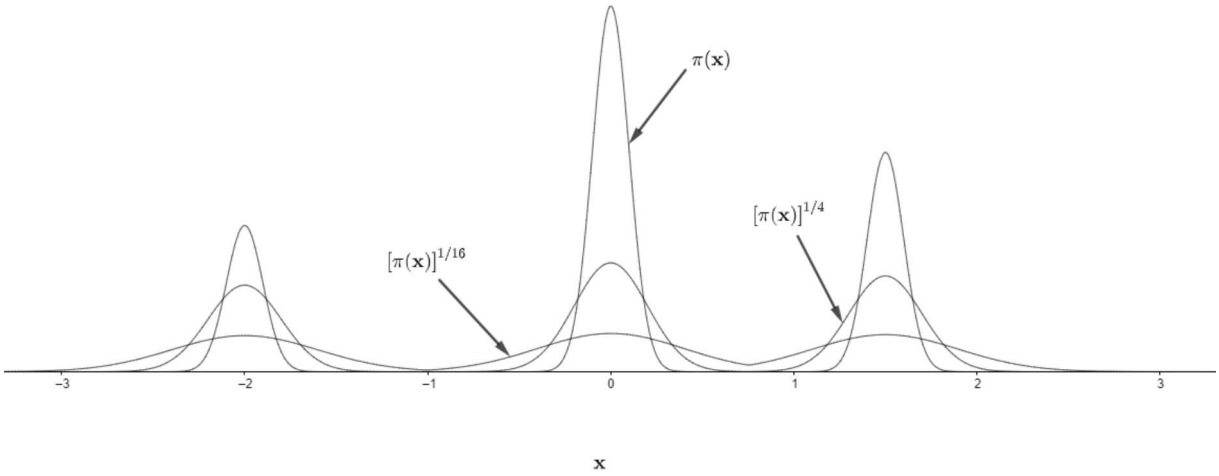


Figure 4.1: Mixture of three Gaussians $\pi(\boldsymbol{x})$ with means -2, 0, 1.5 and variance 0.001. Overlaid are two flattened densities proportional to $[\pi(\boldsymbol{x})]^{1/4}$ and $[\pi(\boldsymbol{x})]^{1/16}$

Under geometric tempering, the acceptance probability of a swap between chain i and $i+1$ (Eq. 4.1) takes the following form

$$\min\left\{1, \left(\frac{P(Y|X,T_{i+1})P(T_{i+1}|X)}{P(Y|X,T_i)P(T_i|X)}\right)^{B_i} \left(\frac{P(Y|X,T_i)P(T_i|X)}{P(Y|X,T_{i+1})P(T_{i+1}|X)}\right)^{B_{i+1}}\right\} \tag{4.3}$$

While the MH acceptance probability for moves within chain i is

$$\alpha(T,T^*) = \min\left\{\frac{q(T^*,T)}{q(T,T^*)}\left(\frac{P(Y|X,T^*)P(T^*)}{P(Y|X,T)P(T)}\right)^{\beta_i}, 1\right\}$$

Note that $\alpha(T,T^*)$ does not enjoy the same properties detailed in Section 3.3.4 since it is no longer possible to simplify the transition ratio $q(T^*,T)/q(T,T^*)$ with the tree structure ratio $P(T^*)/P(T)$. They must be computed in full which increases book-keeping and negatively impacts algorithm speed, albeit marginally. The complete procedure to sample from the posterior distribution using tempering and stochastic swap type (ref. Section 4.3) is listed in Algorithm 6.

Algorithm 6: Posterior Sampling for Bayesian CART Models with Tempering

Data: Target variable y (length n), feature matrix X (n rows, p columns)
Result: Posterior list of trees vor cold chain

Initialization
Number of chains: m
Temperature progression: $1 = \beta_1 > ... > \beta_m$
Repeat Algorithm 4 initialization
Set initial trees $T_1, ..., T_m$ to stumps

for $loop \leftarrow 1$ **to** N **do**
  for $chain \leftarrow 1$ **to** m **do**
    Execute one full step of Algorithm 4.
    Compute $\alpha(T_{chain}, T^*_{chain})$, in case of acceptance of T^*_{chain}, replace T_{chain} with T^*_{chain}.
  end
  Generate $u \sim U(0, 1)$
  if $u \leq 0.5$ **then**
    $chain\_seq \leftarrow$ sequence of even chain indexes
  else
    $chain\_seq \leftarrow$ sequence of odd chain indexes
  end
  for $chain$ **in** $chain\_seq$ **do**
    $swap\_p \leftarrow$ compute acceptance probability of swapping T_{chain} and $T_{chain+1}$, see **??**
    Generate $u \sim U(0, 1)$
    if $u \leq swap\_p$ **then**
      execute swap
    end
  end
  if loop % thinning == 0 **then**
    Store trees data.
  end
end

4.2 Shrinkage Tempering

The second formulation for $\pi_i(\boldsymbol{x})$ that we consider is rather different as the heated chains are not flattened. Instead, it seeks to retain some roughness information about the space to make more informed moves at hotter chains, and exploits model specific properties to improve mixing. This can be accomplished by refraining from acting on both the likelihood and the prior, as done in geometric tempering, and focusing instead on the latter by over-emphasizing its effect in hotter chains. We refer to this technique as 'shrinkage tempering'.

To see how it works, recall that the underlying reason why MH fails under multimodality in our setting is that space exploration proceeds by a sequence of local moves, rather than global ones. Local moves for a tree are splitting a terminal node, pruning one or attempting to change/swap splitting rules. Instead, global moves would entail for example growing or pruning entire branches, wildly changing the tree structure. Unfortunately, defining such moves while retaining reversibility for the algorithm is not trivial. Then, splits made in the early stages of the algorithm, those close to the root, are rarely if ever revisited. In the simulated example of Section 3.5 this is particularly troublesome because the "correct" split (along the categorical variable) is somewhat masked by that along the quantitative variable, so that the tree tends to start off with the least promising split. Following this intuition, we would like hotter chains to have on average smaller trees where it is generally easier through a sequence of steps to change tree configuration close to the root. Ideally, the hottest chain would generate trees with few terminal regions, striking a balance between the ability to change splitting rule at the root node (easier with fewer leafs) and keeping m as low as possible for computational concerns (a lower chain spacing in terms of terminal regions requires a bigger m for the cold chain to have the same number of leafs, on average). By swapping with the hottest chain, the $m-1$ chain has the chance to "restart" from small, potentially better trees and grow additional nodes until reaching the average number of leafs for chain $m-1$. Obtaining global changes using local moves is still difficult, so the tree will tend to be modified only about the new leafs, looking for the best configuration given the roots using a stochastic search for the best splits that leverages roughness information provided by the likelihood. The process is then repeated one chain up in parallel. To control the average tree size for different chains we can rely on the corresponding component of the prior distribution, $p_{SPLIT}(\eta, T)$, which is indexed by two parameters α and β. They should be optimized so to have acceptance rates between 0.2 and 0.4, as previously mentioned, which generally means having some overlap between chains in terms of terminal nodes. It is easy to see why: looking back at Figure 3.1, a tree with 20 leafs in chain i having $\alpha_i = 0.95$, $\beta_i = 0.5$ has a discrete probability of being observed, at least from a prior perspective. If we attempted to swap with chain j having $\alpha_j = 0.95$, $\beta_j = 1.5$, the prior would sensibly penalize such a tree, making the swap more difficult.

4.2.1 Swap Acceptance Probability

Consider a sequence of parameters $(\alpha, \beta)_{i=1}^m$ and let $\gamma_i = (\alpha, \beta)_i$. We make explicit the dependence between the prior distribution and γ as follows $P(T|X, \gamma)$. It is useful to interpret the sequence $\gamma_{i=1}^m$ to have the same meaning as the temperatures for geometric tempering. The acceptance probability of a swap between chain i and $i+1$ is

$$\min\left\{1, \frac{P(T_{i+1}|X, Y, \gamma_i)}{P(T_i|X, Y, \gamma_i)} \frac{P(T_i|X, Y, \gamma_{i+1})}{P(T_{i+1}|X, Y, \gamma_{i+1})}\right\} \tag{4.4}$$

where the fraction on the left is the probability that chain i accepts tree T_{i+1}, and that on the right the probability that chain $i+1$ accepts T_i. Writing the posterior distribution in full we obtain

$$\min\left\{1, \frac{P(Y|X, T_{i+1})P(T_{i+1}|X, \gamma_i)}{P(Y|X, T_i)P(T_i|X, \gamma_i)} \frac{P(Y|X, T_i)P(T_i|X, \gamma_{i+1})}{P(Y|X, T_{i+1})P(T_{i+1}|X, \gamma_{i+1})}\right\}$$

Note that the integrated likelihoods are unchanged under shrinkage tempering, so they simplify.
Let $\tilde{T}_i$ be the set of terminal nodes of the tree in the ith chain and similarly let $(T - \tilde{T})_i$ be the set of internal ones. Consider $P_{RULE}(\rho_\eta|\eta, T_i)$ as defined in Section 3.1.2, where ρ_η is taken to be the splitting rule assigned to node η in the relevant tree considered (here would be T_i). Finally, we extend the notation for p_{SPLIT} to include explixit dependence on γ: $p_{SPLIT}(\eta, T_i|\gamma_i)$. Then we have the following:

$$\begin{aligned}\min\Bigg\{1, &\frac{\prod_{\eta\in(T-\tilde{T})_{i+1}} p_{RULE}(\rho_\eta|\eta, T_{i+1})\; p_{SPLIT}(\eta, T_{i+1}|\gamma_i)}{\prod_{\eta\in(T-\tilde{T})_i} p_{RULE}(\rho_\eta|\eta, T_i)\; p_{SPLIT}(\eta, T_i|\gamma_i)} \times \\ &\frac{\prod_{\eta\in(T-\tilde{T})_i} p_{RULE}(\rho_\eta|\eta, T_i)\; p_{SPLIT}(\eta, T_i|\gamma_{i+1})}{\prod_{\eta\in(T-\tilde{T})_{i+1}} p_{RULE}(\rho_\eta|\eta, T_{i+1})\; p_{SPLIT}(\eta, T_{i+1}|\gamma_{i+1})} \times \\ &\frac{\prod_{\eta\in\tilde{T}_{i+1}}(1 - p_{SPLIT}(\eta, T_{i+1}|\gamma_i))}{\prod_{\eta\in\tilde{T}_i}(1 - p_{SPLIT}(\eta, T_i|\gamma_i))} \frac{\prod_{\eta\in\tilde{T}_i}(1 - p_{SPLIT}(\eta, T_i|\gamma_{i+1}))}{\prod_{\eta\in\tilde{T}_{i+1}}(1 - p_{SPLIT}(\eta, T_{i+1}|\gamma_{i+1}))}\Bigg\}\end{aligned}$$

In the above, p_{RULE} simplifies, while p_{SPLIT} does not because of the presence of chain-specific parameters γ_i and γ_{i+1}

$$\begin{aligned}\min\Bigg\{1, &\frac{\prod_{\eta\in(T-\tilde{T})_{i+1}}\; p_{SPLIT}(\eta, T_{i+1}|\gamma_i) \prod_{\eta\in\tilde{T}_{i+1}}(1 - p_{SPLIT}(\eta, T_{i+1}|\gamma_i))}{\prod_{\eta\in(T-\tilde{T})_i}\; p_{SPLIT}(\eta, T_i|\gamma_i) \prod_{\eta\in\tilde{T}_i}(1 - p_{SPLIT}(\eta, T_i|\gamma_i))} \times \\ &\frac{\prod_{\eta\in(T-\tilde{T})_i}\; p_{SPLIT}(\eta, T_i|\gamma_{i+1}) \prod_{\eta\in\tilde{T}_i}(1 - p_{SPLIT}(\eta, T_i|\gamma_{i+1}))}{\prod_{\eta\in(T-\tilde{T})_{i+1}}\; p_{SPLIT}(\eta, T_{i+1}|\gamma_{i+1}) \prod_{\eta\in\tilde{T}_{i+1}}(1 - p_{SPLIT}(\eta, T_{i+1}|\gamma_{i+1}))}\Bigg\}\end{aligned} \tag{4.5}$$

The formula above may seem complex, but it simply states that the acceptance probability depends on the ratio of the 'structural probability' of a tree under each chain, where with 'structural probability' we mean the probability of growing a binary tree where each terminal node splits with probability p_{SPLIT} and splitting rule assignment is ignored.

4.2.2 MH Acceptance Probability

As we have seen in Section 3.3.4, when computing the MH acceptance probability within a chain, there is substantial simplification between the transition ratio $q(T^*, T)/q(T, T^*)$ and the structure ratio $P(T^*)/P(T)$, for each move type. This is preserved under shrinkage tempering because the acceptance probability formula is unchanged, only the values of (α, β) change. Thanks to the many simplifications in computing acceptance probabilities, shrinkage tempering results six to seven times faster than geometric tempering. The lower average size of trees in the heated chains also has a major impact. The algorithm for shrinkage tempering is very similar to Algorithm 6, and therefore not shown.

4.3 Alternative MH Swap Types

The generic formulation of tempering described earlier in this chapter attempts to swap the state of two chains via a MH step. Due to chain spacing, swaps between chains that are far apart are rarely accepted and we usually consider swaps between adjacent chains only. Note that this may not always be optimal. For instance, under shrinkage tempering it has been observed from simulations that sometimes the chain may end up in a local mode having a sensibly higher number of leafs for the pair of (α, β) used. Recall that every such pair induces a different posterior and a corresponding marginal distribution for the terminal nodes; running the chain several times can give an intuition about this object. Therefore, if the chain converges to a local mode consisting of higher or lower than usual leaves, it will have difficulties in exchanging states with the adjacent chains due to a lack of overlap. This behavior has been observed only at the very beginning of the simulation; if chains manage to swap since the beginning no case has been reported of them getting stuck at some later point. In any case, non-adjacent swaps may help nonetheless in such a scenario.

One could argue that attempting multiple swaps every iteration, rather than just one, could speed up mixing thanks to faster exchanges and quicker propagation of trees to the cold chain. This intuition seems backed up empirically, at least for the model here considered. Then, it is useful to attempt all non-overlapping swaps every iteration. These are either swaps between odd to even chains ($i \to i+1,\ i$ odd), or even to odd chain ($i \to i+1,\ i$ even). We refer to such scheme as Even-Odd (E-O) moves. The E-O scheme itself does not alter the reversibility of the MH algorithm, however its implementation might; we consider two of them.

4.3.1 Stochastic E-O moves

Stochastic E-O moves (SEO) is a direct generalization of single swaps which retains reversibility. Every iteration it chooses between even or odd chains swap with a probability of 0.5 for each. This swap type is used in Algorithm 6. To understand how it performs, consider the index process displayed in Figure 4.2, taken from Syed et al. (2019). Loosely speaking, the index process tracks how the state of the corresponding chain evolves over time thanks to the swap moves (actually it keeps track of the annealing parameters, such as the temperature t_i, as in a distributed context it is faster to exchange them rather than states because these may be high dimensional or complex and require longer transmission times over the network). In Figure 4.2, note how the index process highlighted in the simulation on the right does not manage to traverse all the chains in the timespan considered.

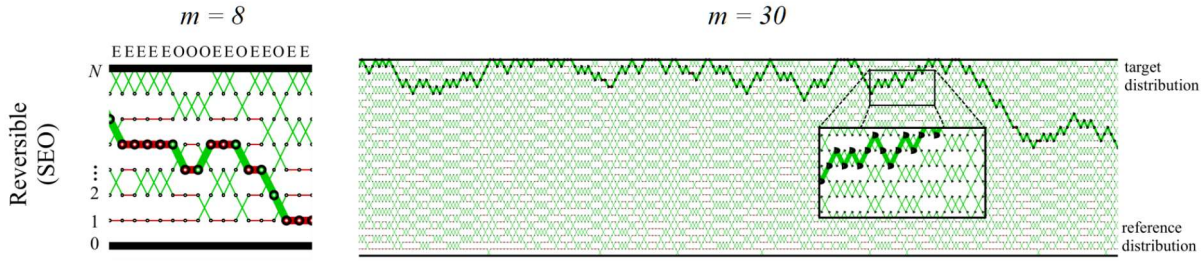


Figure 4.2: Reversible swap type with $m = 8$ (left) and $m = 30$ auxiliary chains (right) using equally spaced annealing parameters on a Bayesian change-point detection model. The sequence of swap moves forms N + 1 index process trajectories (paths formed by the red and green edges); one such path is shown in bold. Figure taken from Syed et al. (2019).

4.3.2 Deterministic E-O moves

Deterministic E-O swap type (DEO) attempts to improve the behavior of the index process by deterministically alternating even and odd moves every iterations. For instance, one could swap even chains at even iteration number, and odd chains otherwise. Figure 4.3 compares the performance of the two. The downside of DEO is that it is not reversible, however it still converges to the target distribution. Additionally, for optimal performance, the authors suggest to tune the algorithm using an alternative metric, rather than the acceptance rate. For more details on both these points, we refer to the original paper (Syed et al., 2019).

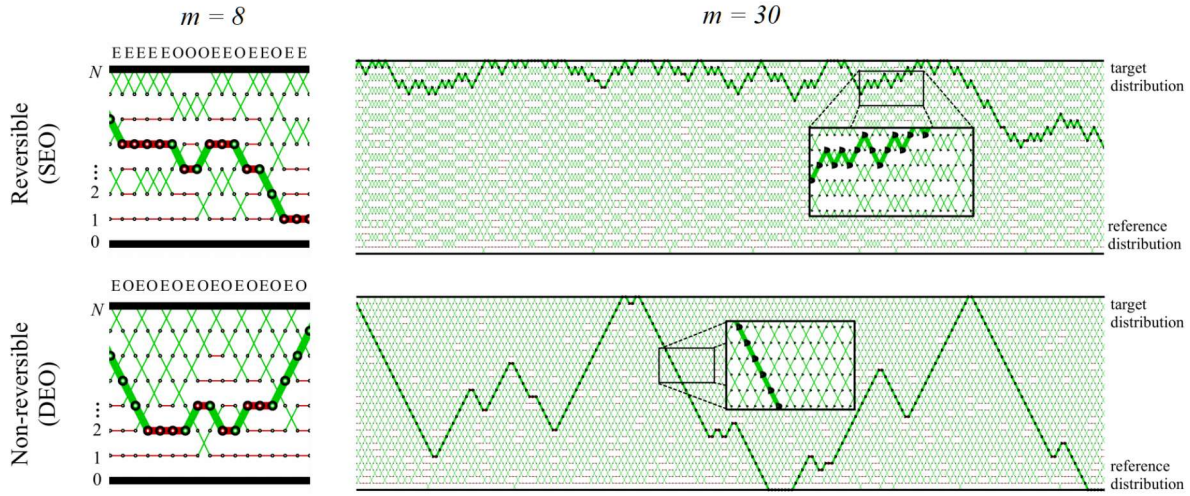


Figure 4.3: Reversible (top) and non-reversible swap type (bottom) with $m = 8$ (left) and $m = 30$ auxiliary chains (right) using equally spaced annealing parameters on a Bayesian change-point detection model. Figure taken from Syed et al. (2019).

Chapter 5

Preliminary Results

In this section we attempt to use geometric and shrinkage tempering to sample from the posterior distribution of Bayesian CART models. We wish to remark that what is presented here are merely preliminary results, giving insights into the inner workings of such algorithms and hinting at their performance and drawbacks. We are conscious that further research is needed to make any definite claim regarding our objective, including using more quantitative techniques to asses convergence, and considering real-world data analysis examples, among other things; we defer that to a later work. Nonetheless, even preliminary results are important to gauge whether a certain research direction has potential and is worth pursuing, or if the intuition is not verified and in practice things do not work out.

To compare performance of the various procedures detailed in the previous chapters, we run different simulations for each algorithm on a dataset of 800 observation sampled from Eq. 3.18. Each simulation consists of 8 independent runs started using different random seeds, with the following hyper-parameters

$$\alpha = 0.95 \quad \beta = 1 \quad a = 1/3 \quad \bar{\mu} = 4.85 \quad \nu = 10 \quad \lambda = 4$$

Moreover, we set a minimum number of observations for each leaf to 5, and provide equal weights for each MH move (*grow*, *prune*, *change*, *swap*). Weights can be used to prefer a move type over another, when sampling them in Algorithm 4. Geometric tempering was run using the following temperature progression

$$(\beta_i)_{i=1}^{8} = (1, 0.85, 0.7, 0.48, 0.31, 0.2, 0.08, 1e^{-7})$$

while shrinkage tempering was set with the pairs

$$\begin{aligned}\big(\alpha_i, beta_i\big)_{i=1}^{15} = \big(&(0.95, 1), (0.35, 0.75), (0.35, 1.45), (0.25, 2.1), (0.3, 3.1),\\ &(0.3, 4), (0.3, 5), (0.3, 5.9), (0.4, 7.4), (0.4, 9.08),\\ &(0.5, 9.92), (0.35, 10.5), (0.35, 11.7), (0.35, 13), (0.35, 14.3)\big)\end{aligned}$$

These temperature progressions have been empirically selected to optimize the relevant metrics mentioned in the corresponding sections.

Runs are important to analyze convergence because the algorithm gets stuck in different local modes every time it is restarted, as seen in the simulated example of Section 3.5. By comparing marginal distributions of features of interest (e.g. terminal nodes), one can easily see if there are convergence problems. The converse is not necessarily true: marginal plots for a scalar quantity of interest may all look similar across runs, but convergence has not been achieved. Multiple plots for a range of scalar attributes may give more confidence in the result. Table 5.1 summarizes the various simulations tried. Note in particular the number of iterations per chain, which has been set so that every run executes the same number of MH steps, for fair comparison. The table also displays a proxy of the running time for each algorithm ("iters/min"), intended to be used only as a rough measure, as no specific procedure has been set to provide for an optimal comparison, other than running each simulation on the same machine.

Simulation type	runs	chains	swap type	iters/chain	MH steps	iters/min (avg)
no tempering	8	1	None	1.2M	1.2M	9898
Geometric	8	8	SEO	150K	1.2M	3814
Geometric	8	8	DEO	150K	1.2M	4034
Shrinkage	8	15	SEO	80K	1.2M	25589
Shrinkage	8	15	DEO	80K	1.2M	30282

Table 5.1: Summary of simulations for three different algorithms: no tempering (Algorithm 4), geometric and shrinkage tempering. "iters/chain" indicates the number of MH moves carried out per chain, while "MH steps" is the overall amount. "iters/min" is a rough measure of execution speed of each algorithm, showing the number of MH moves done by the algorithm per minute (on average).

5.1 Without Tempering

In what follows, we focus on the number of leaves as the main attribute used to gauge convergence; that will be enough for the scope of this chapter concerning preliminary results.

Consider Algorithm 4 and the simulated example of Section 3.5. The simulation in this section can be considered an analogous of Figure 3.5, where each chain is run far longer, to let enough time for the algorithm to converge (hopefully). Figure 5.1 displays

the evolution of the number of terminal regions for eight runs, overlaid. We note the same issues that we have already pointed out, namely the chains fail to explore the posterior space successfully due to sharp peaks in the posterior space. Take for example the chain at the top, virtually none of the trees it visits have been explored by the other seven chains.

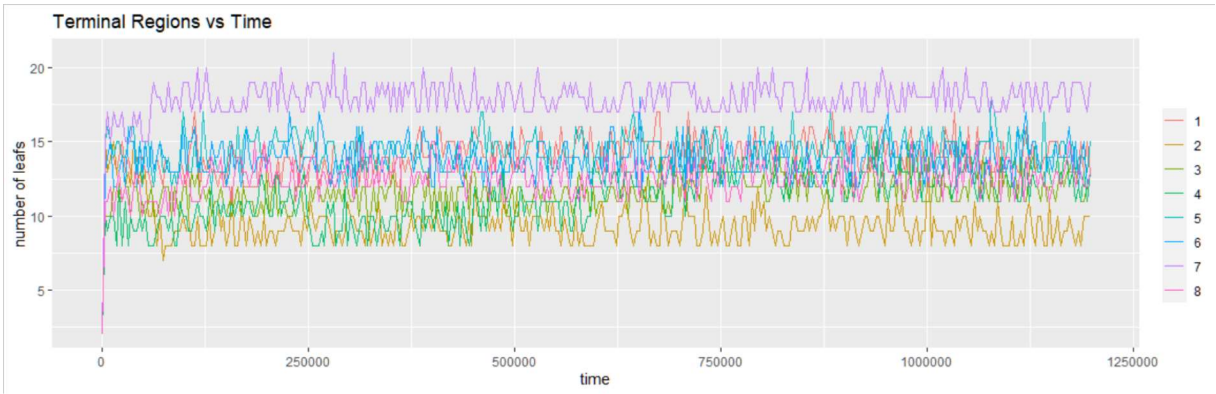


Figure 5.1: No tempering. Evolution of the number of terminal regions for eight different runs of the algorithm, overlaid. Multi-modality and mixing issues stand out clearly.

For a cleaner view of the same plot, consider the (marginal) posterior distribution of leaves in each run, placed one next to the other for easier comparison, in Figure 5.2. Upon convergence, all marginal distributions should be nearly identical. Instead, sensible variation is observed.

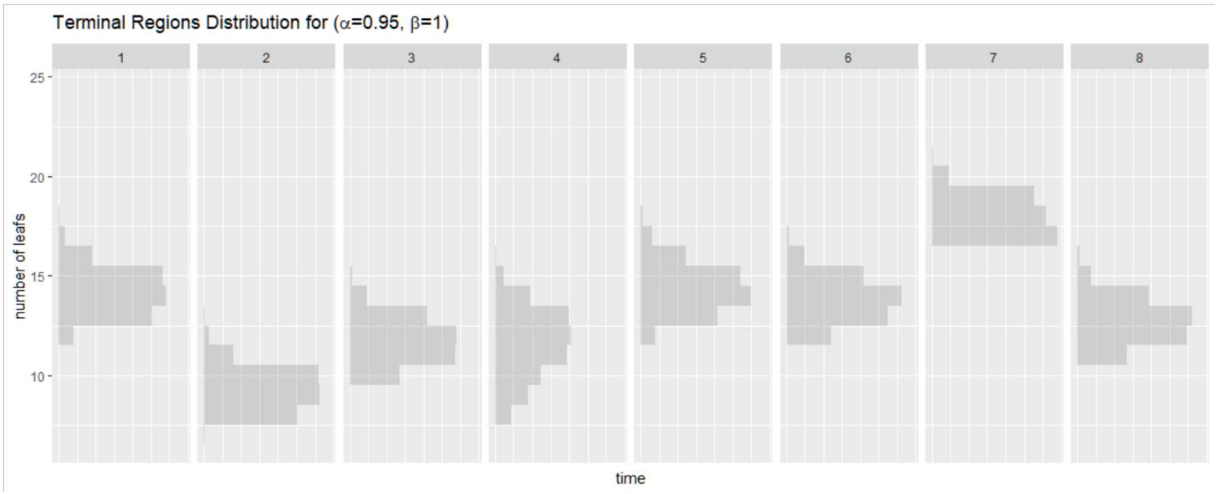


Figure 5.2: No tempering. Marginal distribution of the number of terminal regions for each chain, side by side, computed after removing burn-in.

Finally, it is insightful to look at the five most frequently visited trees for each run. Their characteristics can be summarized as follows:

- The most frequent tree compared across runs has consistently different number of terminal regions. We would expect the most likely tree a posteriori to have approximatively the same sampling probability in each run, had they converged.
- The most frequent tree compared across runs tends to have different splitting rules for the first node, both in terms of splitting variable, but also for the splitting values when the two variables match. Again, this is a symptom of failed convergence, seen from another perspective (a different attribute of the tree).
- Within the same chain, the split rule for the root node is virtually the same for the most visited trees (up to the 5th most common, covering on average 80% of the sampled trees for each chain). This suggests that the algorithm has great difficulties in changing the splitting rule close to the root node: once the tree is grown to a specific configuration early in the simulation, the roots are basically left untouched in the majority of the trees explored later, due to our specification of MH moves that only manage to work locally.
- In seven runs out of eight, the two most common trees amount for 40-80% of all the trees visited. By itself this information is not very insightful, however it is sensibly higher when compared with that of the tempered simulations. A possible interpretation is that fewer, more "concentrated" trees are explored.

We take these results as our benchmark, and compare it with tempering to investigate potential of this technique.

5.2 Geometric Tempering

Similarly to the previous section, we considering convergence results when using geometric tempering with SEO moves. DEO swap moves exhibit similar performance.

Before jumping to the results, we try to inspect some of the inner workings of such algorithm. Every row in Figure 5.3 displays the evolution of terminal regions for each chain, where 1 is the cold chain and 8 is the hottest one. Every row is a different run of the algorithm. Note how trees at higher temperatures have a sensibly higher number of terminal nodes on average, suggesting that valleys in between peaks are filled by particularly large trees, and flattening the space allows the sampler to explore such regions as well (flattening effect is stronger the higher the temperature). This may not be optimal for this specific model: the bigger the tree, the smaller is the impact of a MH step. Hence, if one managed to have the same mixing properties without passing by large trees, one would obtain an algorithm that is able to almost exclude valley exploration,

which have close to no posterior weight, thereby saving computational time. This is the motivation behind shrinkage sampling.

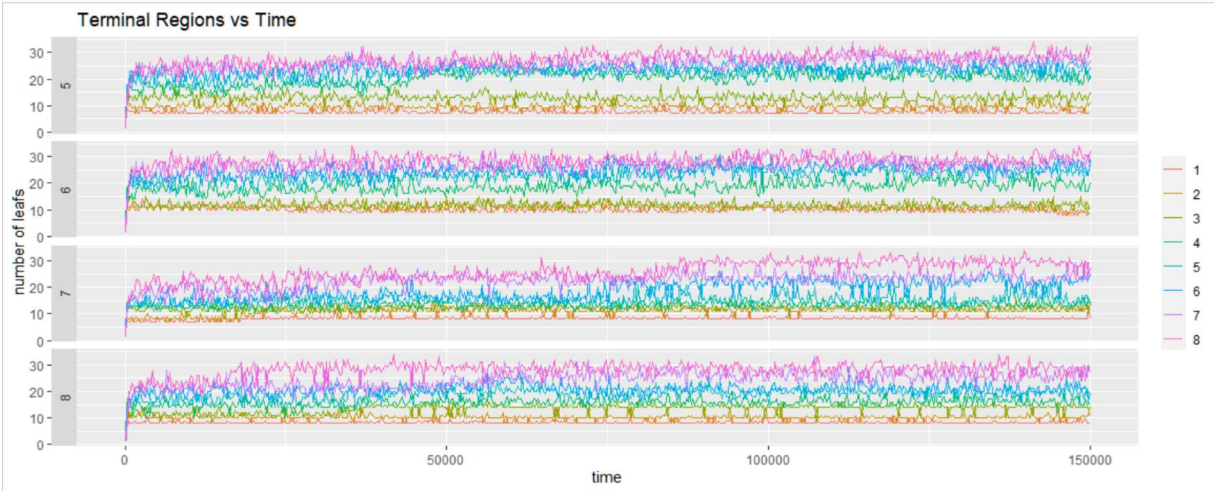


Figure 5.3: Geometric tempering with SEO. Each row: evolution of number of leaves for each chain of a run. A total of four different runs are displayed.

The marginal distribution of terminal nodes across runs for geometric tempering is displayed in Figure 5.4. Runs 1, 7 and 8 have similar, very concentrated distributions, but failure of convergence is still obvious even by focusing on just one attribute. However, judging merely from Figure 5.4, one could suppose that tempering does help in improving mixing with respect to our benchmark. This is further supported by analyzing the most frequent trees:

- The most common tree across runs tends to split consistently along the categorical variable at the root node, contrary to what is observed for shrinkage or no tempering. Recall that splitting along the qualitative feature is fundamental in order obtain the smallest, "true", partition of the sample space. Therefore, by comparison with the other simulations, there is evidence to suggest that geometric tempering is able to find the best split variable for the root node, whether by immediately splitting on it or changing it at a later time.

- The most common tree across runs tend to identify (A, B) as first split (the optimal split for the root node). Moreover, for five runs the first three splits of the most common three in each run are almost equal.

- Within-run variability of the splitting rule at the root node is higher than shrinkage or no tempering, suggesting again that the chain is able to modify the roots grown in the early stages of the algorithm.

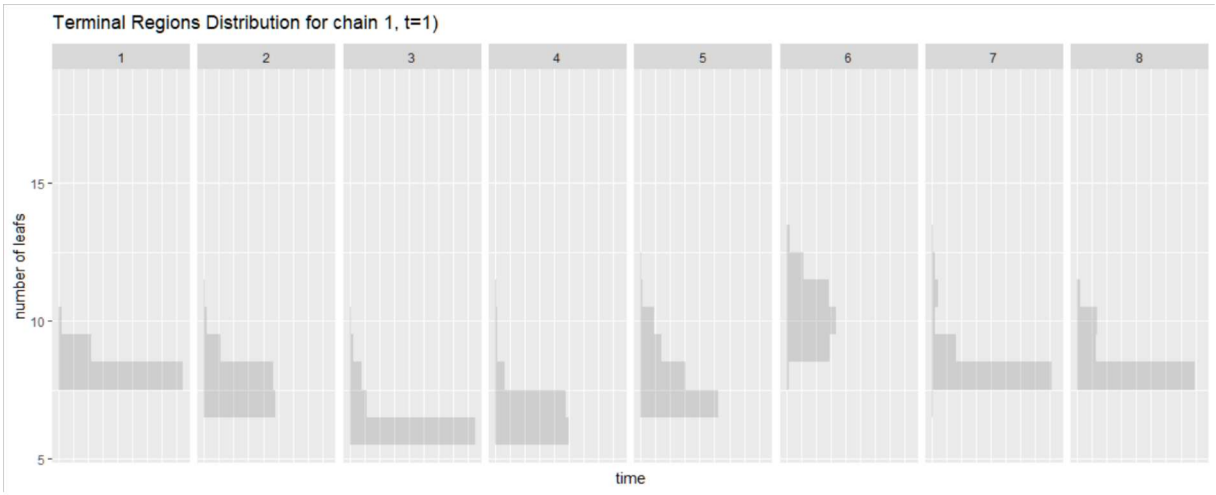


Figure 5.4: Geometric tempering with SEO. Marginal distribution of the number of terminal regions of the cold chain, for each run, computed after removing burn-in.

5.3 Shrinkage Tempering

As for the previous section, we begin by describing the inner workings of shrinkage tempering using SEO moves; an analysis of deterministic E-O swap moves will follow. Recall that hotter chains are tuned to have lower number of posterior terminal regions on average, so that each chain can affect the final tree at a specific depth. For simplicity, think of it as an onion, where the core is the root of the tree and every layer consists of additional nodes built on top of the previous nodes. Every chain then appends one layer and modifies it via MH steps, finding the optimal set of splitting rules and tree configuration given the previous layers. The whole onion then represents the final tree as delivered to the cold chain. Empirically, this intuition is reflected in Figure 5.5. Note how hotter chains sample smaller trees, on average.

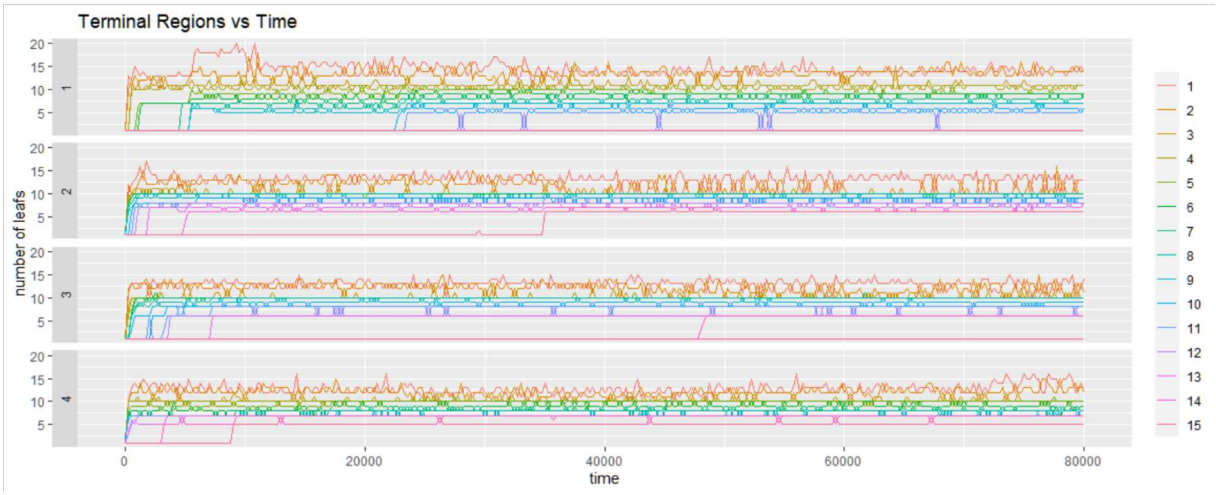


Figure 5.5: Shrinkage tempering with SEO. Each row: evolution of number of leaves for each chain of a run. Chains tend to arrange themselves from cold (top) to hotter (bottom). A total of four different runs are displayed.

Mixing results are displayed in Figure 5.6. Although improving with respect to the benchmark, there are still clear convergence issues. Most frequent trees analysis yields similar results than geometric tempering: there is greater variability for the splitting rule of the first few nodes for a given run; the first split across runs is more consistent, both in terms of splitting variable and splitting values (although the first split is along the quantitative variable, contrarily to geometric tempering). Additionally, within-run trees tend to be less concentrated than geometric or no tempering, meaning that that there are more smaller groups of unique trees rather than few big groups that make up the vast majority of sampled trees. This is another indication of faster mixing.

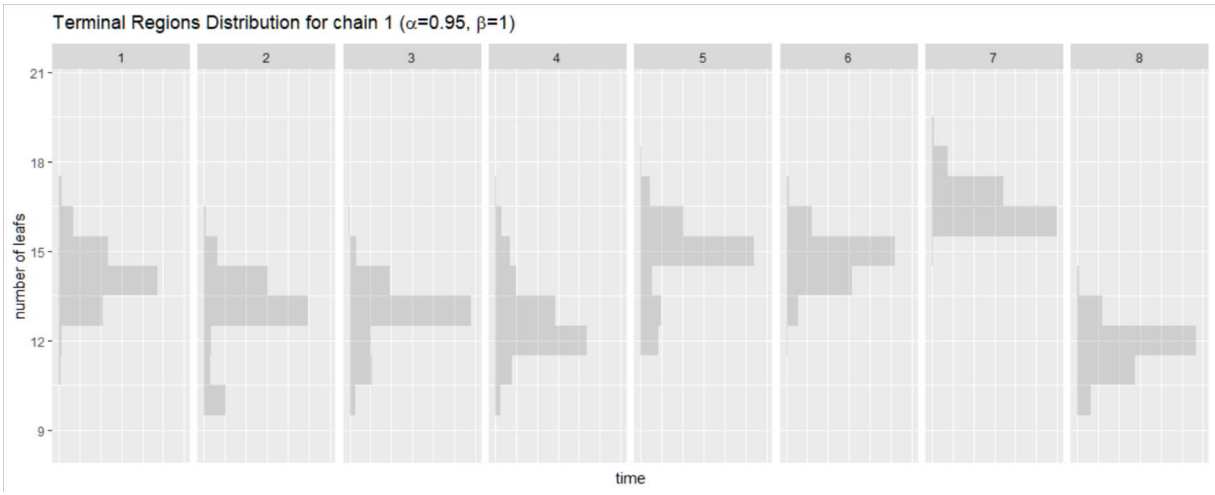


Figure 5.6: Shrinkage tempering with SEO. Marginal distribution of the number of terminal regions of the cold chain, for each run, computed after removing burn-in.

We conclude this section by commenting simulation results for shrinkage tempering using DEO moves. As can be seen in Figure 5.7, there is a sensible improvement with respect to the SEO case: for some runs, the marginal distribution of terminal nodes is sparser (runs 3,4,6,7), however the others are quite concentrated and the marginals are far from being the same. For instance, the vast majority of trees sampled during run 1 for the cold chain have never been visited by runs 5 and 6. Major convergence issues still persist.

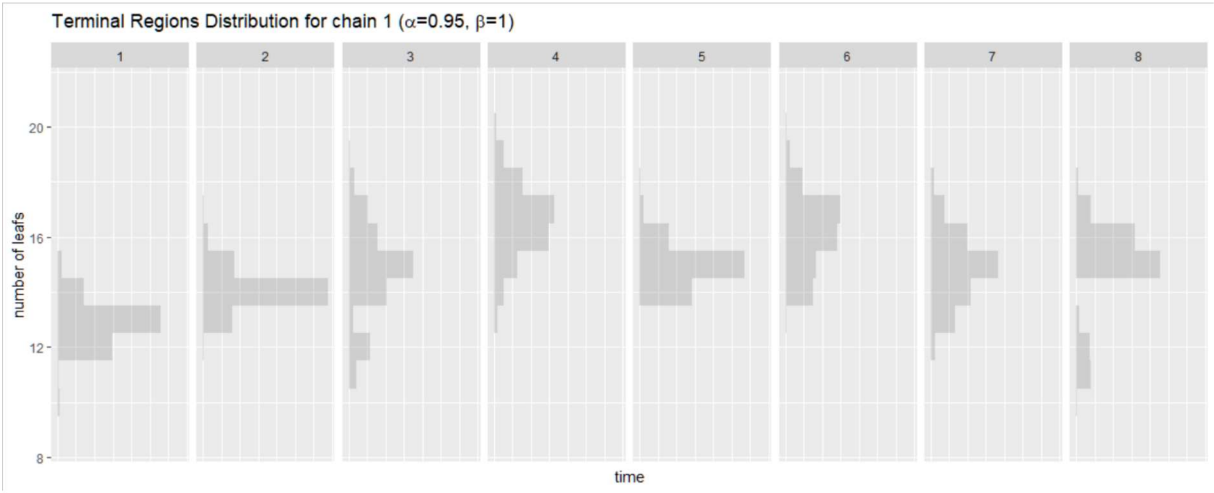


Figure 5.7: Shrinkage tempering with DEO. Marginal distribution of the number of terminal regions of the cold chain, for each run, computed after removing burn-in.

Chapter 6

Further Research

In this work we have reviewed CART models following the original discussion by Breiman et al. (1984), and built on such knowledge to discuss at length their Bayesian extension as given by Chipman, George, et al. (1998). We have commented on the advantages of Bayesian CART models over their greedy counterparts in inference tasks, where model interpretability is more important than predictive accuracy. However, notable computational issues arise in exploring the posterior distribution due to multimodality. As a possible solution we have explored the use of $(MC)^3$, and introduced a novel, ad-hoc tempering procedure for Bayesian CART that exploits the structure of the model to produce a more computationally efficient sampler, thereby improving mixing. Preliminary results suggest the speed-up is obtained by avoiding sampling trees in low-probability valleys, which is instead done by $(MC)^3$'s heated chains in order to move between peaks. Even though both samplers exhibit improved mixing compared to the single-chain MH procedure, they fail to converge to the posterior distribution for the simulated example considered, showing that further work is needed. We conclude by listing potential future development for our work:

- Adopt more robust ways to carry out convergence analysis, such as considering multiple scalars attributes of interest, and relying on more quantitative tests (see Cowles and Carlin (1996) and Brooks and Roberts (1998) for a review of qualitative and quantitative techniques). Note that, as discussed in Brooks and Roberts (1998), any convergence diagnostic procedure cannot be guaranteed to successfully diagnose convergence, hence they should not be unilaterally relied upon. Comparing results for a few techniques and inspecting the sampler output from multiple perspectives leveraging some understanding of the target distribution can give more confidence in the result.

- Apply geometric and shrinkage tempering to real-world datasets and compare convergence findings.

- Modify the prior distribution $p(T)$ to provide a better pairing with the shrinkage tempering approach, for example by removing the inner bias for bushy trees and to provide an easier way to control the tree size (as in Wu et al., 2007).
- Incorporate alternative approaches coming from the literature that already improve mixing, such as improved prior specification or more efficient proposals for the MH algorithm (see introduction of Chapter 3).
- Design techniques to efficiently use the posterior distribution over trees for inference tasks, such as introducing the notion of 'average tree' and of a 'confidence interval' for trees.

Bibliography

Altekar, Gautam et al. (2004). "Parallel metropolis coupled Markov chain Monte Carlo for Bayesian phylogenetic inference". In: *Bioinformatics* 20.3, pp. 407–415.

Angelopoulos, Nicos and James Cussens (2005a). "Exploiting Informative Priors for Bayesian Classification and Regression Trees." In: *IJCAI*. Citeseer, pp. 641–646.

— (2005b). "Tempering for Bayesian C&RT". In: *Proceedings of the 22nd international conference on Machine learning*, pp. 17–24.

Atchadé, Yves F, Gareth O Roberts, and Jeffrey S Rosenthal (2011). "Towards optimal scaling of Metropolis-coupled Markov chain Monte Carlo". In: *Statistics and Computing* 21.4, pp. 555–568.

Berk, Richard (Jan. 2008). *Statistical Learning From a Regression Perspective*.

Breiman, Leo et al. (1984). *Classification and regression trees*. The Wadsworth statistics/probability series. CRC.

Brooks, Stephen P and Gareth O Roberts (1998). "Convergence assessment techniques for Markov chain Monte Carlo". In: *Statistics and Computing* 8.4, pp. 319–335.

Buntine, Wray Lindsay (1992a). "Learning classification trees". In: *Statistics and computing* 2.2, pp. 63–73.

— (1992b). "A theory of learning classification rules". PhD thesis. Citeseer.

Chipman, Hugh A, Edward I George, and Robert E McCulloch (1998). "Bayesian CART model search". In: *Journal of the American Statistical Association* 93.443, pp. 935–948.

— (2002). "Bayesian treed models". In: *Machine Learning* 48.1, pp. 299–320.

Chipman, Hugh A, Edward I George, Robert E McCulloch, et al. (2010). "BART: Bayesian additive regression trees". In: *The Annals of Applied Statistics* 4.1, pp. 266–298.

Chipman, Hugh A and Robert E McCulloch (2000). "Hierarchical priors for Bayesian CART shrinkage". In: *Statistics and Computing* 10.1, pp. 17–24.

Cowles, Mary Kathryn and Bradley P Carlin (1996). "Markov chain Monte Carlo convergence diagnostics: a comparative review". In: *Journal of the American Statistical Association* 91.434, pp. 883–904.

Denison, David GT, Christopher C Holmes, et al. (2002). *Bayesian methods for nonlinear classification and regression*. Vol. 386. John Wiley & Sons.

Denison, David GT, Bani K Mallick, and Adrian FM Smith (1998). "A bayesian cart algorithm". In: *Biometrika* 85.2, pp. 363–377.

Geyer, Charles J (1991). "Markov Chain Monte Carlo Maximum Likelihood". In: *Computing Science and Statistics: Proceedings of the 23rd Symposiumc on the interface.*

— (2011). "Importance sampling, simulated tempering, and umbrella sampling". In: *Handbook of Markov Chain Monte Carlo*, pp. 295–311.

Geyer, Charles J and Elizabeth A Thompson (1995). "Annealing Markov chain Monte Carlo with applications to ancestral inference". In: *Journal of the American Statistical Association* 90.431, pp. 909–920.

Gilks, Walter R and Gareth O Roberts (1996). "Strategies for improving MCMC". In: *Markov chain Monte Carlo in practice* 6, pp. 89–114.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman (2001). *The Elements of Statistical Learning*. 2nd ed. Springer Series in Statistics. Springer New York Inc.

Hastings, W Keith (1970). "Monte Carlo sampling methods using Markov chains and their applications". In:

Kapelner, Adam and Justin Bleich (2013). "Bartmachine: A powerful tool for machine learning". In: *stat* 1050, p. 8.

Kirkpatrick, Scott, C Daniel Gelatt, and Mario P Vecchi (1983). "Optimization by simulated annealing". In: *science* 220.4598, pp. 671–680.

Kone, Aminata and David A Kofke (2005). "Selection of temperature intervals for parallel-tempering simulations". In: *The Journal of chemical physics* 122.20, p. 206101.

Lingenheil, Martin et al. (2009). "Efficiency of exchange schemes in replica exchange". In: *Chemical Physics Letters* 478.1-3, pp. 80–84.

Marinari, Enzo and Giorgio Parisi (1992). "Simulated tempering: a new Monte Carlo scheme". In: *EPL (Europhysics Letters)* 19.6, p. 451.

Metropolis, Nicholas et al. (1953). "Equation of state calculations by fast computing machines". In: *The journal of chemical physics* 21.6, pp. 1087–1092.

Miasojedow, Błażej, Eric Moulines, and Matti Vihola (2013). "An adaptive parallel tempering algorithm". In: *Journal of Computational and Graphical Statistics* 22.3, pp. 649–664.

Muggleton, Stephen et al. (1996). "Stochastic logic programs". In: *Advances in inductive logic programming* 32, pp. 254–264.

Pratola, Matthew T et al. (2016). "Efficient Metropolis–Hastings proposal mechanisms for Bayesian regression tree models". In: *Bayesian analysis* 11.3, pp. 885–911.

Syed, Saifuddin et al. (2019). "Non-reversible parallel tempering: a scalable highly parallel MCMC scheme". In: *arXiv preprint arXiv:1905.02939*.

Tierney, Luke (1994). "Markov chains for exploring posterior distributions". In: *the Annals of Statistics*, pp. 1701–1728.

Torrie, Glenn M and John P Valleau (1977). "Nonphysical sampling distributions in Monte Carlo free-energy estimation: Umbrella sampling". In: *Journal of Computational Physics* 23.2, pp. 187–199.

Wu, Yuhong, Håkon Tjelmeland, and Mike West (2007). "Bayesian CART: Prior Specification and Posterior Simulation". In: *Journal of Computational and Graphical Statistics* 16.1, pp. 44–66. DOI: 10.1198/106186007X180426.

Yang, Ziheng (2014). *Molecular evolution: a statistical approach*. Oxford University Press.