BAYESIAN TREE MODELS

by

Yuhong Wu

Institute of Statistics and Decision Sciences
Duke University

Date: _____

Approved:

_____
Dr. Mike West, Supervisor

_____
Dr. Feng Liang

_____
Dr. Sayan Mukherjee

_____
Dr. Håkon Tjelmeland

Dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in the Institute of Statistics and Decision Sciences
in the Graduate School of
Duke University

2006

ABSTRACT

(Statistics)

BAYESIAN TREE MODELS

by

Yuhong Wu

Institute of Statistics and Decision Sciences
Duke University

Date: _____

Approved:

_____
Dr. Mike West, Supervisor

_____
Dr. Feng Liang

_____
Dr. Sayan Mukherjee

_____
Dr. Håkon Tjelmeland

An abstract of a dissertation submitted in partial
fulfillment of the requirements for the degree
of Doctor of Philosophy in the
Institute of Statistics and Decision Sciences in the Graduate School of
Duke University

2006

Abstract

This dissertation presents the statistical framework of Bayesian analysis of tree models with various applications. Prior specification for such models and development of algorithms for sampling from the posterior distributions are both challenging problems. This thesis addresses each of these issues and extends the Bayesian tree model in several ways, including data resampling (Dirichlet process prior), random threshold in the splitting rules, and autoregressive processes for modeling nonlinear structure in time series data. We also demonstrate various numerical techniques to reduce computational burden.

This thesis is divided into two parts. The first part, including the first three chapters, mainly describes the general framework of our Bayesian tree model. Chapter 1 introduces the formal definition of binary tree models. Several key aspects, including tree structure, splitting rules and leaf node distributions, are discussed, setting the foundation for the discussion of prior specification and posterior exploration.

Chapter 2 discusses the prior specification for tree models in detail. A *Pinball* prior for the tree generating process is defined; this allows for the combination of an explicit specification of a distribution for both the tree *size* and the tree *shape*. Both the data-dependent and data-independent prior for splitting rules are discussed. Comparisons are made with existing prior specifications.

Chapter 3 develops an efficient method for simulation from the posterior tree model space. The core computational innovations involve a novel Metropolis–Hastings method that can dramatically improve the convergence and mixing properties of MCMC methods for Bayesian tree analysis. Existing MCMC methods

simulate Bayesian tree models using very local MCMC moves, proposing only small changes. Our new Metropolis–Hastings move makes large changes in the tree, but is at the same time local in that it leaves unchanged the partition of observations into leaf nodes.

The second part of this thesis gives several examples. Chapter 4 presents a synthetic data example, illustrating basic proposals and restructure proposal in detail. By exploring this simple example, we illustrate the convergence of our MCMC method. Chapter 5 provides a more complicated example. We present exploratory tools to diagnose the convergence problem, make comparisons with existing MCMC methods, and introduce an importance sampling method to reduce the computational burden in assessing prediction validity.

The remaining chapters present extensions of Bayesian tree models. Chapter 6 presents a resampling method inspired by a study in proteomics. A Dirichlet process prior for resampling is introduced and discussed. Chapter 7 introduces the idea and methods of random thresholds in tree models, leading to a novel class of "smooth threshold" trees. Chapter 8 develops an autoregressive tree model, aiming to model nonlinear structure in time series data, with some illuminating examples.

Appendix A describes the implementation of Bayesian tree models and demonstrates the usage of C++ code, developed as part of this research.

Contents

List of Tables

List of Figures

Acknowledgements

I would like to acknowledge with gratitude my friends and colleagues at ISDS who have helped and supported me over the past few years. My primary thanks go to my advisor, Prof. Mike West, for his constant support and motivating encouragement during the development of this research and throughout my graduate school experience. It has been a privilege and a honor to learn about statistics and science by working with him. His enthusiasm for and commitment to research work set a strong example that I hope to follow.

I would like to thank Håkon Tjelmeland for his important role as a collaborator in the early stage of this research. I am grateful for the careful revision of the thesis carried out by the committee members: Sayan Mukherjee, Feng Liang and Håkon Tjelmeland and the proofreading through the thesis by Satkartar Kinney.

Thanks to my fabulous fiancee (wife soon) Fei for her endless support, love and enthusiasm. This dissertation would not have been possible without her encouragement.

Special thanks to my family. Your every word from the other side of the globe has been my greatest support.

Chapter 1

CART Model Structure and Notation

Tree models partition the predictor space into sub-regions in which the distribution of the response variable is more homogeneous. When the linear assumption does not hold, the expected mean level of the response may be a highly non-linear function of the covariates. Tree models aim to reduce the complexity by approximating that function locally in each sub-region. In each sub-region, a simple distribution (Normal, Bernoulli, etc.) is assumed for the observations belonging to this region.

As discussed in Breiman *et al.* (1984), we do not claim that tree models are always better than other regression methods. But tree models do give an interesting and illuminating way to look at the data. In this chapter, we will first study a breast cancer dataset with generalized linear models. The linear relationship between the logit of expected mean of the response and the covariate is questionable. From there, we introduce the idea of tree models. The classical method of tree finding and its weakness are discussed.

The formal definition of a binary tree is introduced along with the notation for the numbering, subtree, and recursive partitioning. Tree modeling is a very

flexible method, partly due to various distribution specifications in the leaf node. Different leaf node distributions are reviewed in this chapter, including normal distribution (regression tree), binomial distribution (classification tree) and Weibull distribution (survival tree). We also introduce new tree models in later chapters, including tree models with random thresholds and the Autoregressive tree models.

1.1 Introduction

1.1.1 Illustrative example

In this section, we attempt to analyze the breast cancer dataset used in Chipman *et al.* (1998) using a generalized linear model. The data will be revisited and studied in detail using our Bayesian tree model in Chapter 5. The data set has nine predictors and one binary response, indicating benign (0) and malign (1). A generalized linear model with logit as the link function is used:

$$\text{logit}(\text{E}(Y_i)) \quad = \quad \beta_0 + \sum_{j=1}^{9} \beta_j X_{ij} \tag{1.1}$$

for $i = 1, 2, \ldots, n$. We run the function glm in R to obtain the estimates for the parameters. Results are displayed in Table 1.1. At the 0.1 significance level, this model indicates that the predictors x_2, x_3, x_5 and x_9 are not significant; however, this may not be true. If we take a look at the observations with $x_2 \leq 0.25$ and $x_2 > 0.25$, we find that the responses in the first set are mostly 0 (12 out of 418 are 1) and the responses the second set are mostly 1 (38 out of 265 are 0). This indicates x_2 has great predictive power in distinguishing between the benign and malign state. So the assumption that $\text{logit}(\text{E}(y))$ is a linear function of $x_1, x_2, \ldots, x_9$ may be questionable.

```
Call:
glm(formula = Y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9,
              family = binomial)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -10.1039     1.1749  -8.600  < 2e-16 ***
x1            5.3501     1.4202   3.767 0.000165 ***
x2           -0.0628     2.0908  -0.030 0.976039
x3            3.2271     2.3060   1.399 0.161688
x4            3.3064     1.2345   2.678 0.007400 **
x5            0.9664     1.5659   0.617 0.537159
x6            3.8302     0.9384   4.082 4.47e-05 ***
x7            4.4719     1.7138   2.609 0.009073 **
x8            2.1303     1.1287   1.887 0.059115 .
x9            5.3484     3.2877   1.627 0.103788
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 884.35  on 682  degrees of freedom
Residual deviance: 102.89  on 673  degrees of freedom
AIC: 122.89

Number of Fisher Scoring iterations: 8
```

Table 1.1: Summary of the GLM parameter estimates from the breast cancer dataset.

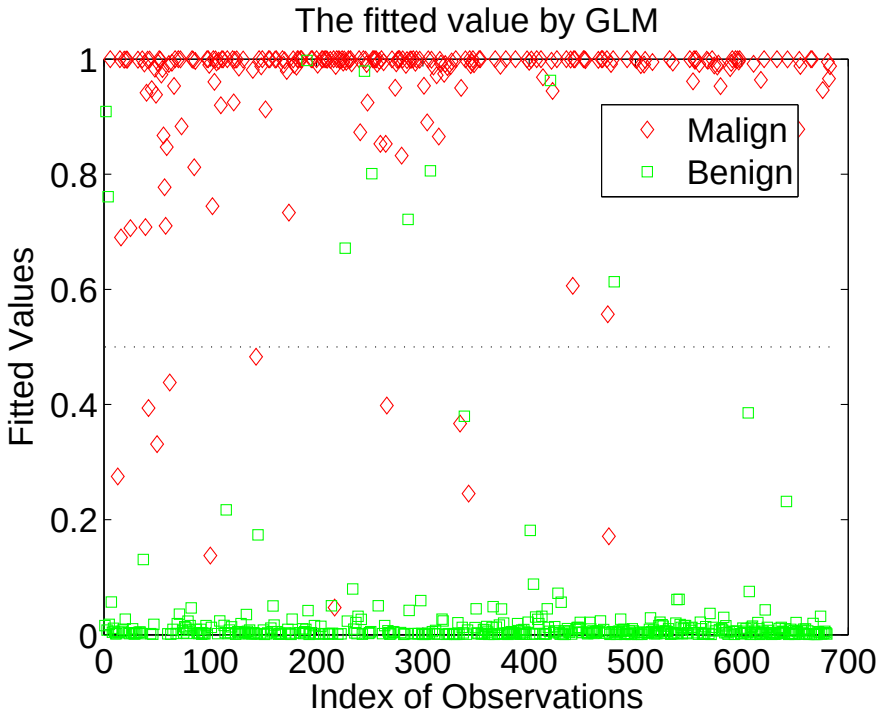


Figure 1.1: The fitted values of y by GLM. The red diamond points and the green square points indicate malign and benign observations respectively.

Further, Figure 1.1 displays the fitted values of y using GLM. The malign and benign observations are marked with red diamonds and green squares respectively. There are 11 red diamonds below the 0.5 threshold and 10 green squares above the 0.5 threshold. Therefore we have $21/683 = 0.031$ misclassified observations. Can we do better by relaxing the linear assumption? We will give the answer in Chapter 5 after our Bayesian tree model is introduced.

Generally, when the linear model is not appropriate as in this breast cancer example, we may want to use a non-linear model:

$$\mathrm{E}(y) \quad = \quad f(x_1, x_2, \ldots, x_p) \tag{1.2}$$

where $f(\cdot)$ is non-linear function. There are different choices for f, such as non-parametric regression, neural network or tree models. A tree model recursively

partitions the predictor space into sub-regions in which the distribution of y is more homogeneous. The partition is defined by splitting rules. We choose to study tree models for the following reasons:

- Logical statement and interpretable rules. A tree model is very easy to understand. One only need to browse from the top node to the bottom. At each visited node, answer the question (splitting rule) and then go to the directed child node. The meaning of each splitting rule is therefore straight forward. For example, one may encounter questions such as, "Is the cell size greater than 0.24?", and, "Is the marginal adhesion greater than 0.3?", in order to determine if the tumor is malign or benign.

- Clear indication of important predictors. By looking at the results from tree model analysis, it is easy to determine the important predictors. For example, a predictor that is used in the splitting rule at the top node may be more informative than the other predictors. A predictor that is used more than once may also be very important.

- Correlated structure between predictors. A correlation structure that is hierarchically defined can be well visualized by tree model.

1.1.2 Greedy algorithm

In classical methods, a cost function is first defined for the tree in order to do model selection. Some examples of the cost function include the residual sum of squares for regression trees and deviance for classification trees. Then a set of trees are evaluated and the one with the lowest cost is selected. Each tree corresponds to a partition of the predictor space, so in the classic method one starts with the

whole predictor space, which corresponds to the single node tree, and then uses the recursive partitioning algorithm to find a good tree.

Suppose now we want to find a regression tree for a dataset with n observations and each observation has p predictors. The partitioning algorithm is as follows:

1. Consider all partitions that split one region into two regions where the division is made at one of the predictors.
2. Compute the residual sum of square (RSS) for the new partitions.
3. Choose the partition that minimize the RSS.
4. Sub-partition the partitions recursively until the decrease in RSS is no more than some pre-specified value ϵ.

At each step, the partition that minimizes the cost function is chosen; however, this choice is not necessarily the optimal solution. Hence this procedure is called the "greedy" algorithm.

While the choice of ϵ is not clear, this algorithm may stop too quickly as discussed by Faraway (2005). An example is given in Figure 1.2. The red star and the blue circle points are draws from $\mathrm{N}(\mu_1, 1)$ and $\mathrm{N}(\mu_2, 1)$ respectively. It is easy to show that if the difference between μ_1 and μ_2 is much greater than 1 (variance), the decrease in the cost function by any split on just one predictor is very small. A better fit can be achieved only by making the splits on both x_1 and x_2. Therefore, the greedy algorithm may stop immediately with one single node and fail to find the partition that is defined by the lines $x_1 = 0$ and $x_2 = 0$, which is the optimal solution.

One could enumerate and evaluate all the possible trees in order to find the optimal solution; however, this is feasible only in the case of very small trees. Let

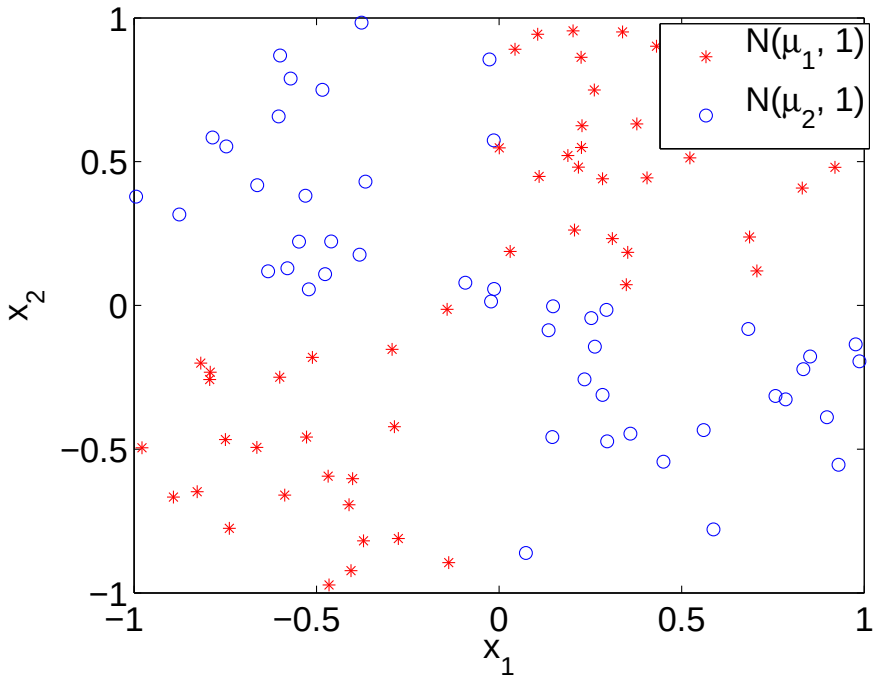


Figure 1.2: Extreme example data. There are only four data points. The red stars have the response value of 1 while the blue circle have the response value of 2.

$\mathcal{C}(i)$ denote the total number of possible trees, ignoring the difference in splitting rules, with i terminal nodes (the size of tree). Table 1.2 shows the values of $\mathcal{C}(i)$ for different values of i. The formal definitions of splitting rules, size of tree and the tree will be introduced in Section 1.2. We can see that the number of trees given the size grows very quickly. Actually the number of trees is an increasing function in the tree size with, if not faster than, an exponential rate of growth. First, we notice that $\mathcal{C}(n) = \sum_{i=1}^{n-1} \mathcal{C}(i) \cdot \mathcal{C}(n-i)$. Then we have:

$$\begin{aligned} \mathcal{C}(n) &= \sum_{i=1}^{n-1} \mathcal{C}(i) \cdot \mathcal{C}(n-i) \\ &\geq \mathcal{C}(1) \cdot \mathcal{C}(n-1) + \mathcal{C}(n-1) \cdot \mathcal{C}(1) \\ &= 2\mathcal{C}(n-1) \end{aligned} \tag{1.3}$$

for every $n > 1$. Noting that $\mathcal{C}(2) = \mathcal{C}(1) = 1$, we have $\mathcal{C}(n) \geq 2^{n-1}$ for every

$n > 1$. Furthermore, for each tree there are many choices of splitting rules for each node, so it is infeasible to enumerate all the trees.

i	1	2	3	4	5	6
$\mathcal{C}(i)$	1	1	2	5	14	42

Table 1.2: The number of trees for different sizes of the trees.

1.1.3 Cost-complexity for trees

Most cost functions, e.g. residual sum of squares for regression tree, are decreasing functions of the size of the tree. There will be overfitting is there if no restriction on the size of tree. Cross validation can be used to assess the prediction validity; however, there may be too many trees to choose from and cross validation would be too expensive. Pruning can be used to reduce the set of trees to be considered.

Usually a cost-complexity function for the tree is defined as:

$$\text{CC}(Tree) = \text{fit of the tree} + \lambda \cdot \text{size of the tree}. \tag{1.4}$$

Large λ encourages small trees and vice versa. In the greedy algorithm, a large tree will be grown and then pruned back according to the cost-complexity function.

In the Bayesian tree model, which will be formally defined and discussed in later sections, an analogy to the cost-complexity function is the negative log posterior probability, which is:

$$-\log(\text{Posterior probability}) = \text{constant} - \log(\text{Likelihood}) - \log(\text{Prior}). \tag{1.5}$$

The log likelihood function is similar to the fit at the nodes. The prior for the tree includes the prior for the size and the shape of the tree. Usually we specify a prior for the tree so that a simple tree model is encouraged, i.e. extremely large

(complicated) trees should have relatively low prior probabilities. This is more flexible by choosing appropriate prior. Not only the size but also the shape of the tree are under control according to our prior preference.

1.2 Binary tree: notations, node numbering and subtree

In this section, we introduce the notation that will be used to describe the tree models and the numbering of nodes. The example tree in Figure 1.3 illustrates the unique numbering of nodes, from the root (0) to the set of terminal nodes (or leaves). Each internal node (non-terminal) u in a tree has two children, a left child and a right child. We let $l(u)$ and $r(u)$ denote the left and right child of node u, respectively. Then

$$l(u) = 2u + 1 \text{ and } r(u) = 2u + 2. \tag{1.6}$$

Except for the root, $u = 0$, all nodes has exactly one parent node and one sister node. The parent node of u is

$$p(u) = \left\lceil \frac{u}{2} \right\rceil - 1 \text{ for } u \geq 1, \tag{1.7}$$

where $\lceil x \rceil$ is the smallest integer larger or equal to x. The sister node of $u \geq 1$ is

$$s(u) = \begin{cases} u+1 & \text{if } u \text{ is odd,} \\ u-1 & \text{if } u \text{ is even.} \end{cases} \tag{1.8}$$

Each node belongs to a certain level, or *floor*, in the tree: the root node is at level 0; each node u is in level

$$f(u) = \lfloor \log_2(u+1) \rfloor, \tag{1.9}$$

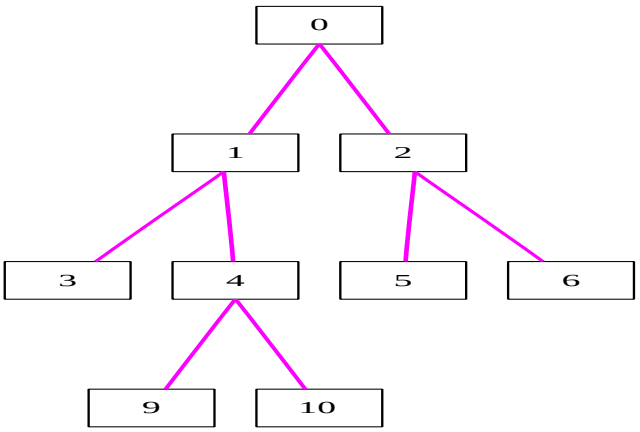


Figure 1.3: A (recursively defined, binary) tree with unique numbering of nodes

where $\lfloor x \rfloor$ is the largest integer smaller or equal to x.

Letting $\mathcal{N} = \{0, 1, 2, \ldots\}$, we can now give a formal definition of a finite binary tree. For a set A we let $|A|$ denote the number of elements in A.

Definition 1. *A set $T \subset \mathcal{N}$ is a (finite) binary tree if (i) $|T| < \infty$, (ii) $0 \in T$, and (iii) for any $u \in T \setminus \{0\}$ also $p(u), s(u) \in T$.*

We let $\mathcal{T}$ denote the set of all possible finite binary trees. The nodes in $T \in \mathcal{T}$ are naturally divided into two groups, the internal nodes and the terminal nodes or leaves. We denote the two subsets of nodes by $a(T)$ and $b(T)$, respectively, i.e.,

$$a(T) = \{u \in T | l(u) \in T\} \text{ and } b(T) = T \setminus a(T). \tag{1.10}$$

We let $m(T) = |b(T)|$ denote the number of leaves in T. Notice that $|T|$ is the number of elements in T, which is different from the number of leaves in T. In this thesis, the size of tree T refers to $m(T)$.

For any tree T, the subtree from node $u \in T$ is just the tree from (inclusively) u on downward to the leaves of T below u, which we denote by

$$S_u(T) = \{v | v + u2^{f(v)} \in T\}. \tag{1.11}$$

We sometimes write $m_u(T)$ for the number of leaves in the subtree from u, i.e. $m_u(T) = m(S_u(T))$. Evidently, $S_0(T) = T$ and $m_0(T) = m(T)$.

For the example tree shown in Figure 1.3, we have $T = \{0, 1, 2, 3, 4, 5, 6, 9, 10\}$, $a(T) = \{0, 1, 2, 4\}$, and $b(T) = \{3, 5, 6, 9, 10\}$. The subtree of node 1 is $S_1(T) = \{0, 1, 2, 5, 6\}$.

1.3 Splitting rules

We let $\mathcal{Y}$ denote the space over which the response variable is defined. We are interested in using a tree model to predict variable $y \in \mathcal{Y}$ based on a set of candidate covariates (or predictor variables) $\boldsymbol{x} = (x^1, \ldots, x^p)'$. With sample spaces $x^j \in \mathcal{X}_j$ we have $\boldsymbol{x} \in \mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_p$. A tree partitions $\mathcal{X}$ into regions by assigning a splitting rule to each internal node of a binary tree T. An internal node $u \in a(T)$ will be split as follows:

- Choose a predictor variable index $k_T(u) \in \mathcal{P} = \{1, \ldots, p\}$ and a splitting threshold for that variable $\tau_T(u) \in \mathcal{X}_{k_T(u)}$;
- Variables $(y, \boldsymbol{x})$ are assigned to the left child $l(u)$ of u if $x^{k_T(u)} \leq \tau_T(u)$, otherwise to the right child $r(u)$.

The left panel in Figure 1.4 displays an example tree with splitting rules in its internal nodes. Note that some of the predictor variables may be categorical. In this case, the splitting threshold (or more accurately, splitting set) becomes $\tau_T(u) \subset \chi_{k_T(u)}$ and the variable $(y, \boldsymbol{x})$ are assigned to the left child $l(u)$ of u if $x^{k_T(u)} \in \tau_T(u)$, otherwise to the right child $r(u)$. In the following text if the splitting variable is categorical, we still write the splitting rules as "$x^{k_T(u)} \leq \tau_T(u)$",

which actually means $x^{k_T(u)} \in \tau_T(u)$. There are only a finite number of splitting sets for a categorical splitting variable. In contrast, there are an infinite number of splitting thresholds for the continuous predictor variable, in which case we allow the splitting threshold, $\tau_{T(u)}$, to take any value, including unobserved values; in contrast, Chipman *et al.* (1998) restrict $\tau_{T(u)}$ at the observed values of $x^{k_T(u)}$. Such assignment of splitting rules will depend on $\mathcal{X}_{k_T(u)}$. This dependence and its potential advantages was discussed by Buntine (1992). Another explanation of data-dependent priors based on the Dirichlet process will be discussed in Section 2.2.2 when we specify the prior for the splitting threshold.

The splitting rules defined above are said to be *deterministic* because in each internal node, variables $(y, \boldsymbol{x})$ are assigned either to the left or to the right according to the specified values. An alternative to deterministic splitting rules are *random* splitting rules, where $(y, \boldsymbol{x})$ are assigned the child nodes randomly. This will be discussed in Chapter 7.

Writing $\mathsf{T} = (T, \boldsymbol{k}_T, \boldsymbol{\tau}_T)$, where $\boldsymbol{k}_T = \{k_T(u), u \in a(T)\}$ and $\boldsymbol{\tau}_T = \{\tau_T(u), u \in a(T)\}$, recursively induces regions $R_{\mathsf{T}}(u)$ to each node $u \in T$, where

$$R_{\mathsf{T}}(u) = \begin{cases} \mathcal{X} & \text{if } u = 0, \\ R_{\mathsf{T}}(p(u)) \cap \{x \in \mathcal{X} | x^{k_T(u)} \leq \tau_T(u)\} & \text{if } u \text{ odd}, \\ R_{\mathsf{T}}(p(u)) \cap \{x \in \mathcal{X} | x^{k_T(u)} > \tau_T(u)\} & \text{if } u > 0 \text{ and even}. \end{cases} \tag{1.12}$$

In particular, the regions in the leaves $\{R_{\mathsf{T}}(u), u \in b(T)\}$ are disjoint and form a partition of $\mathcal{X}$. The right panel in Figure 1.4 displays the regions in the leaves of the tree in the left panel.

Corresponding to the definition of subtrees $S_u(T)$ above, we let $S_u(\boldsymbol{k}_T)$ and $S_u(\boldsymbol{\tau}_T)$ denote the splitting variables and splitting thresholds in the subtree $S_u(T)$. More precisely, for $u \in T$ we have $S_u(\boldsymbol{k}_T) = \{k_{S_u(T)}(v) : k_{S_u(T)}(v) = k_T(v +$

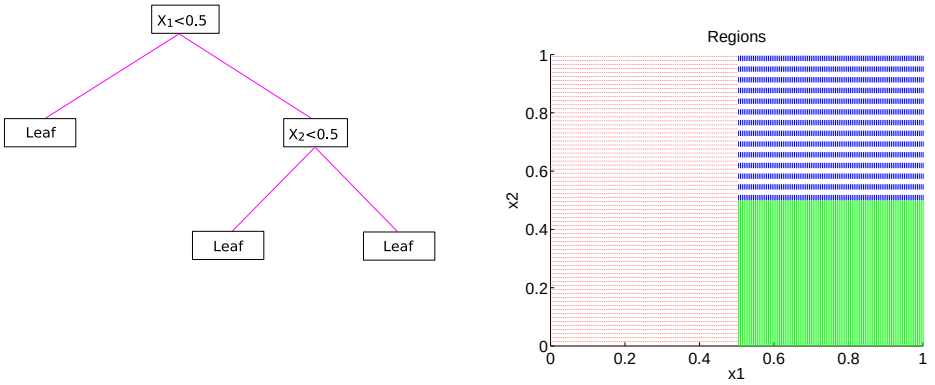


Figure 1.4: An example tree with splitting rules and the induced regions in its leaves.

$u2^{f(v)}), v \in a(S_u(T))\}$ and $S_u(\boldsymbol{\tau}_T) = \{\tau_{S_u(T)}(v) : \tau_{S_u(T)}(v) = \tau_T(v + u2^{f(v)}), v \in a(S_u(T))\}$. For $\mathsf{T} = (T, \boldsymbol{k}_T, \boldsymbol{\tau}_T)$ and $u \in T$ we will also use the notation $S_u(\mathsf{T}) = (S_u(T), S_u(\boldsymbol{k}_T), S_u(\boldsymbol{\tau}_T))$.

1.4 Recursive partitioning

Suppose we have observations $(y_i, \boldsymbol{x}_i)$, $i \in \mathcal{I}$, where $\mathcal{I}$ denotes the index set $\{1, \ldots, n\}$. The tree model specification (above) recursively partitions the data, assigning subsets of $\mathcal{I}$ to each node corresponding to the regions $R_{\mathsf{T}}(u)$ defined above. Starting with the full data set at the root, $\mathcal{N}_{\mathsf{T}}(0, \mathcal{I}) = \mathcal{I}$, the subset at any node u is $\mathcal{N}_{\mathsf{T}}(u, \mathcal{I}) = \{i \in \mathcal{I} | \boldsymbol{x}_i \in R_{\mathsf{T}}(u)\}$. The subsets in the leaves define a partition of the full data set, denoted by $\mathcal{L}_{\mathsf{T}}(\mathcal{I}) = \{\mathcal{N}_{\mathsf{T}}(u), u \in b(T)\}$. One may express the partition of $\mathcal{I}$ in terms of partitions induced by subtrees. As an example, for the tree in Figure 1.5 we list the partition of $\mathcal{I}$ in Table 1.3. This notation will be helpful for illustrating the changes in the leaves when describing the proposals in Chapter 3.

Subtree $S_u(\mathsf{T})$	Partition of the full data set $\mathcal{L}_{S_u(\mathsf{T})}(\mathcal{N}_{\mathsf{T}}(u, \mathcal{I}))$
T	$\{\{1,2,3\},\{4,5,6\},\{7,8,9\},\{10,11,12\}\}$
$S_1(\mathsf{T})$	$\{\{1,2,3\},\{4,5,6\}\}$
$S_2(\mathsf{T})$	$\{\{7,8,9\},\{10,11,12\}\}$
$S_3(\mathsf{T})$	$\{\{1,2,3\}\}$
$S_4(\mathsf{T})$	$\{\{4,5,6\}\}$

Table 1.3: The partition of $\mathcal{I}$ in terms of partitions induced by subtrees

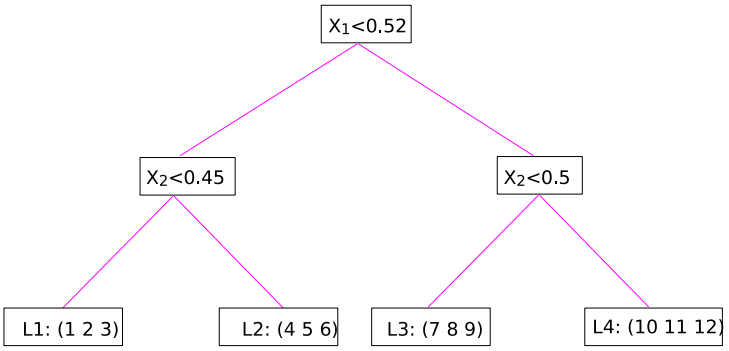


Figure 1.5: An example tree. The subsets in the leaves define a partition of the full data set

1.5 Statistical model in leaves

We focus on tree models in which the subset of y outcomes in any leaf u is viewed as a random sample from a distribution with density $\phi(\cdot|\theta_u)$ (Chipman *et al.*, 1998). Key examples that will be discussed in later chapters are normal and Bernoulli distributions for y.

Let $\{y\}_u$ denote all the outcomes that belong to node u, where u is a leaf node

in $b(T)$. In the case of Bernoulli distributions, we assume

$$y = \begin{cases} 1 & w.p. \quad p_u, \\ 0 & w.p. \quad 1 - p_u, \end{cases} \tag{1.13}$$

for every $y \in \{y\}_u$. In the case of normal distributions, we assume

$$y \sim N(\mu_u, \sigma_u^2) \tag{1.14}$$

for every $y \in \{y\}_u$. These two specifications indicate that the parameters defining the distributions differ across leaves. This allows us to specify the prior for the leaves parameters independently across leaves. A variation is to assume

$$y \sim N(\mu_u, \sigma^2) \tag{1.15}$$

for every $y \in \{y\}_u$. This differs from equation (1.14) that we assume a single variance parameter for all the possible leaves in the tree. This specification is important when we consider the *Autoregressive Tree* model, which will be discussed in Chapter 8. Aside from this application, we restrict attention to random sample models within leaves in the remaining chapters in this thesis, although the new tree priors and MCMC innovations are applicable more generally.

Another key example is the Weibull distribution for y. In this case, we assume

$$y \sim \text{Weibull}(\alpha_u, \beta_u) \tag{1.16}$$

for every $y \in \{y\}_u$, where α_u and β_u are the shape parameter and scale parameter respectively. The density of the Weibull distribution is defined as $f(y) = \alpha\beta y^{\alpha-1} e^{-\beta y^\alpha}$. When the survival data is right-censored, there are two types of data. If y_i is censored ($c_i = 1$), it means at time y_i the corresponding subject is still alive. If y_i is observed ($c_i = 0$), y_i is the death time of the corresponding

subject. Define a function $S(y_i) = P(Y > y_i|\alpha, \beta) = \int_{y_i}^{\infty} f(y)dy = e^{-\beta y_i^{\alpha}}$. Thus the likelihood function is given by

$$\begin{aligned} & L(\alpha_u, \beta_u; \{y, c\}_u,) \\ = & \prod_{j \in \mathcal{N}_{\mathsf{T}}(u,\mathcal{I})} (f(y_j)(1 - c_j) + S(y_i)c_j) \\ = & \alpha^{n-n_c}\beta^{n-n_c} \prod_{j \in \mathcal{N}_{\mathsf{T}}(u,\mathcal{I}) \& c_j = 0} y_j^{\alpha-1} e^{-\beta \sum_{j \in \mathcal{N}_{\mathsf{T}}(u,\mathcal{I})} y_j^{\alpha}} \end{aligned} \tag{1.17}$$

where $n_c = \sum_{j \in \mathcal{N}_{\mathsf{T}}(u,\mathcal{I})} c_j$ is the number of censored observations in leaf node u and $n = |\mathcal{N}_{\mathsf{T}}(u, \mathcal{I})|$ is the total number of observations in leaf node u, including both censored and observed data.

In the above models, the terminal node distribution of Y does not depend on $\boldsymbol{x}$, which is mainly for constructing the partition. One further extension is to define the conditional distribution of Y given $\boldsymbol{x}$ in each terminal node. Such specification enriches the model structure in the terminal node. For example, as an extension to equation (1.14) we can assume

$$y|\boldsymbol{x} \sim N(\boldsymbol{x}\beta_u, \sigma_u^2) \tag{1.18}$$

for every $\{y, \boldsymbol{x}\} \in \{y, \boldsymbol{x}\}_u$, where $\{y, \boldsymbol{x}\}_u$ denotes all the observations belonging to terminal node u. Such treed models are discussed in Chipman *et al.* (2002); they can be used to describe a wider range of distributions. A special example, Autoregressive tree models, will be discussed in Chapter 8, where y is time series data and in each leaf node we regress the current value of y on the past values of y.

Chapter 2

Prior Model

For a complicated tree model $\mathsf{T} = \{T, \boldsymbol{k}_T, \boldsymbol{\tau}_T\}$, as specified in Chapter 1, model selection is a challenge. Even considering only the tree structures without splitting rules in the internal nodes, the number of trees increases rapidly with the size of tree, as discussed in Section 1.1. Therefore it is difficult to enumerate and evaluate all the tree models so as to find the "best" model. For example, suppose the data has n observations and each observation has p predictors. Even if we restrict our interest to all the trees with size of 5, there are 14 possible trees to choose from. For each tree, we need to specify four splitting rules. In the limited case that all splitting thresholds are restricted to the observed values, there are $(n-2)p$ different splitting rules that can be used in each internal node, resulting in about $56(n-2)p$ trees to be evaluated. Furthermore, not every tree will be equivalently favored. A very large tree leads to overfitting, while a very small tree, in the extreme case a single node tree, does not have much predictive power. Therefore, instead of listing all possible trees, we specify a prior distribution for $\mathsf{T} = \{T, \boldsymbol{k}_T, \boldsymbol{\tau}_T\}$. Then we will explore the posterior distribution of the tree model given the data. The prior specification can reflect our prior belief on tree size, tree shape, the choice

of splitting rules, and the leaf node distribution.

A closed form prior specification is not very feasible, so we define a tree generating process described as follows:

1. Tree structure. Tree structure refers to the binary tree as defined in Section 1.2. This includes the size of the tree and the shape of the tree, which could be balanced or skewed. Other than that, tree structure alone does not define any partition.

2. Splitting rules. In step 1, tree structure defines internal nodes and terminal nodes. In each internal node, we need to specify a splitting rule. Then the partition of the predictor space is defined.

3. Leaf node distribution. Step 1 and step 2 define a partition of the predictor space. In each sub-region, a distribution is specified in association with the leaf.

In short, we define the "skeleton" of the tree first and then fill in splitting rules so as to define the partition. This is different from the tree generating process in Bayesian CART discussed in Chipman *et al.* (1998) and Denison *et al.* (1998). Chipman *et al.* (1998) starts with a single node tree and then at each leaf node makes a split with some probability. This is repeated until no split is made. In this case, the size of tree depends on the choice of the splitting probability. Furthermore, a more sophisticated splitting probability, depending on the depth (level) of the node, is introduced for controlling the tree shape. However, how the tree shape, balanced or skewed, depends on that splitting probability is obscure, being defined implicitly by the construction rather than explicitly. Denison *et al.* (1998) claims that the tree shape (topology) is unimportant. Trees with the same size

are considered to be from the same "class". This specification does not consider the hierarchically dependent structure of splitting rules when splitting rules are filled in the tree. In our model, the prior distribution is specified hierarchically so that the prior for each component in the tree model is clear. Furthermore, as we will see in Chapter 3, this prior specification makes it easier to design proposals in the Metropolis-Hastings algorithm.

2.1 Pinball prior for tree generation

The tree structure is determined by two factors: the size of the tree (the number of terminal nodes) and the shape of the tree (the structure of nodes). Denison *et al.* (1998) specified a prior on the number of leaves, and then a uniform prior over trees with that number of leaves because trees with the same size are considered to come from the same "class". Note that this gives high prior probability to unbalanced trees. Chipman *et al.* (1998) defined a tree-generating process where the prior on the number of nodes and the shape of the tree is implicit, but make it relatively difficult to incorporate a prior on the number of leaves. We would like a prior distribution that can well control the size and the shape of the tree. The novel *pinball prior* here overcomes this and extends both earlier approaches. It is implemented by first specifying the size of the tree and then, given the size, specifying the tree layout (balanced or skewed).

The construction is simple and intuitive: the prior generates a number of terminal leaves $m(T)$, and then these leaves are cascaded down from the root node, randomly splitting into left/right nodes at any node, according to a defined probability distribution at that node until they define individual leaves - the leaves fall down the tree as pinballs. This process is formally described as follows:

- Specify a prior density for tree size (number of terminal nodes) $m(T) \sim \alpha(m(T))$ with support on a subset of $\{1, 2, \cdots\}$.

- Noting $m_0(T) = m(T)$, recall that $m_u(T)$ is the number of leaves in the subtree $S_u(T)$ below node u. At this node, these leaves are distributed to either the left or the right according to some prior. Take $\beta(m_{l(u)}(T)|m_u(T))$ as a prior density governing the generation of this split, where $m_{l(u)}(T)$ is the number of leaves, of the current total $m_u(T)$, sent to the left child node $l(u)$, the remainder going to $r(u)$. Note that the support of $\beta(m_{l(u)}(T)|m_u(T))$ is on $\{1, 2, \cdots, m_u(T) - 1\}$ so that both the left child node and the right child node has at least one leaf. Therefore the process ends when $m_u(T) = 1$ at node u.

Then a full specification for T is given by

$$\pi(T) = \alpha(m_0(T)) \prod_{u \in a(T)} \beta(m_{l(u)}(T)|m_u(T)) \quad \text{for} \quad T \in \mathcal{T} \tag{2.1}$$

An obvious candidate for the prior distribution of tree size is a Poisson with restriction that tree size is not 0, say $m(T) = 1 + \text{Pois}(\lambda)$ for specified λ. Practically, λ is specified according to some prior knowledge on the problem and/or the size of data. An alternative candidate is a discrete uniform distribution on $\{1, 2, \cdots, N_{max}\}$ for some N_{max}. The size of tree cannot be larger than the size of data; therefore, we can specify a sufficient large N_{max}, e.g. the number of observations in the data.

Different choices are possible for function $\beta(i|m)$. Recall that $\mathcal{C}(i)$ denotes the number of possible tree structures with i leaves. Taking $\beta(i|m) = \frac{\mathcal{C}(i) \cdot \mathcal{C}(m-i)}{\mathcal{C}(m)}$ gives the uniform distribution over all trees of size m adopted in Denison *et al.*

i	1	2	3	4	5	6
$\beta(i\|7)$	0.34	0.09	0.07	0.07	0.09	0.34

Table 2.1: $\beta(i|7)$ for different i, where $\beta(\cdot|m)$ is the one gives uniform distribution over all trees of size m.

(1998) and in Chipman *et al.* (1998) as the one of the prior choices for splitting probability. Except for very small tree sizes, the number of unbalanced trees of a given size is much larger than the number of balanced trees of the same size, so this choice of $\beta(i|m)$ assigns a high prior probability to unbalanced trees. Table 2.1 shows the values of $\beta(i|m)$ for different values of i when m is 7. The probability of generating an unbalanced tree $\beta(1|7)$ is about five times the probability of generating a balanced tree $\beta(3|7)$. A natural alternative is a uniform distribution for $\beta(i|m)$, i.e. $\beta(i|m) = \frac{1}{m-1}$ for $i = 1, \cdots, m-1$, or $\beta(i|m) = \frac{1}{2}[\text{Bin}(i-1; m-2, p) + \text{Bin}(i-1; m-2, 1-p)]$, where $\text{Bin}(\cdot; n, p)$ denotes the binomial mass function with parameters n and p, where p is specified so as to represent the prior knowledge on the shape of the tree. With $p = 1/2$, the latter gives a balanced tree with high probability, whereas with p close to zero, the unbalanced trees get higher prior probabilities. When this prior is used, we call the prior for this tree generating process a pinball prior. Each leaf falling down as a pinball has 1/2 probability of going to the left child node independently. So the number of pinballs sent to the left child node, assuming there is at least one, follows a truncated binomial distribution. Table 2.2 shows the case when $p = 1/2$.

Simulation from this symmetric prior is direct from its definition: see examples in Figure 2.1. Note that asymmetric prior could be used to encourage trees that are more skewed in overall shape.

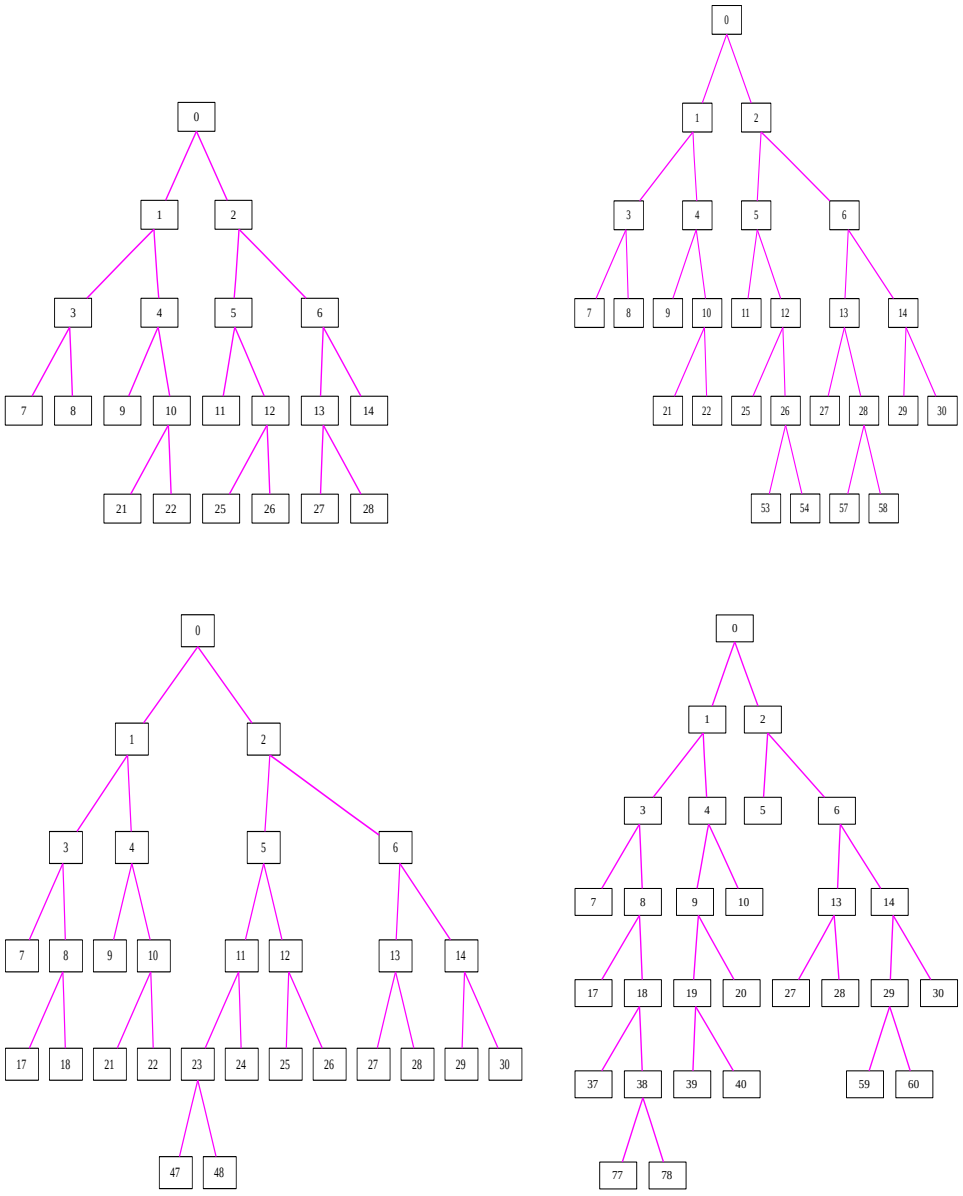


Figure 2.1: Four samples from the pinball prior with $\alpha(m) = 1 + \text{Pois}(m-1; 15)$ and $\beta(i; m) = 1 + \text{Bin}(i-1; m-1, 0.5)$.

i	1	2	3	4	5	6
$\beta(i\|7)$	0.03	0.16	0.31	0.31	0.16	0.03

Table 2.2: $\beta(i|7)$ for different i, where $\beta(\cdot|m)$ is used in pinball prior for $p = 0.5$.

2.2 Prior for splitting rules

In Section 2.1, we specified the prior for the tree structure. Given the tree structure, we need to specify splitting rules to each internal node so as to define a partition in the predictor space. For each internal node, we will first specify the prior for selection of the splitting variable, and then, given the selected splitting variable, we specify the prior for the splitting threshold.

2.2.1 Prior for selection of splitting variables

A simple approach to specify the prior for selection of splitting variables is to assume each $k_t(u)$ is independently generated from a fixed discrete distribution over the p predictor variables and then choose variable x^i with probability $\gamma(i)$, $i \in \mathcal{P}$. Thus

$$\pi(\boldsymbol{k}_T|T) \quad = \quad \prod_{u \in a(T)} \gamma(k_T(u)). \tag{2.2}$$

The independence between priors for each internal node can simplify the computation of the acceptance probability in Metropolis-Hastings algorithm. Furthermore, this specification can also help circumvent Reversible Jump MCMC (Green, 1995) in the case that the new splitting rules in the proposal are sampled from the prior, as discussed in Chapter 3. Note that the same predictor may be used as a splitting variable in several internal nodes.

A common choice of $\gamma(\cdot)$ is the probability mass function of the discrete uni-

form distribution. Alternatively, higher prior weight can be assigned to predictors that are thought to be more important. For this purpose, the correlation between predictors in the data can be used to specify $\gamma(\cdot)$. A more sophisticated way of specifying $\gamma(\cdot)$ is to redistribute the prior weight dynamically depending on the splitting variables in "ancestor nodes", assuming that the splitting rules are filled in top-down order. For example, higher prior weight can be assigned to predictors that have been already used in an ancestor node, which reflects the belief that the predictors in ancestor nodes have some explaining power and we would like to continue exploring their explanatory ability rather than randomly selecting one of the other unknown predictors. The ancestor nodes of any node u are all the nodes from which the subtree contains u. Let $A_u(T)$ denote such a set. Formally

$$A_u(T) \quad = \quad \{v | u \in S_v(T), v \in T\}. \tag{2.3}$$

For example, for the tree shown in Figure 1.3, $A_9(T) = \{0, 1, 4\}$ and $A_5(T) = \{0, 2\}$. Given the ancestor set, $\gamma(\cdot)$ for the current node u depends on $A_u(T)$. And therefore this probability is no longer independent of the others and is denoted as $\gamma_{A_u(T)}(k_T(u))$. Thus

$$\pi(\boldsymbol{k}_T|T) \quad = \quad \prod_{u \in a(T)} \gamma_{A_u(T)}(k_T(u)). \tag{2.4}$$

The dependence in prior specification for the selection of splitting variables can be used to give preference to some "important" predictors in choosing the splitting variable. This variable selection idea is discussed in Chipman *et al.* (1998), where they choose to put more mass on variables already used in the ancestor set. However, the use of i^{th} predictor should be restricted if it has been used for quite a few times so that the other predictors can be visited as well. The introduction of ancestor sets makes it possible to incorporate this restriction.

One obvious drawback of such prior specification is the complex computation. If a splitting rule in the top level node (close to the root node) of the tree is changed, all the prior probabilities associated with the nodes in the subtree need to be updated. Obviously, this increases the computation time significantly. Furthermore, this specification requires the use of the reversible jump MCMC.

2.2.2 Prior for splitting threshold

There are two different types of splitting variables: categorical and continuous. Different prior distributions are specified for these two cases.

When the chosen splitting variable ($k_T(u)$) in node u is a categorical variable, the splitting threshold $\tau_T(u)$ (more precisely, splitting set) is a subset of the space $\chi_{k_T(u)}$. As discussed in Section 1.3, the splitting rule is written as "$x^{k_T(u)} \leq \tau_T(u)$", indicating that $x^{k_T(u)} \in \tau_T(u)$. There are a total of $2^{|\chi_{k_T(u)}|} - 2$ possible splitting rules, where $|\chi_{k_T(u)}|$ denotes the number of possible outcomes of splitting variable $k_T(u)$. A natural choice for the prior distribution of such splitting rules is the discrete uniform distribution, that is, each possible subset is chosen with probability $1/(2^{|\chi_{k_T(u)}|} - 2)$.

When the chosen splitting variable ($k_T(u)$) is continuous, there are two ways to choose the values for the splitting threshold. In the first approach, some grid points, e.g. quantiles, within the range of the predictor are chosen as the possible splitting thresholds (Chipman *et al.*, 1998; Denison *et al.*, 1998). In the second approach, this restriction is relaxed by allowing the splitting threshold to take any value from the space $\chi_{k_T(u)}$.

If the size of a tree is greater than the number of observations, the posterior probability of such tree, as discussed in Chapter 3, is 0, so there are only a

finite number of trees (without considering splitting rules) with positive posterior probability. In the first approach when our interest is restricted to finite possible splitting rules, the number of trees in the state space to be explored is finite as well. Therefore, when new splitting rules are proposed in the Metropolis-Hastings algorithm, the proposal can be considered as a transition from one state to the other in a finite tree space. Then there is no dimensional change in the parameter and reversible jump MCMC can be avoided. While this prior specification relies on the data, however, it can be interpreted as an approximation to the distribution of $\tau_T(u)$ given the underlying unknown distribution of $x^{k_T(u)}$, $F_{k_T(u)}$.

Suppose we observe $X^{k_T(u)} = \{x_1^{k_T(u)}, x_2^{k_T(u)}, \ldots, x_r^{k_T(u)}\}$ for the splitting variable $k_T(u)$ in node u. These r values are random samples from an unknown distribution $F^{k_T(u)}(\cdot)$. We assume a Dirichlet process prior on $F^{k_T(u)}(\cdot)$

$$F \sim \mathcal{D}(\alpha F_0) \tag{2.5}$$

where $\mathcal{D}(\alpha F_0)$ denotes a Dirichlet process with parameter αF_0. Then the posterior distribution of $F^{k_T(u)}$ given $X^{k_T(u)}$ is also a Dirichlet process

$$p(F^{k_T(u)}|X^{k_T(u)}) \propto \mathcal{D}(\alpha F_0 + \sum_{j=1}^{r} \delta_{x_j^{k_T(u)}}). \tag{2.6}$$

Therefore a data-dependent prior for the splitting threshold $\tau_T(u)$ can be explained as

$$\begin{aligned} p(\tau_T(u)|X^{k_T(u)}) &= \int p(\tau_T(u)|F^{k_T(u)}) dp(F^{k_T(u)}|X^{k_T(u)}) \\ &\approx p(\tau_T(u)|\hat{F}^{k_T(u)}) \end{aligned} \tag{2.7}$$

where $\hat{F}_i$ is $(\alpha F_0 + \sum_{j=1}^{r} \delta_{x_j^{k_T(u)}})/(\alpha + r)$, just the empirical CDF for $X^{k_T(u)}$ as α goes to 0. Therefore the data-dependent prior is an approximation to $p(\tau_T(u)|\hat{F}^{k_T(u)})$

and $p(\tau_T(u)|\hat{F}^{k_T(u)})$ is used when it is hard to find a data-independent prior $p(\tau_T(u)|F^{k_T(u)})$.

In the case that the splitting threshold can take any value from $\chi^{k_T(u)}$ given that the splitting variable $k_T(u)$ is chosen for node u, the support of the prior distribution for this splitting threshold must contain $\chi^{k_T(u)}$. We denote its density function as $\delta_{k_T(u)}(\tau)$. Usually, $\chi^{k_T(u)}$ varies greatly with $k_T(u)$, so the data are pre-processed so that there exists a χ such that $\chi^{k_T(u)} \subseteq \chi$ and the prior distribution for the splitting threshold is defined on χ. For this purpose, there are two natural choices. The first one is

$$\tilde{x}^i = \frac{x^i - b^i}{a^i - b^i} \tag{2.8}$$

for $i = 1, 2, \ldots, p$, where $a^i, b^i \in \chi^i$ such that $\tilde{x}^i$ is within $[0, 1]$. Given the data, we can simply take $a^i = \max(X^i)$ and $b^i = \min(X^i)$. In this case a uniform distribution on $[0, 1]$ is specified as the prior for the splitting threshold.

Another way to pre-process the data is

$$\tilde{x}^i = \frac{x^i - \bar{X}^i}{\hat{\text{sd}}(X^i)} \tag{2.9}$$

for $i = 1, 2, \ldots, p$, where $\bar{X}^i$ and $\hat{\text{sd}}(X^i)$ are the mean and the standard deviation estimated from the data. A normal distribution with mean 0 and standard deviation 1 is a natural choice for the prior.

In our tree model there are two parameters $k_T(u)$ and $\tau_T(u)$ associated with each internal node. So adding or deleting nodes in the tree will lead to a change in the number of splitting rules and hence the dimension of parameters. Green (1995) designed a reversible jump MCMC algorithm for such situation. We will discuss this problem in detail in Section 3.2.

Usually we take $\tau_T(u)$ to be conditionally independent. Thus the prior for splitting thresholds given the selection of splitting variables is

$$\pi(\boldsymbol{\tau}_T|T, \boldsymbol{k}_T) = \prod_{u \in a(T)} \delta_{k_T(u)}(\tau_T(u)). \tag{2.10}$$

As in the selection of splitting variables, this conditional independence simplifies the computation of the acceptance probability in the Metropolis-Hastings algorithm. Similarly, we can also specify the prior distribution for splitting threshold conditional on the ancestor nodes. In some cases this can reduce the computational cost. For example, if a categorical predictor is already used in one of the ancestor nodes, then the size of the possible splitting rules will be reduced when we consider the splitting rules with this predictor. This is because some splitting rules will lead to an empty node. Thus we do not need to consider these "invalid" splitting rules and a discrete uniform distribution over all the "valid" splitting sets is used instead. Conditioning on the ancestor nodes, the size of these "valid" splitting rules is usually small.

2.3 Prior for leaf parameters

Denote the observed response data as $\boldsymbol{y} = \{y_i\}_{i=1}^n$. We note that, given T and $\boldsymbol{\theta}$, the likelihood function depends on T only through the induced partition of observations to the leaves, $\mathcal{L}_{\mathsf{T}}(\mathcal{I})$. If we assume that the leaf parameters are independent of each other across leaves, e.g. equations (1.13), (1.14), (1.16) and (1.18), then the data within leaves are independent random samples.

$$f(\boldsymbol{y}|\boldsymbol{\theta}_T, \mathsf{T}) = \prod_{u \in b(T)} f(\{y\}_u|\boldsymbol{\theta}_u, \mathsf{T}) \tag{2.11}$$

where $\{y\}_u$ denotes all the outcomes that belong to node u and $\boldsymbol{\theta}_u$ is the corresponding parameters. If the observations across leaves share some common parameter as in equation (1.15), we have

$$f(\boldsymbol{y}|\boldsymbol{\theta}_T, \mathsf{T}) = \prod_{u \in b(T)} f(\{y\}_u|\boldsymbol{\theta}_u, \boldsymbol{\theta}, \mathsf{T}) \tag{2.12}$$

where $\boldsymbol{\theta}$ is the common parameter shared by the leaves. Our focus is to explore the space of the tree model, more specifically the induced partition of observations. In other words, we are interested in $\pi(\mathsf{T}|\boldsymbol{y})$, which is

$$\begin{aligned} \pi(\mathsf{T}|\boldsymbol{y}) &\propto \int f(\boldsymbol{y}|\mathsf{T}, \boldsymbol{\theta}_T)\pi(\boldsymbol{\theta}_T|\mathsf{T})\pi(\mathsf{T})d\boldsymbol{\theta}_T \\ &= \pi(\mathsf{T}) \int f(\boldsymbol{y}|\mathsf{T}, \boldsymbol{\theta}_T)\pi(\boldsymbol{\theta}_T|\mathsf{T})d\boldsymbol{\theta}_T \\ &= \pi(\mathsf{T})f(\boldsymbol{y}|\mathsf{T}). \end{aligned} \tag{2.13}$$

In exploring the space of the tree model, we do not need to specify the leaf parameters explicitly. The leaf parameters are specified after the partition is formed. Therefore we only need to compute the implied marginal likelihood

$$f(\boldsymbol{y}|\mathsf{T}) = \int f(\boldsymbol{y}|\boldsymbol{\theta}_T, \mathsf{T})\pi(\boldsymbol{\theta}_T)d\boldsymbol{\theta}_T. \tag{2.14}$$

In the case that the leaf parameters are independent of each other across leaves, the computation of the marginal likelihood is reduced to the multiplication of the marginal likelihoods in each leaf

$$f(\boldsymbol{y}|\mathsf{T}) = \prod_{u \in b(T)} f(\{y\}_u|\mathsf{T}). \tag{2.15}$$

However, in the case of equation (2.12), we do not have such separation of the

marginal likelihood across leaves. Instead we have

$$f(\boldsymbol{y}|\mathsf{T}) = \int \prod_{u \in b(T)} f(\{y\}_u|\boldsymbol{\theta}, \mathsf{T}) d\boldsymbol{\theta} \tag{2.16}$$

which brings in more complexities in leave-one-out cross validation, as discussed in Chapter 5 and Chapter 7.

In most cases, we assume that the prior $\rho(\boldsymbol{\theta}_u)$ is such that we can analytically compute the implied marginal likelihood, yielding the overall marginal likelihood via tree structures, so the analytical form of the marginal likelihood is available. In the case that there is no conjugate prior, we need numerical approximation of the implied marginal likelihood.

2.3.1 Parameter priors for classification trees

For the leaf distribution specified in equation (1.13), a natural choice for the prior is the beta distribution

$$p_u \sim \text{Beta}(a, b) \tag{2.17}$$

where a and b are the pre-specified hyperparameters. Under this prior, the marginal likelihood for leaf node u is given by

$$f(\{y\}_u|\mathsf{T}) = \frac{B\left(\sum\limits_{j \in \mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j + a, n_u - \sum\limits_{j \in \mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j + b\right)}{B(a, b)} \tag{2.18}$$

where $B(\cdot, \cdot)$ is the beta function, $\mathcal{N}_\mathsf{T}(u, \mathcal{I})$ is all the observations in leaf node u, and $n_u = |\mathcal{N}_\mathsf{T}(u, \mathcal{I})|$ is the number of observations in leaf node u. Therefore the

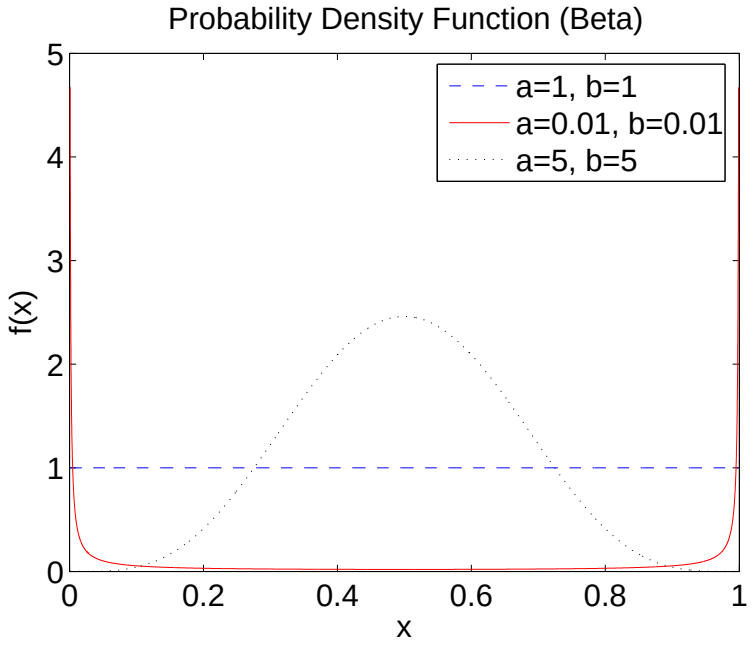


Figure 2.2: The probability density function of the beta distributions for different parameters.

overall marginal likelihood is

$$\begin{aligned} & f(\boldsymbol{y}|\mathsf{T}) \\ = & \prod_{u \in b(T)} f(\{y\}_u|\mathsf{T}) \\ = & \prod_{u \in b(T)} \frac{B(\sum_{j \in \mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j + a, n_u - \sum_{j \in \mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j + b)}{B(a,b)}. \end{aligned} \tag{2.19}$$

Without any prior knowledge on p, we can choose $a = 1$ and $b = 1$. In this case the beta prior in equation (2.17) is uniformly distributed. Alternatively, we can assign small values for both a and b to give prior preference to "clean" leaf nodes. Figure 2.2 displays the probability density function of the Beta distribution for $(a = 5, b = 5)$, $(a = 1, b = 1)$ and $(a = 0.01, b = 0.01)$. The red solid line, corresponding to $(a = 0.01, b = 0.01)$, assigns most of the probability mass to small p and large p.

It is easy to show that for a given tree, the overall marginal likelihood $f(\boldsymbol{y}|\mathsf{T})$ will be larger if the distributions of $\{y\}_u$ at each node are more homogeneous. If the outcome has more than two possible values, a Dirichlet prior is used and the computation of overall marginal likelihood is similar.

2.3.2 Parameter priors for regression trees

For the leaf distribution specified in equation (1.14), we choose the inverse gamma distribution as the prior for σ_u^2 and normal distribution as the prior for μ_u given σ_u^2, namely

$$\begin{array}{rcl} \mu_u|\sigma_u^2 & \sim & \mathrm{N}(\mu_0, \sigma_u^2/n), \\ \sigma_u^2 & \sim & \mathrm{IG}(\alpha, \beta), \end{array} \tag{2.20}$$

where μ_0, n, α and β are pre-specified hyperparameters. A random variable x following a $\mathrm{IG}(\alpha, \beta)$ is defined by its density function

$$f(x) = \frac{1}{\Gamma(\alpha)\beta^\alpha} \frac{1}{x^{\alpha+1}} e^{-\frac{1}{\beta x}}, \quad x > 0. \tag{2.21}$$

Under this prior, the marginal likelihood for leaf node u is given by

$$f(\{y\}_u|\mathsf{T}) = \left(\frac{1}{\sqrt{2\pi}}\right)^{n_u} \frac{\Gamma(\alpha + \frac{n_u}{2})}{\Gamma(\alpha)\beta^\alpha} \sqrt{\frac{n}{n_u + n}} \left(\frac{1}{B_u}\right)^{\alpha+2n_u} \tag{2.22}$$

where

$$B_u = \frac{n\mu_0^2 + \sum_{j\in\mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j^2 - \frac{(n\mu_0 + \sum_{j\in\mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j)^2}{n_u+n}}{2} + \frac{1}{\beta}. \tag{2.23}$$

Then the overall marginal likelihood is just the multiplication of $f(\{y\}_u|\mathsf{T})$ over all the leaf nodes. We can use the observed $\boldsymbol{y}$ to guide the choice of the hyperparameters $(\mu_0, n, \alpha, \beta)$. For example, we may want to choose a small n so that the prior for μ_u is flat.

For the distribution specified in equation (1.15), the prior for (μ_u, σ^2) is given by

$$\begin{array}{rcl} \mu_u|\sigma^2 & \sim & \mathrm{N}(\mu_0, \sigma^2/n), \\ \sigma^2 & \sim & \mathrm{IG}(\alpha, \beta). \end{array} \tag{2.24}$$

Under this prior, for each leaf node u we can compute

$$\begin{array}{rcl} & & f(\{y\}_u|\sigma^2, \mathsf{T}) \\ & = & \sqrt{\frac{n}{n_u+n}}(\frac{1}{\sqrt{2\pi\sigma^2}})^{n_u} e^{-\frac{n\mu_0^2+\sum_{j\in\mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j^2 - \frac{(n\mu_0+\sum_{j\in\mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j)^2}{n_u+n}}{2\sigma^2}} \end{array} \tag{2.25}$$

Therefore the overall marginal likelihood is

$$\begin{array}{rcl} f(\boldsymbol{y}|\mathsf{T}) & = & \displaystyle\int \prod_{u\in b(T)} f(\{y\}_u|\sigma^2, \mathsf{T})\pi(\sigma^2)d\sigma^2 \\ & = & \displaystyle\Gamma(A)\left(\frac{1}{B}\right)^A \frac{1}{\Gamma(\alpha)\beta^\alpha} \prod_{u\in b(T)} \sqrt{\frac{n}{n_u+n}} \left(\frac{1}{2\pi}\right)^{n_u} \end{array} \tag{2.26}$$

where

$$A = \sum_{u\in b(T)} (\alpha + \frac{n_u}{2} + 1) - 1 \tag{2.27}$$

and

$$B = \frac{1}{\beta} + \sum_{u\in b(T)} \frac{n\mu_0^2 + \sum_{j\in\mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j^2 - \frac{(n\mu_0+\sum_{j\in\mathcal{N}_\mathsf{T}(u,\mathcal{I})} y_j)^2}{n_u+n}}{2} \tag{2.28}$$

Note that $f(\boldsymbol{y}|\mathsf{T}) \neq \prod_{u\in b(T)} f(\{y\}_u|\mathsf{T})$.

2.3.3 Parameter priors for survival trees

If the leaf distribution is Weibull as specified in equation (1.16), no conjugate prior is available. We have to compute the marginal likelihood approximately for

some prior. The prior we use is

$$\begin{array}{lcl} \beta_u|\alpha_u & \sim & \Gamma(a,b), \\ \alpha_u & \sim & \text{Unif}(L,U), \end{array} \tag{2.29}$$

where L and U represent prior knowledge on the lower bound and the upper bound of α_u, and a and b are pre-specified hyper-parameters. Generally (a,b) cannot be arbitrarily determined. We know that, given (α_u,β_u), the median survival time m satisfies

$$\beta_u m^{\alpha_u} = \log 2 \tag{2.30}$$

Therefore the choice of (a,b) should change with α_u so that the median survival time from the prior will be consistent with the data. Suppose from observing the data we have prior "guesses" on the median survival time as $(m_1,m_2,\ldots,m_k)$. For each imaginary "observation" on the survival time m_i, we want $f(\beta_u|\alpha_u) \propto \beta_u e^{-\beta_u \frac{m_i^{\alpha_u}}{\log 2}}$ so that the prior mean of β_u satisfies $\mathrm{E}(\beta_u|\alpha_u)m_i^{\alpha_u} = \log 2$. With the k guesses $(m_1,m_2,\ldots,m_k)$, we want $f(\beta_u|\alpha_u) \propto \beta_u^k e^{-\beta_u \frac{\sum_{i=1}^k m_i^{\alpha_u}}{\log 2}}$. Therefore an appropriate choice for (a,b) is $a=k$ and $b=\frac{\sum_{i=1}^k m_i^{\alpha_u}}{\log 2}$.

Under this prior, we can compute $f(\{y,c\}_u|\alpha_u,\mathsf{T})$ analytically

$$\begin{array}{rl} & f(\{y,c\}_u|\alpha_u,\mathsf{T}) \\ = & \alpha_u^{n-n_c} \displaystyle\prod_{j\in\mathcal{N}_{\mathsf{T}}(u,\mathcal{I})\&c_j=0} y_j^{\alpha-1} \frac{b^a}{\Gamma(a)} \frac{\Gamma(n-n_c+a)}{(b+\sum_{j\in\mathcal{N}_{\mathsf{T}}(u,\mathcal{I})} y_j^{\alpha})^{n-n_c+1}} \end{array} \tag{2.31}$$

The further marginalization $\displaystyle\int f(\{y,c\}_u|\alpha_u,\mathsf{T}) f(\alpha_u|\mathsf{T}) d\alpha_u$ can not be computed analytically. Instead we will compute the integral approximately by Riemann summation. Suppose $(\alpha_1,\alpha_2,\ldots,\alpha_N)$ are K uniform grid points from $[L,U]$. The

Riemann sum

$$\sum_{j=1}^{K} f(\{y, c\}_u | \alpha_j, \mathsf{T}) f(\alpha_j | \mathsf{T}) \Delta \tag{2.32}$$

is an approximation to the integral $\int f(\{y, c\}_u | \alpha_u, \mathsf{T}) f(\alpha_u | \mathsf{T}) d\alpha_u$, where $\Delta = (U - L)/K$.

2.4 Discussion

In this chapter, we discussed the prior selections for tree model $\mathsf{T} = \{T, \boldsymbol{k}_T, \boldsymbol{\tau}_T\}$ at length. This is the foundation of our Bayesian tree model. The pinball prior for tree generation allows us to represent our preference for tree size and tree shape. This prior does not involve the specification of the splitting rules. This feature is key to the design of the innovative *Restructure* proposal, to be introduced in Chapter 3. Moreover, the basic proposals, introduced by Chipman *et al.* (1998) and Denison *et al.* (1998), also benefit from this pinball prior as the computation cost can be greatly reduced.

We also discussed the choice of prior for splitting rules. We will use the independent prior for both the choice of splitting variable and the splitting threshold; this has the advantage of simplifying the calculation of acceptance probabilities in the Metropolis-Hastings algorithm. Further, we also propose other prior specifications. At each level, the prior for the choice of splitting variable and splitting threshold depends on the status of the ancestor nodes. The advantage of this prior specification is discussed. We do not cover the comparison of these two prior specifications, which is an area for future research. Using the Dirichlet process prior, we interpret one special choice of the prior for splitting threshold, where

the splitting threshold only takes values from some grid points from the data.

In the last part of this chapter, we argue that the likelihood function depends on T only through the induced partition of observations to the leaves. In this case, we do not need to specify the leaf node parameters explicitly. This feature reduces the size of the tree model and makes it easier to design the proposals in the Metropolis-Hastings algorithm in Chapter 3.

With the well-defined prior for tree model and the Metropolis Algorithm in Chapter 3, we will be able to explore the posterior distribution. Several examples will be studied in later chapters.

Chapter 3

Posterior Analysis

With the prior $\pi(\mathsf{T}) = \pi(T)\pi(\boldsymbol{k}_T|T)\pi(\boldsymbol{\tau}_T|T, \boldsymbol{k}_T)$ defined in Chapter 2, we have the posterior on tree model space given by

$$\pi(\mathsf{T}|\boldsymbol{y}) \propto \pi(\mathsf{T}) f(\boldsymbol{y}|\mathsf{T}), \tag{3.1}$$

In this specification, there is no restriction on the number of data points in the leaf. In this case, a tree with an empty leaf might be sampled. Such trees are not desirable in terms of computation and inference. A technical point is that we actually need to restrict the choice to trees with non-empty leaves; more generally, we may decide to require a minimum of say, h data points in leaf, and thus we aim to sample from

$$\tilde{\pi}(\mathsf{T}|\boldsymbol{y}) \propto \pi(\mathsf{T}|\boldsymbol{y}) \prod_{u \in b(T)} I[|\mathcal{N}_{\mathsf{T}}(u, \mathcal{I})| \geq h]. \tag{3.2}$$

In the following text, if not otherwise explicitly stated we use $\pi(\mathsf{T}|\boldsymbol{y})$ to denote the posterior on the tree model space with the restriction ($h = 1$) to simplify the notation.

If we further restrict the splitting threshold on some discrete values, there will be only a finite number of trees of interest since any tree with the size greater than the number of observation has a posterior probability of 0. However, even though in this special case the number of trees is finite, there will be too many trees to evaluate, as discussed in Chapter 1 and Chapter 2, and therefore a direct sampling is not feasible. For such huge spaces, a Metropolis-Hastings algorithm is used to explore the posterior tree space.

3.1 Metropolis-Hastings algorithm

For an arbitrary tree T^0, a chain (T^n) is generated using the Metropolis-Hastings algorithm. A conditional density $q(y|x)$, the support of which is larger than our objective density $\pi(\mathsf{T}|\boldsymbol{y})$, is then defined. Under mild conditions, such algorithms, as described in Figure 3.1, produce a chain with limiting distribution $\pi(\mathsf{T}|\boldsymbol{y})$. Thus, for a large enough n, T^n is considered to have distribution from $\pi(\mathsf{T}|\boldsymbol{y})$. In practice we usually generate dependent samples $\mathsf{T}^n, \mathsf{T}^{n+d}, \mathsf{T}^{n+2d}, \ldots$ for some d.

Metropolis-Hastings Algorithm
Starting from T^0, iteratively simulate from T^i to T^{i+1} by
Generate $\mathsf{T}' \sim q(\mathsf{T}'|\mathsf{T}^i)$
Take $\mathsf{T}^{i+1} = \mathsf{T}'$ with probability $\rho(\mathsf{T}^i, \mathsf{T}')$ and $\mathsf{T}^{i+1} = \mathsf{T}^i$ otherwise
where $\rho(\mathsf{T}^i, \mathsf{T}') = \min\left\{\frac{\pi(\mathsf{T}'|\boldsymbol{y})}{\pi(\mathsf{T}^i|\boldsymbol{y})}\frac{q(\mathsf{T}^i|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T}^i)}, 1\right\}$.
End.

Figure 3.1: Metropolis-Hastings algorithm for generating trees.

Note that the normalizing constant is not needed to calculate the acceptance probability $\rho(\mathsf{T}^i, \mathsf{T}')$. To implement this algorithm, we need to specify the conditional density proposal $q(\mathsf{T}'|\mathsf{T})$. The Metropolis-Hastings methods in Chipman

et al. (1998) and Denison *et al.* (1998) are quite similar and form our starting points. These two key articles also share the same honest conclusion: their MCMC methods, based on "local moves" around the tree space, are extraordinarily slow to converge. They recommend restarts of the MCMC algorithm combined with ad-hoc weighting of the generated trees. Because the calculation of the weights is not clear, inferences based on model averaging does not seem appropriate. Our major contribution here is to introduce novel MCMC moves that seem to solve the convergence problem, without restarts or weighting of the realizations. All trees (after convergence) should be assigned the same weight for model averaging. Moreover, if the distribution has multiple modes, the probability mass contained in a specific mode can be easily estimated by the fraction of time that the Markov chain spends in the mode.

Our MCMC method uses four Metropolis-Hastings proposals, namely "change", "grow/prune", "swap" and "(radical) restructure". The first three proposals are those of Chipman *et al.* (1998) and Denison *et al.* (1998). In the notation, we use primes to indicate proposal/potential new values so that, for example, T and T' are the current and the proposed/candidate new trees, respectively. At each step, one of the four proposals is used with some probability. For example, the three basic moves may be chosen with probability 0.33 respectively and the restructure move may be chosen with probability 0.01. Alternatively, one can take one iteration to mean a series of basic moves and restructure moves, for example, 50 change moves, 50 grow/prune moves, 50 swap moves and 1 restructure move. In this text, we use the latter approach since it is easier to compare between algorithms with restructure move and without restructure move.

To illustrate these four proposals, we construct a data set as shown in Table 3.1.

Obs	1	2	3	4	5	6
x^1	0.00	0.10	0.20	0.30	0.40	0.50
x^2	0.10	0.30	0.40	0.48	0.56	0.57
Obs	7	8	9	10	11	12
x^1	0.55	0.60	0.65	0.70	0.80	0.90
x^2	0.00	0.20	0.40	0.60	0.80	1.00

Table 3.1: Example data, with $n = 12$, used to illustrate the Metropolis–Hastings proposals.

There are twelve observations and two predictors in this dataset. Both predictors are continuous variables. The proposals are described as follows:

3.2 Basic moves

The three basic moves, namely "change", "grow/prune" and "swap", propose a small change to the tree structure and/or the splitting rules, and thus are called local moves.

3.2.1 Change proposal

For the dataset in Table 3.1, an illustration of the move is shown in Figure 3.2. The tree in the left panel is the current one. The splitting rule $X_2 < 0.5$ in node $u = 2$ is chosen to be changed. The proposed new splitting rule is $X_2 < 0.3$. As a result, the $9th$ observation is moved to leaf $L4$. Note that this change proposal does not change the tree structure or other splitting rules. So in the case that the new splitting rule is drawn independently of the other splitting rules in the tree and the independent prior for splitting rules is used, the calculation of the posterior probability ratio involves only the change in the likelihood and in the prior probability associated with the chosen splitting rule. The calculation can

then be much simplified, as

$$\begin{aligned} \rho(\mathsf{T}, \mathsf{T}') &= \min\left\{\frac{\pi(\mathsf{T}'|\boldsymbol{y})}{\pi(\mathsf{T}|\boldsymbol{y})}\frac{q(\mathsf{T}|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T})}, 1\right\} \\ &= \min\left\{\frac{f(\boldsymbol{y}|T')\pi(k'(u)|T')\pi(\tau'(u)|k', T')}{f(\boldsymbol{y}|T)\pi(k(u)|T)\pi(\tau(u)|k, T)}\frac{q(\mathsf{T}|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T})}, 1\right\}. \end{aligned} \tag{3.3}$$

It is easy to see that the change move is reversible. In practice, we usually choose $q(\mathsf{T}'|\mathsf{T}^i)$ to be symmetric so that the ratio $\dfrac{q(\mathsf{T}^i|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T}^i)} = 1$.

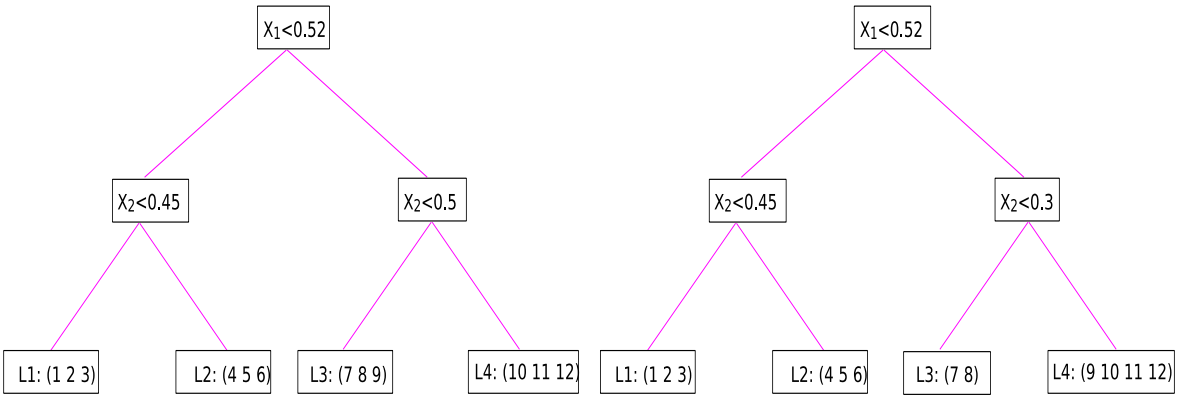


Figure 3.2: Illustration of the change proposal. The tree on the left is the current one in the chain. The splitting rule in node 2 is changed and the tree on the right is proposed.

Figure 3.3 contains pseudo-code for the change proposal mechanism. In proposing the new splitting variable and the corresponding splitting rule, we usually draw them from the prior. In most cases, the independent prior for splitting rules is used.

Function ProposeChange(T,$\mathcal{I}$)
  Set $T' = T$.
  Draw u from a uniform distribution on $a(T')$.
  Draw $k_{T'}(u)$ from $\gamma(\cdot)$.
  Draw $\tau_{T'}(u)$ from $\delta_{k_{T'}(u)}(\cdot)$.
  Set $k_{T'}(v) = k_T(v)$ and $\tau_{T'}(v) = \tau_T(v)$ for $v \in a(T) \setminus \{u\}$.
  Return T'
End.

Figure 3.3: Change Proposal: pseudo-code for generating a potential new tree in the change move. Notation from Chapter 1 is used.

3.2.2 Grow/prune proposal

Grow/prune proposals propose to split a randomly selected leaf into two, or to prune the tree by merging two randomly selected sibling leaves. For the constructed data set shown in Table 3.1, an illustration of the move is shown in Figure 3.4. If we propose a grow/prune move, we will perform a grow move with probability 0.5 and a prune move otherwise. For example, suppose the tree shown in the left panel of Figure 3.4 is the current one. If we decide to perform a grow move, one of the leaf nodes will be chosen. In this example, $L3$ is chosen and we assign a new splitting rule $X_2 < 0.5$ to this node. This node is thus split into two leaf nodes and itself becomes an internal node. The leaf $L3$, which contains observations $(7, 8, 9, 10, 11, 12)$, is then split into two, resulting new leaf nodes $L3$ and $L4$, which contains observations $(7, 8, 9)$ and $(10, 11, 12)$ respectively. Noting that this grow move proposes a new splitting rule as well as a change in the tree structure, the calculation of the posterior probability ratio involves not only a change in the likelihood and splitting rules but also a change in tree structure. However, in grow moves the change in likelihood involves only the chosen leaf in the current tree. The calculation of acceptance probability can be simplified since

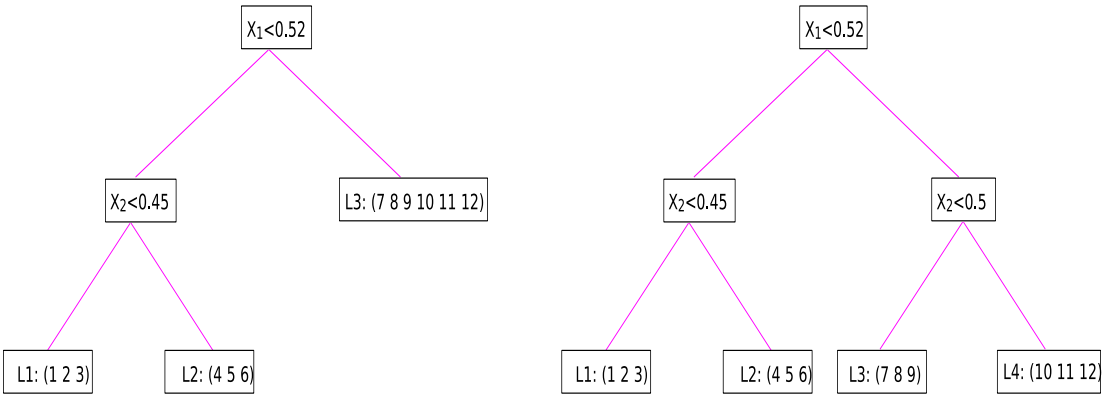


Figure 3.4: Illustration of grow/prune proposal. From the tree on the left to the one on the right, it is a grow move. In the reverse direction, it is a prune move.

the likelihood of the other leaves remains unchanged.

$$\begin{aligned} & \rho(\mathsf{T}, \mathsf{T}') \\ = \; & \min\left\{\frac{\pi(\mathsf{T}'|\boldsymbol{y})}{\pi(\mathsf{T}|\boldsymbol{y})}\frac{q(\mathsf{T}|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T})}, 1\right\} \\ = \; & \min\left\{\frac{f(\tilde{\boldsymbol{y}}|T')\pi(k_1'(u), k_2'(u)|T')\pi(\tau_1'(u), \tau_2'(u)|k_1', k_2'(u), T')\pi(T')}{f(\tilde{\boldsymbol{y}}|T)\pi(k(u)|T)\pi(\tau(u)|k, T)}\frac{q(\mathsf{T}|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T})}, 1\right\} \end{aligned} \tag{3.4}$$

It is easy to see that grow and prune moves are each reversible and complementary. For example, if the tree shown in the right panel of Figure 3.4 is the current tree, one of the possible new trees after prune move is the one shown in the left panel. The calculation of the acceptance probability is similar to equation (3.4). Figure 3.5 gives pseudo-code for a grow/prune move. Similar to the change move, the new splitting rule is usually drawn from the prior.

In the case that all the splitting thresholds for the corresponding splitting variables are restricted to the grid points from the data, there will be only a finite

Function ProposeGrowPrune(T,$\mathcal{I}$)
  With probability $\frac{1}{2}I(m_0(T) > 1)$ set $I_{\text{grow}} = 0$, else set $I_{\text{grow}} = 1$.
  If $(I_{\text{grow}} = 1)$
    Draw u from a uniform distribution on $b(T)$.
    Set $T' = T \cup \{l(u), r(u)\}$.
    Draw $k_{T'}(u)$ from $\gamma(\cdot)$.
    Draw $\tau_{T'}(u)$ from $\delta_{k_{T'}(u)}(\cdot)$.
    Set $k_{T'}(v) = k_T(v)$ and $\tau_{T'}(v) = \tau_T(v)$ for each $v \in a(T)$.
  Else
    Draw u from a uniform distribution on the set $\{u \in a(T) | l(u), r(u) \in b(T)\}$.
    Set $T' = T \setminus \{l(u), r(u)\}$.
    Set $k_{T'}(v) = k_T(v)$ and $\tau_{T'}(v) = \tau_T(v)$ for each $v \in a(T')$.
  End.
  Return T'.
End.

Figure 3.5: Grow/Prune Proposal: pseudo-code for generating a potential new tree in grow/prune move. Notation from Chapter 1 is used.

number of trees in the posterior space, as mentioned in Section 2.2.2. In this case, the grow/prune proposal (and other proposals) just move the chain in a finite discrete space. However, in the case that the splitting threshold can take any value and the prior distribution for the splitting threshold is continuous, the dimension of the parameters is changed when the grow/prune move is proposed. Thus the grow/prune proposal is a reversible jump type of move. Green (1995) shows that the appropriate acceptance probability for such move should be

$$\rho(\mathsf{T}, \mathsf{T}') = \min \left\{ \frac{\pi(\mathsf{T}'|\boldsymbol{y})}{\pi(\mathsf{T}|\boldsymbol{y})} \frac{q(\mathsf{T}|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T})} \left| \frac{\partial \mathsf{T}'}{\partial \mathsf{T}} \right|, 1 \right\} \tag{3.5}$$

The only difference between equation (3.5) and (3.4) is the Jacobian; however, as we propose the new splitting variables and splitting thresholds independently of the other splitting rules, the associated Jacobian will always equal unity. This is

true even a dependent prior for splitting rules is used, so in the following analysis, we will simply refer to equation (3.4) instead of equation (3.5).

3.2.3 Swap proposal

The swap move proposes to swap the splitting rules in a randomly selected parent-child pair that are both internal nodes. For the constructed data shown in table 3.1, an illustration of a swap move is shown in Figure 3.6. The left panel shows the current tree. In this tree, the possible parent-child pairs to be selected are $(0, 1)$ and $(0, 2)$. Suppose $(0, 1)$ is chosen and thus the splitting rules associated with node 0 and 1 are swapped. As a result, the leaf node $L2$, $L3$ and $L4$ have been changed. More generally, after the swap move proposed to the parent-child pair $(u, l(u))$ or $(u, r(u))$, all the leaf node $\mathcal{L}_{S_u(\mathsf{T})}(\mathcal{N}_{\mathsf{T}}(u, \mathcal{I}))$ will be changed. Therefore the calculation of the acceptance probability usually involves a big change in the likelihood if the chosen parent node is in the upper level. But the swap move just swaps the splitting rules in the chosen parent-child pair and does not change the tree structure and the splitting rules. Therefore when the independent priors for splitting rules are used, the swap move does not change the prior ratio and the posterior ratio can be simplified to the likelihood ratio. The acceptance probability is

$$\begin{aligned} \rho(\mathsf{T}, \mathsf{T}') &= \min\left\{\frac{\pi(\mathsf{T}'|\boldsymbol{y})}{\pi(\mathsf{T}|\boldsymbol{y})}\frac{q(\mathsf{T}|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T})}, 1\right\} \\ &= \min\left\{\frac{f(\boldsymbol{y}|T')}{f(\boldsymbol{y}|T)}\frac{q(\mathsf{T}|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T})}, 1\right\}. \end{aligned} \tag{3.6}$$

However, if the dependent prior for splitting rules is used, the prior probability is changed when the splitting rules are swapped. In this case, the calculation of prior ratio cannot be simplified and the prior probability for the splitting rules in

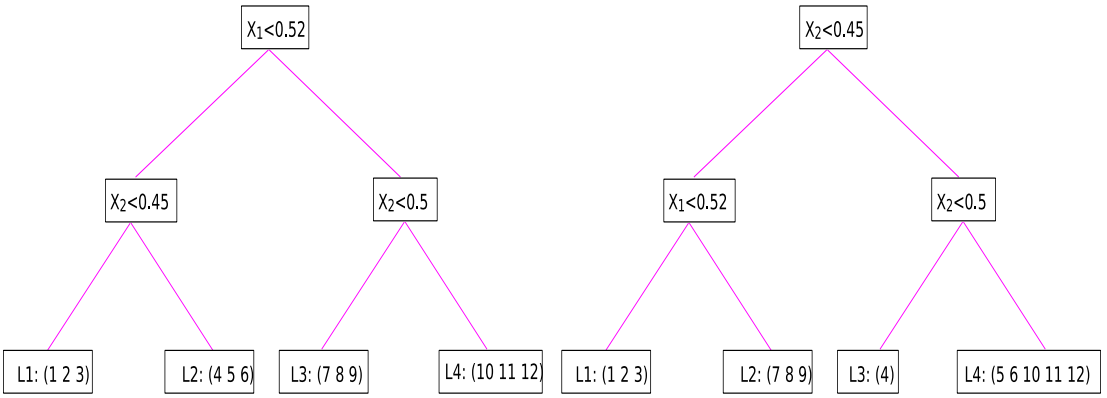


Figure 3.6: Illustration of a swap proposal. The parent-child pair $(0, 1)$ of the current tree in the left panel is chosen and swapped. The resulting tree is shown in the right panel.

$S_u(\mathsf{T})$ need to be evaluated.

A swap move is reversible to itself. Figure 3.7 contains the pseudo-code for the swap proposal mechanism. Out of all the possible parent-child pairs, we choose one of them uniformly. Note that the resulting new tree after the swap proposal could have an empty leaf. For example, if the parent-child pair $(0, 2)$ in the tree displayed in the right panel of Figure 3.6 is chosen to be swapped, that is, the splitting rules $x^2 < 0.45$ and $x^2 < 0.5$ are swapped, the right child of node 2 in the new tree is an empty leaf node. In this case, we just keep the current tree.

3.3 Restructure Moves

From Section 3.2, we can see that the three basic moves only change the tree model $(\mathsf{T} = (T, \boldsymbol{k}, \boldsymbol{\tau}))$ locally and thus change the partition of observations into leaves. In the huge tree space, one should expect there are many "good" trees with high

Function ProposeSwap(T,$\mathcal{I}$)
  Set $T' = T$.
  Draw (u_1, u_2) from a uniform distribution on $a(T')$
    that (u_1, u_2) is a parent-child pair.
  Set $k_{T'}(u_1) = k_T(u_2)$, and $\tau_{T'}(u_1) = \tau_T(u_2)$.
  Set $k_{T'}(u_2) = k_T(u_1)$, and $\tau_{T'}(u_2) = \tau_T(u_1)$.
  Set $k_{T'}(v) = k_T(v)$ and $\tau_{T'}(v) = \tau_T(v)$ for $v \in a(T) \setminus \{u_1, u_2\}$.
  Return T'
End.

Figure 3.7: pseudo-code for the swap move.

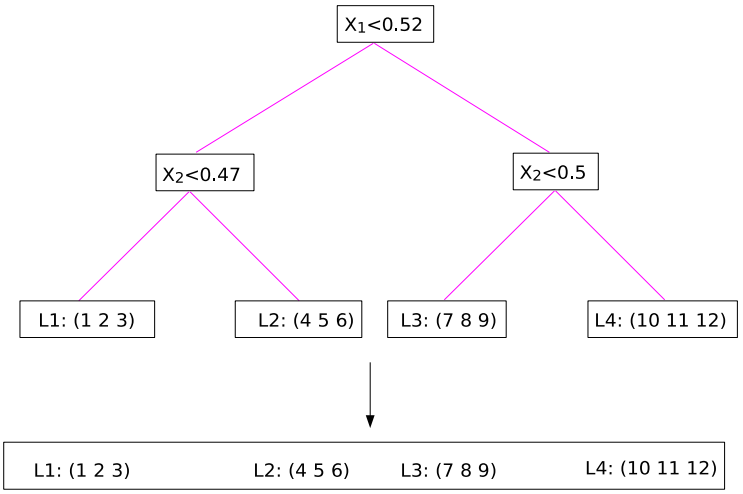
probabilities. Using only basic moves, it may take a long time for the MCMC methods to move from one such good tree to the others. The restructure move is a new proposal which aims to propose large changes in $\mathsf{T} = (T, \boldsymbol{k}, \boldsymbol{\tau})$ without changing the number of leaves nor the partition of observations into leaves. In this section we introduce the restructure move and its variations.

3.3.1 The restructure proposal

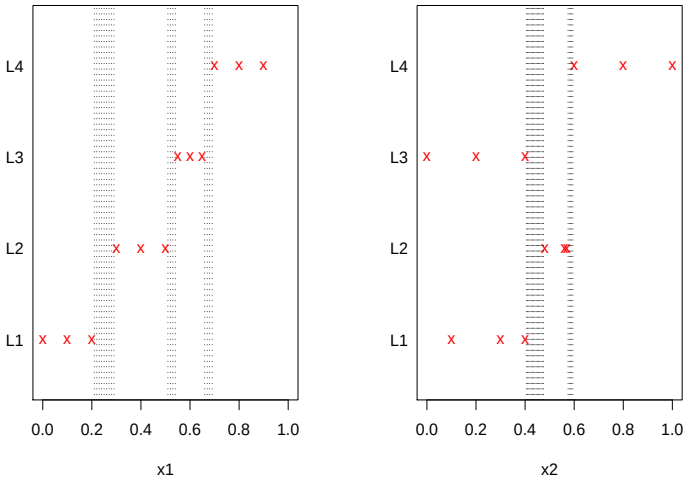
An illustration of the restructure move is shown in Figure 3.8 and Figure 3.9 for the constructed data shown in Table 3.1. Suppose the tree shown in Figure 3.8(a) is the current one. Then the restructure proposal for this tree consists of the following steps:

1. For the tree model T shown in Figure 3.8(a), we first find the corresponding partition of observations into leaves, $\mathcal{L}_{\mathsf{T}}(\mathcal{I})$. Then the tree model T will be discarded and a new tree model $\mathsf{T}' = (T', \boldsymbol{k}', \boldsymbol{\tau}')$ with the same partition $\mathcal{L}_{\mathsf{T}}(\mathcal{I}) = \mathcal{L}_{\mathsf{T}'}(\mathcal{I})$ will be proposed;

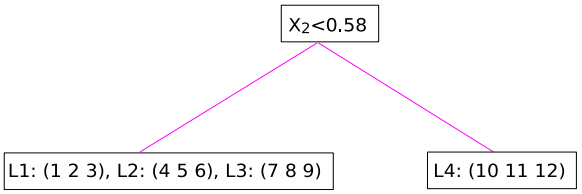
2. In order to propose a candidate T', we first identify all the possible pairs of splitting variables and splitting thresholds for node 0. We require that there



(a)

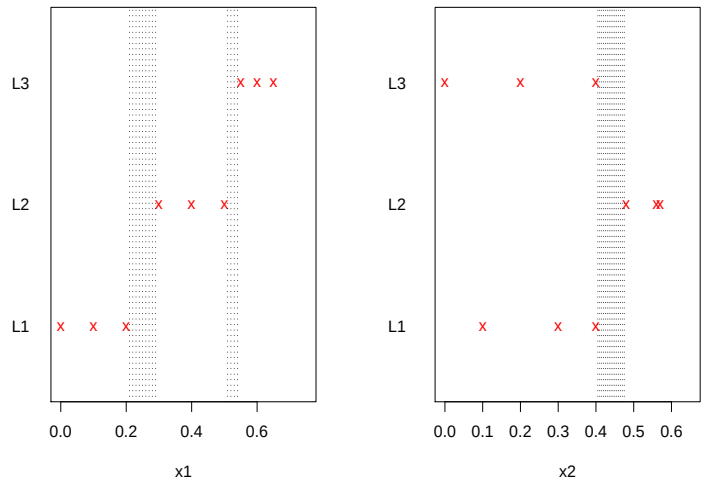


(b)

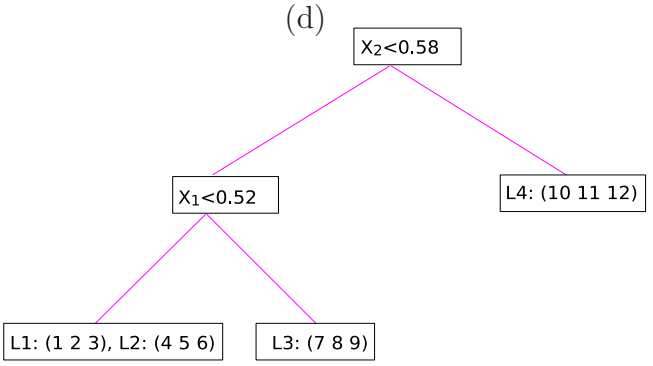


(c)

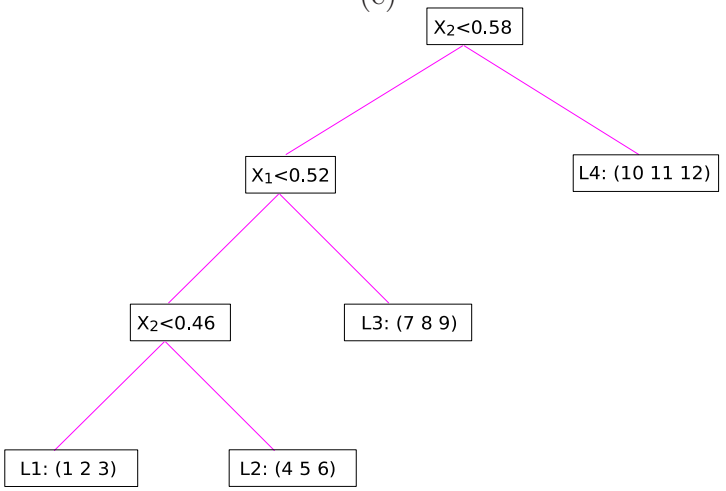
Figure 3.8: Illustration of a restructure move with the constructed data set in Table 3.1. Note that the figure continues on the next page. (a) Current tree and the corresponding partition of observations into leaves; (b) Possible splits for the root node; (c) Choose the first splitting rule and make the split;



(d)



(e)



(f)

Figure 3.9: (cont.) Restructure move: (d) Possible splits for node 1; (e) Choose the splitting rule and make the split for node 1; (f) The candidate tree proposed by the restructure move.

be at least one leaf in each of the two subtrees $S_1(T')$ and $S_2(T')$ of node 0. In Figure 3.8(b) all the possible intervals for $\tau'(0)$ for $k'(0) = 1$ and for $k'(0) = 2$ are shaded. Three possible intervals exist for $k'(0) = 1$, and two intervals for $k'(0) = 2$. They are $\tau'(0) \in (0.2, 0.3) \cup (0.5, 0.55) \cup (0.65, 0.75)$ when x^1 is chosen and $\tau'(0) \in (0.4, 0.5) \cup (0.57, 0.60)$ when x^2 is chosen. We propose values for $(k'(0), \tau'(0))$ by first sampling one of the five possible intervals uniformly at random and thereby generating a value for $\tau'(0)$ uniformly at random within the chosen interval. For example, if the interval $(0.57, 0.60)$ corresponding to x^2 is chosen, we therefore set $k'(0) = 2$. By proposing new splitting threshold $\tau'(0)$, say 0.58, we find the new splitting rule $x^2 < 0.58$ for node 0 of the new tree.

3. After the splitting rule is proposed, T' is partly specified by $(k'(0), \tau'(0))$, $\mathcal{L}_{S_1(\mathsf{T}')}(\mathcal{I})$ and $\mathcal{L}_{S_2(\mathsf{T}')}(\mathcal{I})$, as shown in Figure 3.8(c). More specifically, the splitting rule $x^2 < 0.58$ produces $\mathcal{L}_{S_1(\mathsf{T}')}(\mathcal{I}) = \{\{1, 2, 3, \}, \{4, 5, 6\}, \{7, 8, 9\}\}$ and $\mathcal{L}_{S_2(\mathsf{T}')}(\mathcal{I}) = \{\{10, 11, 12\}\}$. The right subtree $S_2(\mathsf{T}')$ now contains only one leaf $\mathcal{L}_{S_2(\mathsf{T}')}(\mathcal{I})$ ($L4$ in the figure). By the restriction that there is at least one leaf in the subtree, the subtree from this node has been completely specified and no further split will be made for this node.

4. The left subtree $S_1(\mathsf{T}')$ contains three leaves and is split into two subtrees, $\mathcal{L}_{S_3(\mathsf{T}')}(\mathcal{I})$ and $\mathcal{L}_{S_4(\mathsf{T}')}(\mathcal{I})$, by repeating the process just described. Figure 3.9(d) shows the possible intervals for $\tau'(1)$ for $k'(1) = 1$ and $k'(1) = 2$. We again follow the sampling method described in step 2 and a possible result is shown in Figure 3.9(e). In Figure 3.9(e), the splitting rule $x^1 < 0.52$ is set for node 1 and thereby the right subtree $S_4(\mathsf{T}')$ is completely specified.

Thus T' is now partially specified by $\{(k'(u), \tau'(u)), u = 1, 2\}$, $\mathcal{L}_{S_2(\mathsf{T}')}(\mathcal{I})$, $\mathcal{L}_{S_3(\mathsf{T}')}(\mathcal{I})$ and $\mathcal{L}_{S_4(\mathsf{T}')}(\mathcal{I})$.

5. The subtree $S_3(\mathsf{T}')$ still contains more than one leaf and must therefore be split into two subtrees by repeating the above process once more. The possible intervals for $\tau'(3)$ are identified (this step is not shown in Figure 3.9) and used to produce values for $(k'(3), \tau'(3))$.

6. The resulting completely specified T' is shown in Figure 3.9(f).

Notice that the new tree shown in Figure 3.9(f) has the same leaf configuration as in the old tree shown in Figure 3.8(a) after restructure proposal, but the tree structure and the splitting rules are much different. Because the leaf configuration remains unchanged, the likelihood will be the same, that is, $f(\boldsymbol{y}|\mathsf{T}') = f(\boldsymbol{y}|\mathsf{T})$. Therefore in calculating the posterior ratio, we only need to compute the prior ratio, which generally is easy to compute.

Therefore the calculation of the acceptance probability can be simplified to

$$\begin{aligned} \rho(\mathsf{T}, \mathsf{T}') &= \min\left\{\frac{\pi(\mathsf{T}'|\boldsymbol{y})}{\pi(\mathsf{T}|\boldsymbol{y})}\frac{q(\mathsf{T}|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T})}, 1\right\} \\ &= \min\left\{\frac{\pi(\mathsf{T}')}{\pi(\mathsf{T})}\frac{q(\mathsf{T}|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T})}, 1\right\}. \end{aligned} \tag{3.7}$$

Note that with the above proposal procedure there is a unique way to generate T' from T, and a corresponding unique way to propose T from T'. Computation of the associated proposal probability $q(\mathsf{T}'|\mathsf{T})$, and the probability $q(\mathsf{T}|\mathsf{T}')$ for the corresponding reverse move, follows directly from the generating procedure. The proposal probability $q(\mathsf{T}'|\mathsf{T})$ is a product with two factors associated with each internal node of T'; correspondingly, $q(\mathsf{T}|\mathsf{T}')$ has two factors for each internal node

of T. The first factor comes from drawing an interval and is one divided by the number of possible intervals. The second factor, associated with drawing a value for $\tau'(u)$, is one divided by the length of the allowed interval.

Figure 3.10 gives the pseudo-code for the proposal mechanism. Note that in the process of splitting $\mathcal{L}_{S_u(\mathsf{T}')}(\mathcal{I})$ into $\mathcal{L}_{S_{l(u)}(\mathsf{T}')}(\mathcal{I})$ and $\mathcal{L}_{S_{r(u)}(\mathsf{T}')}(\mathcal{I})$ there is a risk that no possible values exist for the pair $(k'(u), \tau'(u))$. If this happens in any of the splits we just set $\mathsf{T}' = \mathsf{T}$, i.e. keep the tree unchanged and increment the iteration variable by one. One should note that even if the tree is kept unchanged in this case, it is essential to count it as an iteration of the Metropolis–Hastings algorithm; otherwise, it would be necessary to compute the probability, by simulation, for this event to happen, which in practice would be impossible except for very small trees.

3.3.2 Variations of the restructure proposal

In this section, we will briefly introduce two variations of the restructure proposal. The first one is designed to reduce the computational cost, while the second one explores a more flexible restructure proposal.

In the restructure move one needs to map all possible splitting variables and the corresponding splitting intervals for each internal node in the new tree. The computational cost is proportional to the number of candidate predictor variables p. In Figure 3.8(b), there are five possible splitting intervals for just two predictors. If p is very large, a complete mapping of possible splitting variables becomes computationally expensive. One may therefore select only a randomly chosen subset of the predictor variables. Letting $C \subseteq \mathcal{P}$ denote the subset of predictor

variables mapped, a simple alternative is to let

$$\mathrm{P}(i \in C|\mathsf{T}) = \begin{cases} 1 & \text{if } i \in \{k_T(u) : u \in \mathsf{T}\}, \\ \alpha & \text{otherwise,} \end{cases} \tag{3.8}$$

independently for each $i \in \mathcal{P}$. The candidate splitting variables set C, from which the restructure move will draw the new splitting variables, includes all the splitting variables used in the current tree with probability 1 and the remaining splitting variables with probability α. The parameter $\alpha \in (0, 1)$ can be used to control the typical size of C. For example, if α is 1, this is just the basic restructure proposal. If α is 0, only splitting variables in the current tree will be considered to be the splitting variables in the new tree. This greatly reduces the computation time, however, it may take a long time for the chain to converge, as discussed in the synthetic example in Chapter 4.

With the definition of C, the restructure move can then be started by sampling a set C. The Metropolis-Hastings acceptance probability (equation (3.7)) must be modified by including the probability of sampling C.

$$\begin{aligned} \rho(\mathsf{T}, \mathsf{T}') &= \min\left\{\frac{\pi(\mathsf{T}'|\boldsymbol{y})}{\pi(\mathsf{T}|\boldsymbol{y})}\frac{q(\mathsf{T}|\mathsf{T}', C)P(C|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T}, C)P(C|\mathsf{T})}, 1\right\} \\ &= \min\left\{\frac{\pi(T')}{\pi(T)}\frac{q(\mathsf{T}|\mathsf{T}', C)P(C|\mathsf{T}')}{q(\mathsf{T}'|\mathsf{T}, C)P(C|\mathsf{T})}, 1\right\}. \end{aligned} \tag{3.9}$$

A second possible modification allows for changes in the partition of observations in leaves. In the ProposeRestructure function (Figure 3.10) a new partition L' can be proposed just after the leaves, L, have been picked. Different possibilities exist for how to sample L' given L. An alternative based on the likelihood function is given in Figure 3.11, where we use the fact that the likelihood function only depends on the partition of observations into leaves. After the leaves are picked,

we uniformly choose one of the leaves to be the "from" leaf and uniformly choose one of the remaining leaf to be the "to" leaf. If the size of the "from" leaf is 1, then we just propose an unchanged tree; however, if the size of the "from" leaf is s, we will have s new leaf configurations by moving one leaf from the "from" leaf to the "to" leaf. With the current leaf configurations we will have $s + 1$ different leaf configurations in total, among which one will be uniformly chosen. This defines one iteration in this variation of the restructure move. Repeating this process r_{iter} times will yield a shuffled leaf configurations. This variation further increases the computational cost, but hopefully proposes a new tree with different prior probability and likelihood. The behavior of such move is also more difficult to describe.

3.3.3 Restructure move vs basic moves

Figure 3.12 illustrates the difference between restructure moves and basic moves. In this figure, although it is not realistically possible, suppose we have an imaginary way of indexing the tree models. If two trees are close to each other (denoted as T and $T + \delta T$ in this figure) using this index, then a basic move can be performed to move from one tree to the other. So the basic moves described above can move the Markov chain along the curve. We expect, in this complicated tree model space, that there will be multiple modes, and it takes a long time for the Metropolis-Hastings algorithm with only basic moves to move from one local mode to another.

On the contrary, instead of moving the chain along the curve, the restructure move tries to find trees with the same likelihood but with a significantly different tree structure and associated splitting rules. This move is illustrated by the red

```
Function ProposeRestructure(T,I)
   Pick the leaves of the current tree, i.e. set L = L_T(I).
   [T', err] = DrawTree(L).
   If (err = 0)
      Return T'
   Else
      Return T
   End.
End.
```

```
Function DrawTree(L)
   Let m denote the number of leaves in L, i.e. m = |L|.
   If (m > 1)
      Find the set of all possible pairs of splitting variables
         and splitting intervals for L. Denote the result
         by {k_j, (τ_j^lower, τ_j^upper), j = 1, ..., n_I}.
      If (n_I > 0)
         Draw k(0) from a uniformly distribution on {1, ..., n_I}.
         Draw τ(0) from a uniform distribution on
            the interval (τ_k(0)^lower, τ_k(0)^upper).
         Set L_l and L_r equal to the subsets of L which have
            x^k(0) < τ(0) and x^k(0) ≥ τ(0), respectively.
         [S_1(T), err_1] = DrawTree(L_l); [S_2(T), err_2] = DrawTree(L_r).
         If (err_1 = 0 and err_2 = 0)
            Set err = 0.
            Return [T, err].
         End.
      End.
      Set T = ∅, T = [T, ·, ·] and err = 1.
      Return [T, err].
   Else                              /* Only one node in the tree */
      Set T = {0}, T = [T, ·, ·] and err = 0.
      Return [T, err].
   End.
End.
```

Figure 3.10: Restructure Proposal: pseudo-code for generating a potential new tree, and the DrawTree function used. Notation from Chapter 1 is used.

Function ProposePartition(L,r_{iter})
  Set $\widetilde{\mathsf{L}}^0 = \mathsf{L}$.
  Let m denote the number of leaves in $\widetilde{\mathsf{L}}^0$ and let $\widetilde{\mathsf{L}}^0_l$ denote
    leaf number l, i.e. $\widetilde{\mathsf{L}}^0 = (\widetilde{\mathsf{L}}^0_1, \ldots, \widetilde{\mathsf{L}}^0_m)$.
  For $(i = 1 : r_{\text{iter}})$
    Draw l_{from} from a uniform distribution on $\{1, \ldots, m\}$.
    Draw l_{to} from a uniform distribution on $\{1, \ldots, l_{\text{from}} - 1,$
      $l_{\text{from}} + 1, \ldots, m\}$.
    Let s denote the number of elements (observations) in $\widetilde{\mathsf{L}}^{i-1}_{l_{\text{from}}}$.
    If $(s > 1)$
      Use $\widetilde{\mathsf{L}}^{i-1}$ to define s new leaf configurations by moving one
        observation from leaf number l_{from} to leaf number l_{to}. Denote
        these configurations by $\breve{\mathsf{L}}^1, \ldots, \breve{\mathsf{L}}^s$.
      Set $\breve{\mathsf{L}}^0 = \widetilde{\mathsf{L}}^{i-1}$.
      Draw an integer $t \in \{0, 1, \ldots, s\}$ where $\mathrm{P}(t = j) \propto f(\boldsymbol{y}|\breve{\mathsf{L}}^j, \boldsymbol{\theta})$.
      Set $\widetilde{\mathsf{L}}^i = \breve{\mathsf{L}}^t$.
    Else
      Set $\widetilde{\mathsf{L}}^i = \widetilde{\mathsf{L}}^{i-1}$.
    End.
  End.
  Return $\widetilde{\mathsf{L}}^{r_{\text{iter}}}$
End.

Figure 3.11: pseudo-code for generating a potential new leaf configuration to be used in the restructure proposal.

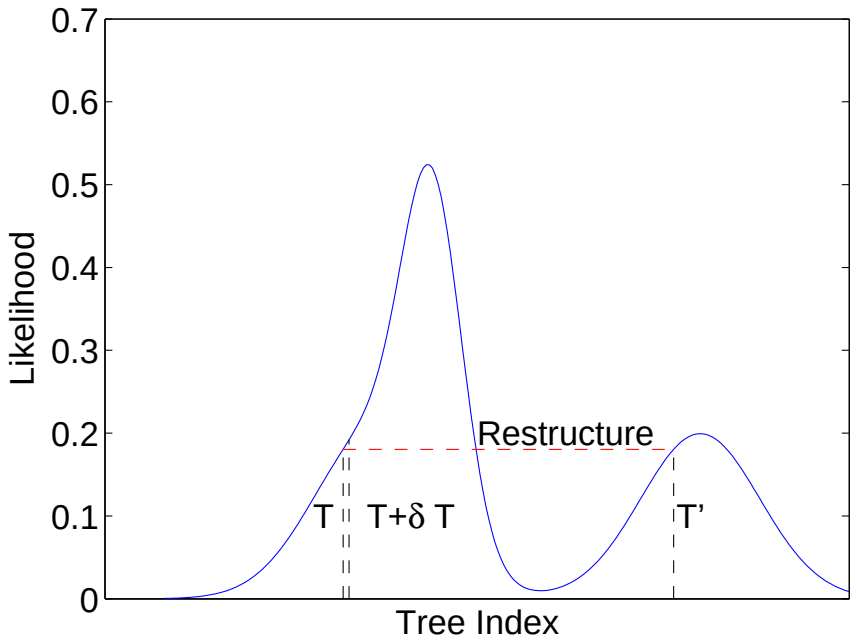


Figure 3.12: An illustration of restructure move and basic moves.

dash line in the figure. By performing such moves, we try to jump from one local mode to another.

3.4 Diagnosis

Diagnosing the convergence of the chain, including convergence to the stationary distribution, convergence of averages, and convergence to iid sampling, is generally not easy even for simple models. A common technique is the graphical method. The trace plot of the output of the chain is drawn in order to detect the lack of convergence. However, even this simple method is not feasible in our analysis, as it is hard to trace the tree graphically. Therefore, graphical exploration of the chain relies on some statistics of the tree, such as the tree size, likelihood and posterior probability. If any of these trace plots exhibits weird behavior, the

chain will not be considered to have converged. Similarly, in applying the other diagnosis methods, we only study some characteristics of the tree and not the tree itself. We will further explore the convergence diagnosis in Chapter 5 when the breast cancer data is examined.

3.5 Discussion

In this chapter, we designed a Metropolis-Hastings algorithm aiming to explore the posterior tree space. We started from three basic proposals, introduced by Chipman *et al.* (1998) and Denison *et al.* (1998), and introduced a new restructure proposal. A simulated dataset was constructed to illustrate each proposal and the pseudo-code was included as well.

The change move and grow/prune move propose a small change in the tree structure and/or the splitting rules, which leads to a small change in likelihood. The likelihood can be greatly changed if a parent-child pair on the top level of the tree is chosen in the swap move; however, in this case the swap move is not likely to be valid because it tends to produce empty leaves. So in most cases the parent-child pair that is picked in a valid swap move is on the lower level. This also leads to a small change in likelihood. The restructure move was actually inspired by the swap move, but in restructure move, the whole tree structure is abolished and a new tree structure is constructed based on the leaf configurations. Theoretically, it is not easy to compare an algorithm with basic moves only and the one with the basic moves and restructure move; however, as we briefly discussed in Section 3.4, we can study some characteristics of the output of the chain from the two algorithms and make comparisons on these statistics instead.

Benefiting from the hierarchical prior specification, the calculation of the ac-

ceptance probabilities for both basic moves and restructure move can be much simplified in most cases.

Variations of the restructure move were also discussed. The first variation includes variable selection in the restructure move; however, how to choose the candidate splitting variable set C and how to evaluate this variation is not clear. Observing that the likelihood is unchanged in a restructure move, we propose to shuffle the observations in the leaf configurations in the second variation. We described a random shuffle across the leaves in the text. However, there may exist a better strategy so that after the shuffle the leaf configurations is better "organized" for some purpose. Again, this is not clear in the text so far. We presented these two variations here solely for exploring potential new proposals in the Metropolis-Hastings algorithm. The evaluation of these moves will be left for future research.

In the beginning of this chapter, we discussed the restriction on the minimum number of data points in leaf. Such restriction leads to tree models with non-empty leaves; however, a tree with empty leaves may be of interest. For the purpose of balancing between computational simplicity and theoretical implication, we may want to allow for at most one empty leaf in the tree. This empty leaf represents a new group in prediction. When new observations come in and fall into the empty leaf, these observations come from distributions that are different from those in the other non-empty leaves. This new modeling will be left for future research.

Chapter 4

Example I: Synthetic Data

In the previous two chapters, we introduced a prior specification for the tree model and the Metropolis-Hastings algorithm for drawing from the posterior distribution. In Chapter 3, we designed a novel proposal, the restructure move, and discussed the basic moves. However we have been left an unanswered question. What is the difference between an algorithm with the restructure move and the one without the restructure move? This question is not easy to address theoretically. First, although the theory of the Metropolis-Hastings algorithm guarantees under mild condition that if one runs the algorithm long enough, the chain will converge to its limiting distribution. However, the stopping criterion is not usually available. Second, some exploratory methods are used to analyze the output of the chain. For tree models, the trace plots of some statistics of the tree are studied in order to assess the convergence; however, there seems to be no single statistic that can be used to represent the whole tree. Therefore the trace plots of these statistics reflect only part of the convergence problem.

We present simulation results for two examples to illustrate the convergence problem. A synthetic example will be discussed in this chapter. The synthetic

data set is designed to yield a posterior distribution with two distinct modes. This toy example is intended to illustrate how the restructure move generates better mixing Markov chains by enabling direct jumps between modes. The second, more complicated, example is discussed in Chapter 5, where we also introduce some exploratory methods specifically for tree models.

4.1 Data

We first discuss MCMC convergence and mixing properties using a synthetic data set having $p = 3$ predictors and $\mathcal{Y} = \mathbb{R}$.

We generated $n = 300$ sets of (x^1, x^2, x^3) , where $x_i^1 \sim \text{Unif}(0.1, 0.4)$ for $i = 1 \ldots 200$, $x_i^1 \sim \text{Unif}(0.6, 0.9)$ for $i = 201 \ldots 300$, $x_i^2 \sim \text{Unif}(0.1, 0.4)$ for $i = 1 \ldots 100$, $x_i^2 \sim \text{Unif}(0.6, 0.9)$ for $i = 101 \ldots 200$, $x_i^2 \sim \text{Unif}(0.1, 0.9)$ for $i = 201 \ldots 300$, $x_i^3 \sim \text{Unif}(0.6, 0.9)$ for $i = 1 \ldots 200$ and $x_i^3 \sim \text{Unif}(0.1, 0.4)$ for $i = 201 \ldots 300$. Given these predictor values, y values were simulated independently as

$$y = \begin{cases} 1 + \text{N}(0, 0.25) & \text{if } x^1 \leq 0.5 \text{ and } x^2 \leq 0.5, \\ 3 + \text{N}(0, 0.25) & \text{if } x^1 \leq 0.5 \text{ and } x^2 > 0.5, \\ 5 + \text{N}(0, 0.25) & \text{if } x^1 > 0.5. \end{cases} \tag{4.1}$$

The partition of observations with respect to x^1, x^2 and x^3 is shown in Figure 4.1, where the symbols refer to the three regions defined in equation (4.1). The mean values of y in these three regions are well separated. It is easy to check that x^1 and x^3 are the key predictors. The splitting variable for the root node has to be x^1 or x^3. If x^2 is chosen as the splitting variable for the root node, the posterior probability of the tree will be close to zero. Once the first splitting variable is chosen, the subsequent splitting variable has to be x^2. If only x^1 and x^3 were the

splitting variables, then the observations $1, 2, \ldots, 200$ are well-mixed in the region that $(x^1, x^3) \in [0, 0.5] \times [0, 0.5]$, as displayed in the top left corner in the middle panel of Figure 4.1.

The mean values of y in these three regions are well separated and it is easy to check that the two trees in Figure 4.2 are the only trees consistent with the regions.

4.2 Analysis

This analysis assumes a prior within-leaf model where the y_i's are independent and $y_i \sim \mathrm{N}(\mu_u, \sigma_u^2)$ when observation i is assigned to leaf u, i.e. $\theta_u = (\mu_u, \sigma_u^2)$. For this regression tree, we use the conjugate priors as specified in equation (2.20). More specifically, $\sigma_u^2 \sim \mathrm{IG}(0.5, 1.5)$ and $(\mu_u|\sigma_u^2) \sim \mathrm{N}(0, \sigma_u^2)$. The pinball prior has $\alpha(m) = 1 + \mathrm{Pois}(m - 1; 10)$ and $\beta(i|m) = 1 + \mathrm{Bin}(i - 1; m - 2, 0.5)$. The splitting variables are chosen uniformly via $\gamma(i) = \frac{1}{3}$, and the splitting thresholds come, for all k, from $\delta_k(\cdot) = \mathrm{N}(0.5, 2)$ truncated to $[0, 1]$ independently.

We initialize the Markov chain with the tree with a single root node, so at the beginning, no swap moves or prune moves can be proposed. The tree gradually grows to one of the trees in Figure 4.2, depending on which splitting variable is chosen in the root node. Suppose now the tree shown in the left panel of Figure 4.2 is the current tree (the splitting thresholds might be a little different). We are about to propose the following moves:

- *Change Proposal*: Regardless of which node among node 0 and node 1 is chosen in the change proposal, the associated splitting variable will likely be kept unchanged. For example, if node 1 is chosen and the proposed splitting

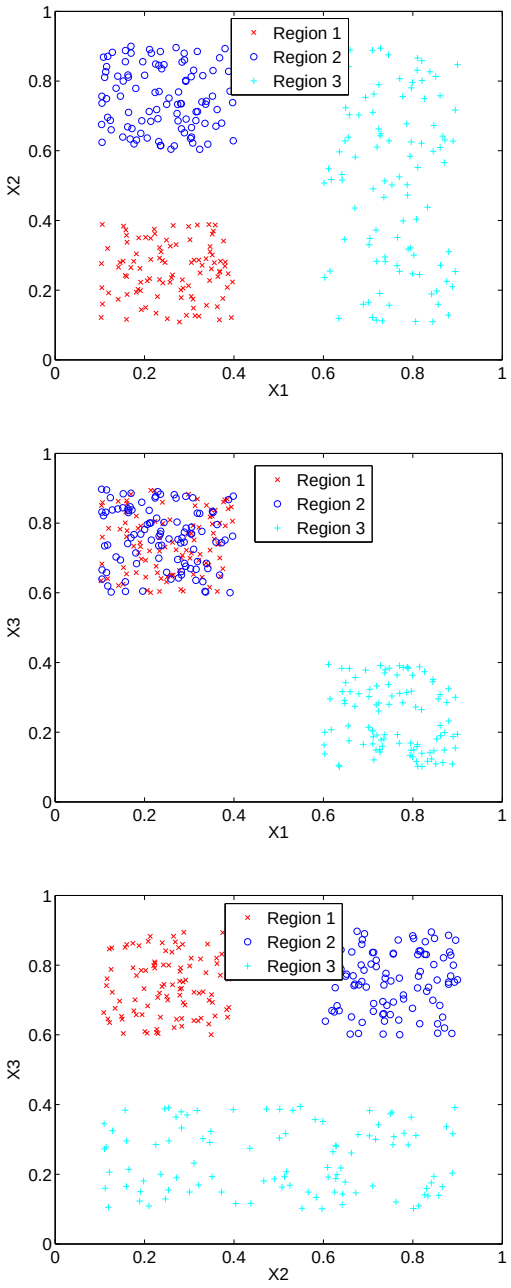


Figure 4.1: Simulated data example: Partition of observations with respect to x^1, x^2 and x^3. The symbols used refer to the three regions defined in equation (4.1). Cross: $x^1 \leq 0.5$ and $x^2 \leq 0.5$, circle: $x^1 \leq 0.5$ and $x^2 > 0.5$; plus: $x^1 > 0.5$.

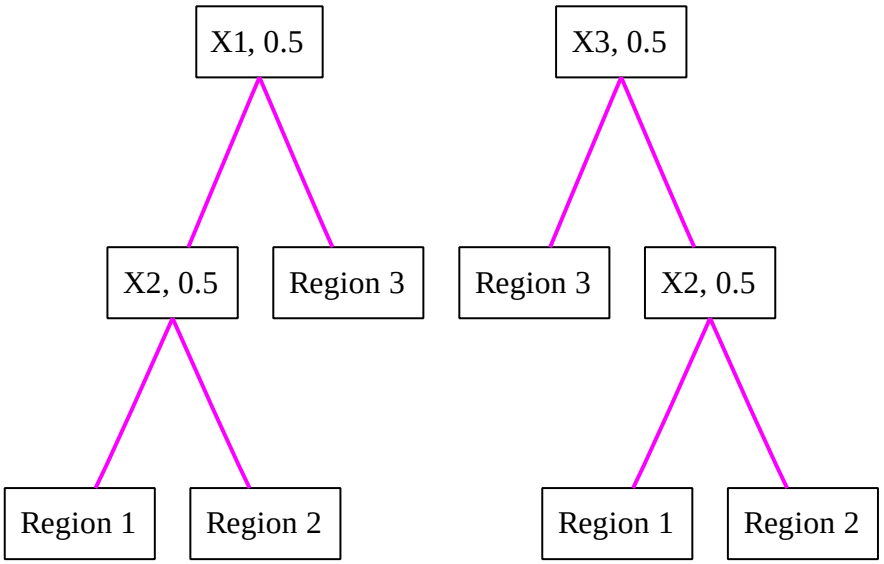


Figure 4.2: Two trees that are consistent with the three regions defined in equation (4.1). The leaf nodes of each tree mark the region they correspond to.

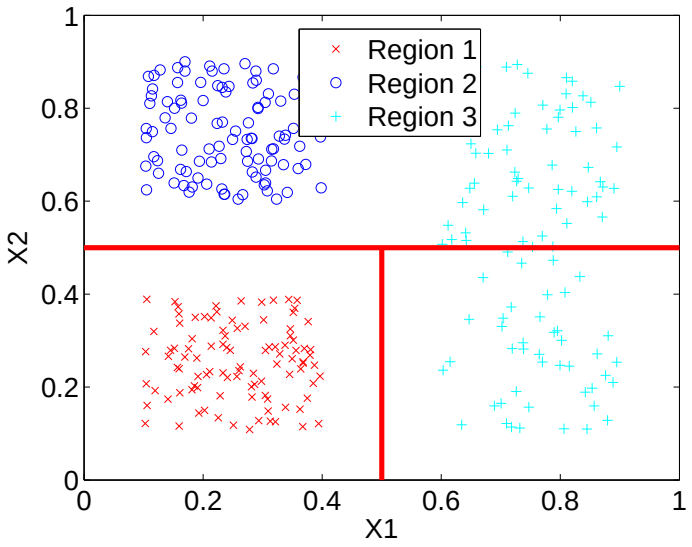


Figure 4.3: The partition induced by the new tree after the swap proposal.

variable is x^3, this new proposal will likely be rejected no matter what the corresponding splitting threshold is because in the space $\{x^1 < 0.5\}$ the observations are well-mixed and the resulting marginal likelihood is very small; if x^1 is the proposed splitting variable in the node 1, for any possible splitting threshold $0 < \tau < 0.5$ the new splitting rule will produce a new partition with a very small marginal likelihood (some of the observations in region 1 and region 2 are mixed together). Actually even if the same splitting variable x^1 is chosen with the proposed splitting threshold outside the interval $[0.4, 0.6]$, this splitting rule is still not very likely to be accepted.

- *Grow/prune Proposal:* If a prune move is going to be performed, the only possible way to proceed is to merge the leaves associated with region 1 and region 2 together. After the change, the observations in region 1 and region 2 will be mixed together. Since the mean values of y in different regions are well-separated, this will lead to a very small marginal likelihood and hence a very small acceptance probability. If a grow move is going to be performed, one of the three regions will be divided into two subregions. Compared to a prune move, such grow move is more likely to be accepted. However, since all the observations in one region come from the same normal distribution with a not very large variance, we still expect the acceptance probability to be very small.

- *Swap Proposal:* If a swap move is performed, the only possible pair that can be chosen is node 0 and node 1. After the swap, the resulting partition is shown in Figure 4.3, where the new partition is marked by the red solid lines. The observations in region 3 are then divided into two, one of which is mixed

with the observations in region 1. Therefore such a swap move produces a partition associated with a small marginal likelihood. This proposal is not likely to be accepted.

- *Restructure Proposal:* After extracting the leaf configurations from the tree model in the first step of the procedure as described in Section 3.3.1, all the possible splitting rules for the root node are shown in Figure 4.4. The first split cannot be made with x^2. The probabilities of choosing x^1 and x^3 as the first splitting variable are about the same. If x^3 is chosen, the splitting variable chosen in the next step must be x^2. This will produce a tree that looks like the one shown in the right panel of Figure 4.2, except that the splitting threshold might be different. Similarly, if x^1 is chosen as the first splitting variable, x^2 must be the second and then a tree which is very similar to the current is proposed. No matter which tree is proposed, the likelihood remains unchanged. Furthermore the difference between the prior probabilities is very small. Therefore the acceptance probability is very close to 1. That is, this proposal is likely to be accepted.

4.3 Comparison

In this section, with this synthetic example we will illustrate the difference between the algorithm with the restructure move and without the restructure move.

We run two algorithms: first, the MCMC using only the grow/prune, swap and change proposals; and second, including the basic restructure proposal. In the first case, we call an "iteration" a series of 60 change, 60 grow/prune and 60 swap moves; when including the restructure proposal, we take one iteration to

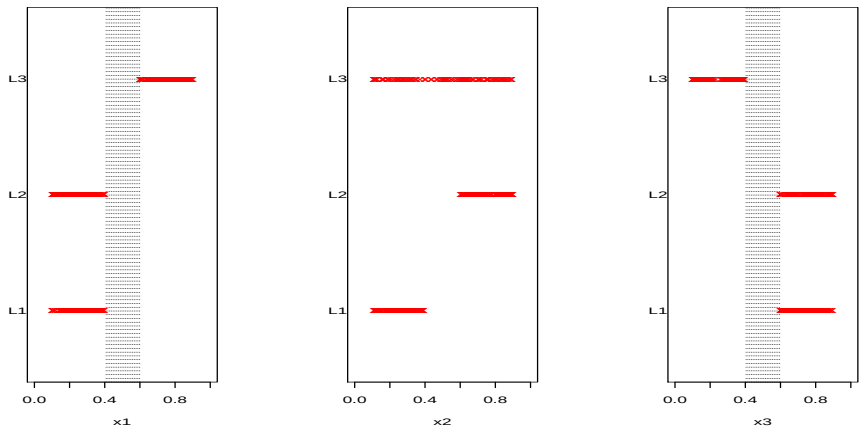


Figure 4.4: All the possible splitting rules for the root node of the tree shown in the left panel of Figure 4.2 for the restructure move.

mean 50 change moves, 50 grow/prune moves, 50 swap moves and 1 restructure move. With our implementation these two types of iterations require essentially the same amount of computing time.

We start the chain from the left tree shown in Figure 4.2 and run each of the algorithms for $8,000$ iterations. We use the last $4,000$ iterations to estimate, for each predictor, the posterior probability that this predictor is used as a splitting variable. In the tree samples produced by the algorithm without the restructure move, both x^1 and x^2 are used in all the 4000 tree samples while x^3 is used only twice. In fact, by examining the tree samples, we find that $3,997$ out of the $4,000$ samples have exactly the same structure and splitting variables as the starting tree. All these trees produce the same leaf configurations. The remaining three trees have one more split than the starting tree, which accounts for the small probability of accepting a grow move. Thus, the tree shown in the right panel in Figure 4.2 has never been visited. Since the implementations of the grow/prune, swap and change proposals are the same as those of Chipman *et al.* (1998) and

Denison *et al.* (1998), the algorithm in Chipman *et al.* (1998) and Denison *et al.* (1998) can hardly find the tree shown in the right panel in Figure 4.2. Restarts of the algorithm can help, but again one needs to weight all the generated trees. In contrast, using the algorithm with the restructure move very rapidly generates tree samples with x^1 and x^3 appearing at roughly equal frequencies (though of course not together in any tree). Across a number of repeat runs, by examining the tree samples we find that each of the two trees in Figure 4.2 is very frequently visited. For numerical detail, we ran the restructure MCMC for 4000 iterations and repeated this simulation 50 times. In each of these 50 runs, we recorded the Monte Carlo frequencies of occurrence of each of x^1 and x^3 as well as the Monte Carlo standard errors. Across the 50 runs, the median (range) of the probability that x^1 appears is 0.503 (0.488, 0.521) with Monte Carlo standard error of about 0.008; the corresponding figures for x^3 are 0.500 (0.488, 0.521) with Monte Carlo standard error of about 0.008.

As a supplementary analysis, we try a variation of the restructure move, where the candidate splitting variable set includes only the ones used in the current tree rather than the whole set of the predictors. This is a special case of equation (3.9) taking $\alpha = 0$. The results from the algorithm without the restructure move are similar. In contrast, using the algorithm with the variated restructure move, x^1 is used 1290 times, x^2 is used 4000 times and x^3 is used 2712 times. As the chain gets longer, the posterior probabilities that x^1 and x^3 appears in the tree move closer to $0.5 : 0.5$ but at a very slow rate. The Monte Carlo standard errors are much larger than what we see for the algorithm with standard restructure move. The problem is that when proposing a new tree in the restructure move from one of the trees in Figure 4.2, only the splitting variables already used will be in the candidate

set. In other words, the candidate set is either $\{x^1, x^2\}$ or $\{x^3, x^2\}$ in most cases. Therefore the current tree is the only choice in this variated restructure move. The only way for this algorithm to move from one local mode to the other is to first grow the tree to include all the predictors as the splitting variables and then propose a restructure move is proposed. The corresponding transition probability is much smaller than that in the algorithm with the standard restructure move. Therefore it takes a longer time for this algorithm to converge.

Finally, to check robustness with respect, particularly, to the key Poisson prior $\alpha(\cdot)$, we re-ran the analysis after replacing the y data with pure noise - a standard normal random sample. This produced a posterior distribution with $\Pr(m_0(T) = 1|\boldsymbol{y}) \approx 0.37$, compared to the prior probability of 0.05, supporting the view that the analysis is relatively robust to the assumed Poisson form; this has been borne out in other examples, including the real data example in Chapter 5.

4.4 Discussion

The necessity of this analysis might be questioned because, in this synthetic example, the two trees shown in Figure 4.2 are basically the same. They produce the same leaf configurations and thus one might think that the tree structure on the top is less relevant. However, this may not be the case in real data analysis. The synthetic example is used simply to illustrate the importance of the restructure move; however, in practice, the data are usually much more complicated. There could be more than two tree models, between which one can hardly find the direct connection, that gives equally important leaves configuration. In this case, we want to fully explore the tree space and hence the correlation between the predictors and the response. As the model gets more complicated, we cannot list

all the possible trees as we can or the synthetic data example. Different diagnostic tools can also be used, as we shall see in the Chapter 5.

Chapter 5

Example II: Breast Cancer Data

In Chapter 4, we explored the difference between the MCMC algorithm with the restructure move and the one without the restructure move through a synthetic data example. In that toy example, it is relatively easy to list all the local modes, therefore the convergence problem of the chain can be studied. In this chapter, we turn to a more complicated example where the convergence property is less clear. There seem to be no available methods for such analysis. In this case, we compare the restructure move with the other basic moves using exploratory methods, e.g. Kolmogorov-Smirnov test and the trace plots of some statistics.

In this chapter we also show how we use the posterior tree samples to make predictions and assess the importance of the predictors. Cross-validation will be used to evaluate prediction validity. An importance sampling method will be proposed to reduce the computational cost for leave-one-out cross-validation. In that analysis, we will find the assumption that the observations across leaves are independent samples to be very important.

5.1 Data description

We analyze the breast cancer data set used in Chipman *et al.* (1998). The data were obtained from the University of California, Irvine repository of machine learning databases (ftp://ftp.ics.uci.edu/pub/machine-learning-databases) and originated in Wolberg and Mangasarian (1990). The same data set is also used for a Bayesian CART model search, but without convergence of the MCMC algorithm, in Chipman *et al.* (1998). The data set has 9 cellular cancer characteristics, all ordered numeric variables, and the response is binary, indicating benign (0) and malign (1) tumors, i.e. $\mathcal{Y} = \{0, 1\}$. The original data set contains some missing values. We deleted cases with missing values and so use 683 of the original 699 observations.

Table 5.1 shows the correlations between predictors, ranging from 0.34 to 0.91. If the tree model is used, the highly correlated predictors are "exchangeable" as used in splitting rules. Therefore, we expect to find many trees equivalently important.

	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	*Name*
x^1	1.00	0.64	0.65	0.49	0.52	0.59	0.55	0.53	0.35	clump thickness
x^2	0.64	1.00	0.91	0.71	0.75	0.69	0.76	0.72	0.46	uniformity of cell size
x^3	0.65	0.91	1.00	0.69	0.72	0.71	0.74	0.72	0.44	uniformity of cell shape
x^4	0.49	0.71	0.69	1.00	0.59	0.67	0.67	0.60	0.42	marginal adhesion
x^5	0.52	0.75	0.72	0.59	1.00	0.59	0.62	0.63	0.48	single epithelial cell size
x^6	0.59	0.69	0.71	0.67	0.59	1.00	0.68	0.58	0.34	bare nuclei
x^7	0.55	0.76	0.74	0.67	0.62	0.68	1.00	0.67	0.35	bland chromatin
x^8	0.53	0.72	0.72	0.60	0.63	0.58	0.67	1.00	0.43	normal nucleoli
x^9	0.35	0.46	0.44	0.42	0.48	0.34	0.35	0.43	1.00	mitoses

Table 5.1: Breast cancer data example: Correlations between predictors.

In the original data, the predictors take integer values between 1 and 9, so we simply divide them by 10 so as to place all values of x^i in the unit value.

5.2 Exploratory data analysis

In Section 1.1, we studied this data set using a generalized linear model, where we found the assumption that logit(E(y)) is a linear function of x^1, x^2, ..., x^9 to be questionable. Meanwhile, in that model, we failed to find the significance of x^2, x^3, x^5 and x^9 in predicting the response, which could be easily validated by looking at the data. Thus, we were motivated to use tree model.

We first use the "greedy" algorithm as described in Chapter 1 before moving to a more complicated model. In R, this algorithm is implemented in *rpart* package. The complexity of the tree is controlled by the parameter cp (complexity parameter), which by default is 0.01. This means that if the decrease in overall lack of fit (e.g., RSS and Gini index) by a further split is smaller than cp, then the algorithm is stopped.

We call rpart with the default cp. The resulting tree is shown in Figure 5.1. The misclassification rate is $22/683 = 0.0322$ (which is even greater than the misclassification rate by the generalized linear model). Even if we run this algorithm with a smaller cp, e.g. 0.00001, we still get the same tree, indicating that the greedy algorithm is stuck in this tree. But is it the best tree in terms of misclassification rate? The answer is no. Figure 5.2 shows another tree with a smaller misclassification rate at $13/683 = 0.019$. The greedy algorithm fails to find this tree because a split that makes the most improvement at each step is not necessarily the right split in the best tree overall.

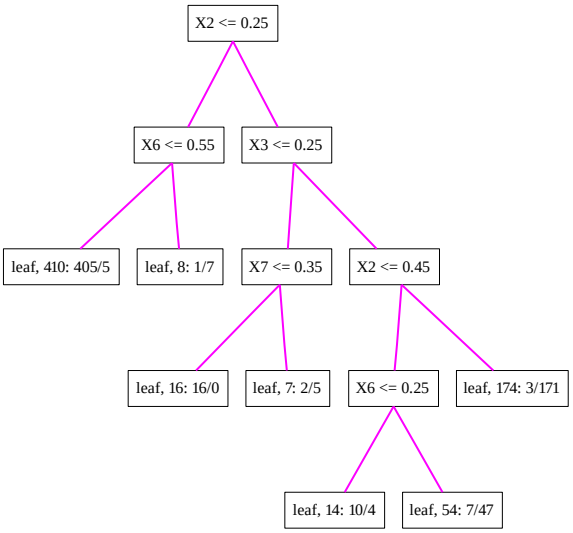


Figure 5.1: A tree produced by the greedy algorithm.

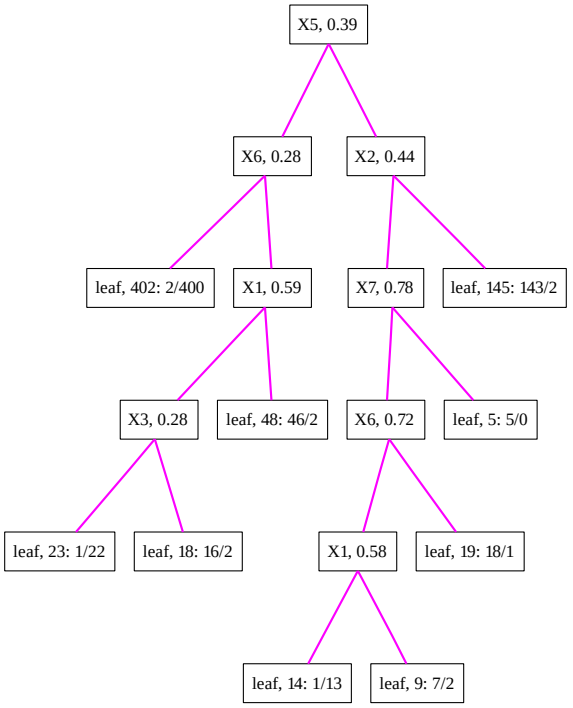


Figure 5.2: A tree with a smaller misclassification rate than the "greedy optimal" tree in figure 5.1.

5.3 Bayesian analysis

In this section, we will first briefly describe the model specification and the algorithm, and then we will introduce two exploratory methods for convergence diagnosis. Lastly we will discuss posterior inferences.

5.3.1 Model specification

For T we will use a similar prior distribution similar to that of Section 4.2. The pinball prior has $\alpha(m) = 1 + \text{Pois}(m-1; 10)$ and $\beta(i|m) = 1 + \text{Bin}(i-1; m-2, 0.5)$. For the breast cancer example we set $m = 10$. The splitting variables are chosen uniformly via $\gamma(i) = \frac{1}{9}$, and the splitting thresholds come, for all k, from $\delta_k(\cdot) = \text{Unif}[0,1]$ independently. Alternatively, the discrete uniform distribution on $\{0.1, 0.2, \ldots, 0.9\}$ ($\{1, 2, \ldots, 9\}$ if without transformation) could be used, since the predictors are all ordered numerical variables with only a few distinct values.

The within-leaf sampling model is Bernoulli with probability of malignancy θ_u in leaf u having independent uniform prior, as specified in equation (2.17), taking $a = b = 1$. MCMC analysis was performed using the same two algorithms as described in Section 4.3, with and without the restructure move.

5.3.2 Exploratory analysis of convergence

Convergence and mixing analysis are more difficult here than in the toy example of Chapter 4. There are numerous methods for exploring MCMC convergence (Cowles and Carlin, 1996; Brooks and Roberts, 1998). The basic idea is to simulate one very long run, assuming this chain is long enough to be considered having converged, and many short runs. By comparing the long run and the short runs,

we are able to explore convergence and mixing properties. As we never know how long we need to run the chain to achieve convergence for such complicated models, we want to study how long the chains need to be to have similar mixing properties as the very long chain.

Here we simulate one long and K short MCMC runs for a given algorithm. We initialize each run by sampling from the prior distribution. For each $i = 1, 2, \ldots$ we pick K realizations from the long run and one realization from each of the short runs. From the long run we use the realizations after $k \cdot i$ iterations for $k = 1, 2, \ldots, K$; whereas for the short runs, we use the realization after iteration number i. If the chain has converged before i iterations and the mixing is sufficient for two realizations i iterations apart to be independent, then the K realizations from the short runs and the K realizations from the long run are independent and all come from the same distribution. For any scalar function of the $2K$ realizations, we compute the Kolmogorov-Smirnov p-value (Conover, 1971) to provide some insight into whether this seems reasonable. The K-S procedure has the advantage of making no assumptions about the distribution of the data.

Before we use this K-S procedure to evaluate the algorithm with the restructure move, we will first illustrate this procedure with a very simple example. Suppose our target distribution is a mixture of normals, namely

$$y \sim 0.3\mathrm{N}(0,1) + 0.7\mathrm{N}(3,1). \tag{5.1}$$

Now we design two algorithms to sample from this target distribution. Algorithm A is an appropriate one. For both the long run and the short runs we sample directly from the target distribution. Algorithm B is an inappropriate one. Due to the starting point, algorithm B will be stuck in one of the local modes. Actually we are knowledgeable to sample from the target distribution so the algorithm B is

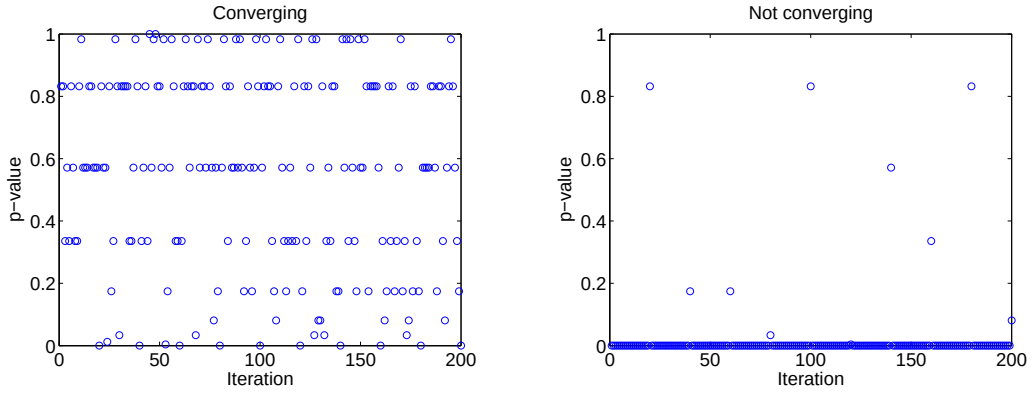


Figure 5.3: The Kolmogorov-Smirnov procedure for the simple example. The computed p-values in the left panel and the right panel corresponds to algorithm A and B respectively.

never used in reality. We mention that here just for illustration. Then we perform the K-S procedure for these two algorithms. If the null hypothesis is true, that the realizations from the long run and the realizations from the short runs come from the same distribution, the computed p-values will follow a uniform distribution over $[0, 1]$. The results for algorithm A and B are displayed in Figure 5.3. Clearly, the output of the chain by the fabricated algorithm B does not converge.

For the breast cancer data, Figure 5.4 shows the result when $K = 250$ and the scalar function is the log posterior density. When including the restructure move, the algorithm converges in fewer than 500 iterations; whereas, without the restructure move convergence takes more than $4,000$ iterations.[1] We have also generated this type of convergence diagnostics plot for various other scalar functions including, among others, the log integrated likelihood, tree size and the number of times a particular predictor variable is used as a splitting variable in the tree. The convergence differences between the two algorithms are more or less striking dependent on which scalar function is used. The case shown in Figure 5.4

[1]With our implementation in early 2005, 500 iterations for this data set took about 6 seconds on an Intel Xeon 3.0Ghz with 2Gb memory.

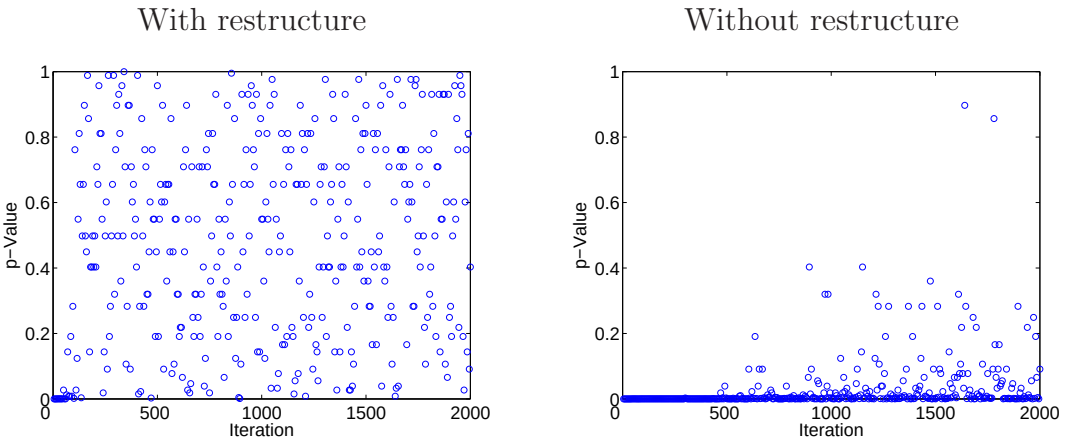


Figure 5.4: Breast cancer data example: P-values of K-S statistics for samples from algorithm with and without restructure move.

indicates the slowest convergence of the non-restructure move approach for this example.

Notice this K-S procedure is not a sufficient tool. In the right panel of Figure 5.4, most of the p-values are very small, which by K-S test means that the distribution of the log posterior density from the two algorithms differ significantly from each other; however, even if every K-S procedure for all possible scalar functions indicates convergence, it is not necessary that the chain has converged. The method is very useful and suggestive but not a "proof" of convergence.

An alternative way to diagnose the convergence is by looking at the trace plots of some statistics. The topology of the tree space is very complex and to get a clear picture of how the two algorithms work is very difficult; however, some insight can be gained from Figure 5.5. This shows, for ten runs with each of the two algorithms, trace plots of the log integrated likelihood and the number of leaves. Each simulation is for 1000 iterations, of which the last 500 iterations are shown in the plots. Vertical dashed lines mark the start of new runs. Again we initiate

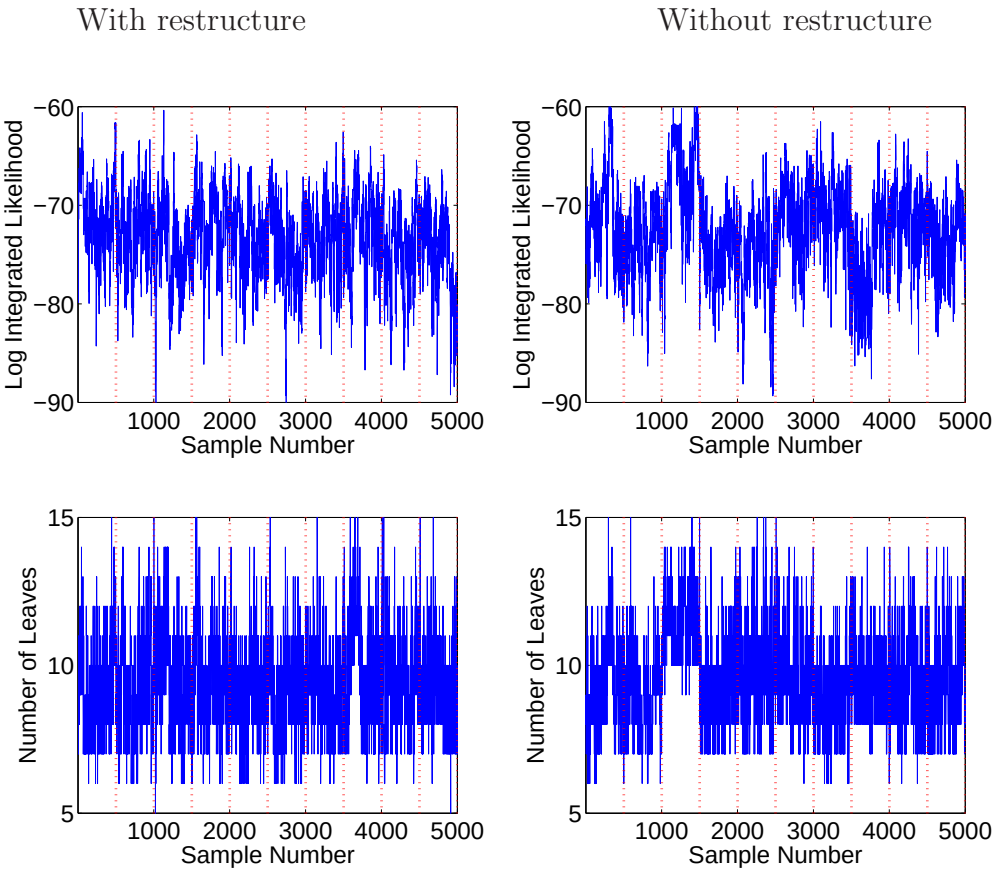


Figure 5.5: Breast cancer data example: Trace plots of log integrated likelihood (upper row) and number of leaves (lower row) for ten runs of the algorithm with the restructure proposal (right column) and without the restructure proposal (left column). In each simulation the algorithms is run for $1,000$ iterations, of which only the last 500 are shown in the figure. The start of a new run is indicated by a vertical dashed line.

each run by sampling from the prior distribution. In the right column of Figure 5.5 we note how some of the runs from the algorithm without the restructure move seem to get trapped in local modes for the whole 500-iteration period. In particular, this is true for runs 3, 7 and 8. For example, in run 3 most of the trees visited have the number of leaves larger than 10, and we can see the corresponding log integrated likelihood first decreases to about -78 and then increases to about -60, which is an odd pattern compared to the other runs. In the trace plots for the algorithm with the restructure move, no such effects were encountered. Therefore, the use of the restructure move seems to improve the convergence and mixing properties of the Markov chain by preventing it from being trapped in local modes over a long period. In turn this removes the need for restarts and the ad hoc weighting discussed in Chipman *et al.* (1998).

5.3.3 Posterior inferences

Exploration of posterior inferences involves inspecting trees generated from the sampler and histograms of posterior samples of key quantities of interest, such as the tree size $m(T)$ and the malign/benign mixing fraction that will be introduced below.

In Figure 5.6 we show two randomly selected trees from the sampler. It can be seen that, as we would expect, x^2 is the key splitting variable. For example, in the root node of the tree on the top and node 2 in the bottom tree, the splitting rule is about x^2 on around 0.45, which results in a similar subtree in the right child. The splitting variable in both the right child is x^4. In that subtree, 172 out of 175 samples indicate malignancy. This shall be further studied to infer the relationship between the malignancy and the uniformity of cell size (x^2) and the

marginal adhesion (x^4).

In the left panel of Figure 5.7, a histogram of the tree size $m(T)$ is shown. We can see that the data give little support to trees with more than 15 terminal nodes, and the posterior mode of the tree size is approximately 9. Recall that the mean of the prior distribution for tree size is also 9, so we need to check if the prior dominates the data. We have tried pinball priors encouraging smaller ($m = 5$) and larger ($m = 15$) trees, and the results are very insensitive to this choice.

The middle panel in Figure 5.7 gives the posterior distribution for the log integrated likelihood. Most of the probability mass lies in the interval $(-81, -65)$, but the run visited trees with log integrated likelihood values up to -60. Chipman *et al.* (1998) report log integrated likelihood values up to -62.2, with most lying below -64. As our run was longer than theirs, this is as one should expect. In addition to tree size we also consider a statistic describing another property of the tree. Given a simulated tree, we first compute the posterior mean of θ in each leaf by

$$\begin{aligned} & \mathrm{E}(\theta_u|\{y\}_u, \mathsf{T}) \\ = & \int \theta_u p(\theta_u|\{y\}_u, \mathsf{T}) \\ = & \int \theta_u \frac{1}{\mathrm{B}(\sum y_i + a, n_u - \sum y_i + b)} \theta_u^{\sum y_i + a - 1} (1-\theta_u)^{n_u - \sum y_i + b - 1} d\theta_u \\ = & \frac{\sum y_i + a}{n_u + a + b} \end{aligned} \tag{5.2}$$

where $\sum y_i$ is the sum of y_i in leaf u, that is $\sum_{i \in \mathcal{N}_{\mathsf{T}}(u,\mathcal{I})} y_i$, and n_u is the number of observations in leaf u. If the posterior mean of θ is larger than 0.5, subjects in this leaf have higher probability for being malign. Thus we classify such leaves as malign and the remaining leaves as benign. We consider the fraction of the sub-

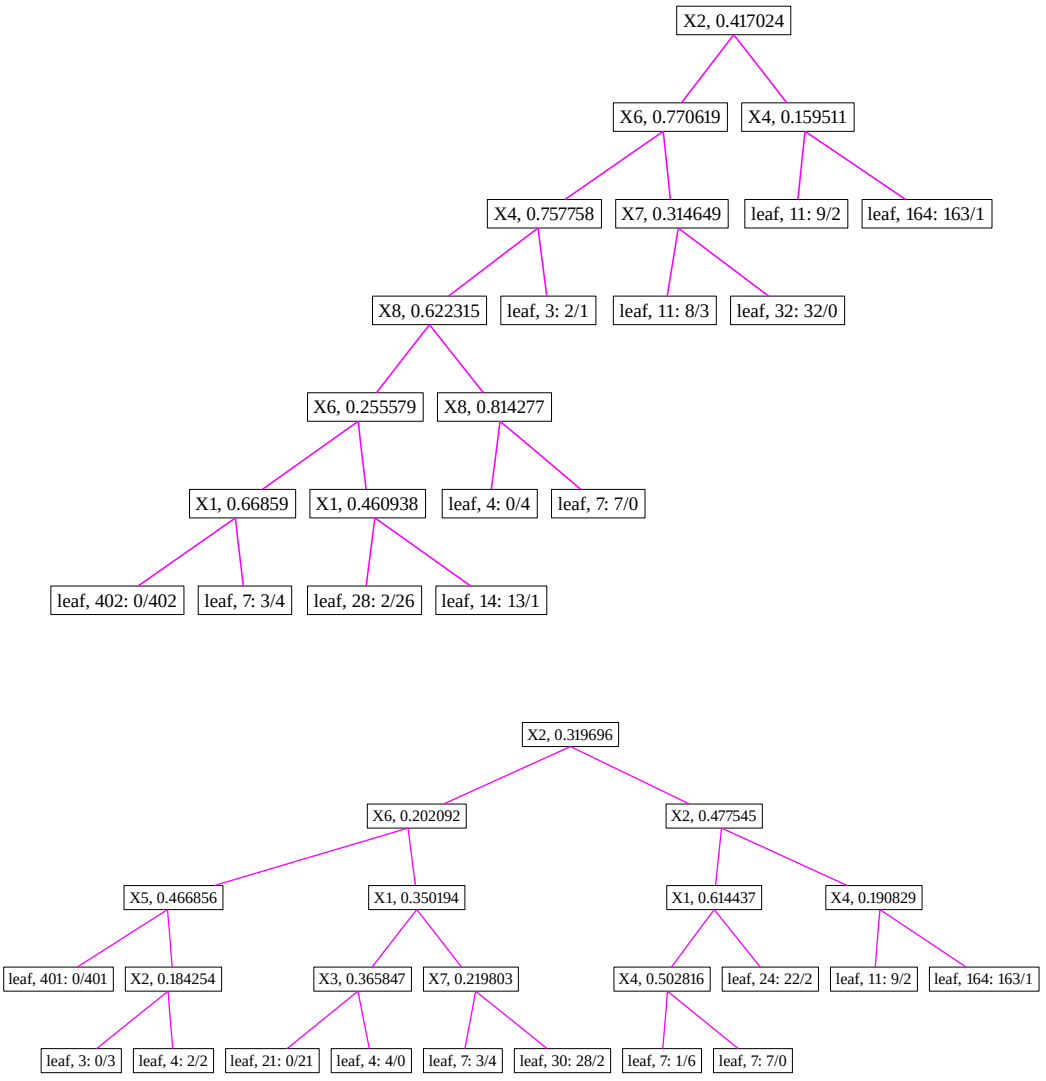


Figure 5.6: Breast cancer data example: Two trees chosen at random from the posterior distribution.

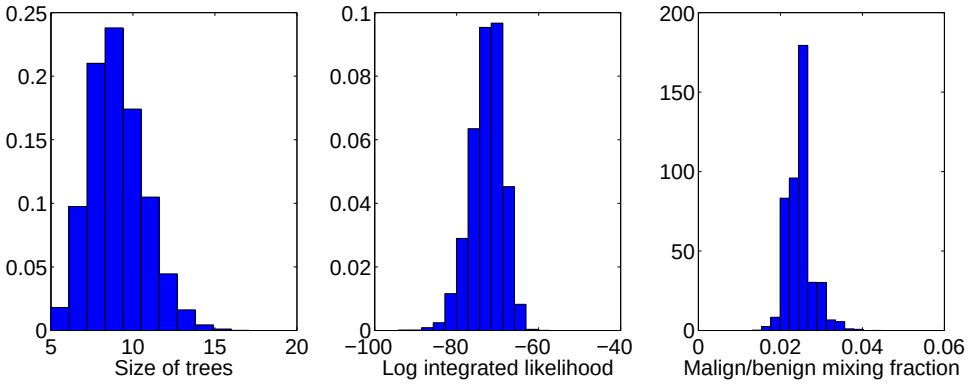


Figure 5.7: Breast cancer data example: Left: Posterior for tree size $m(T)$; Center: Posterior for log integrated likelihood; Right: Summary histogram of Malign/Benign mixing fraction.

jects where subject status differs from leaf status. We call this the "malign/benign mixing fraction". A low value indicates pure malign and benign leaves, whereas a high value results from more mixed leaves. The right panel in Figure 5.7 shows the histogram for this statistic. The mean value is 0.0248, whereas the smallest value found in the run is 0.0132. Thus, only a small degree of mixing occurs in the leaves. As a comparison, the tree found by *rpart* package in R has a mixing fraction 0.0322, which is among the worst cases in our run. The "mis-classification rates" reported in Chipman *et al.* (1998) correspond to mixing fraction as low as 0.016, though they do not report any corresponding mean values.

5.3.4 Importance of the predictors

In addition to looking at each posterior tree sample, we also summarize the posterior to assess relevance of each of the predictor variables. For each predictor we estimate the posterior probability that this predictor is used as a splitting variable (at least once), with results given in Table 5.2. Most of the nine predictors have

	x^1	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	*Name*
x^1	**0.98**	0.96	0.61	0.47	0.51	0.98	0.38	0.60	0.26	clump thickness
x^2	0.96	**0.98**	0.60	0.47	0.50	0.98	0.37	0.60	0.26	uniformity of cell size
x^3	0.61	0.60	**0.62**	0.28	0.32	0.62	0.27	0.35	0.17	uniformity of cell shape
x^4	0.47	0.47	0.28	**0.48**	0.24	0.48	0.15	0.28	0.14	marginal adhesion
x^5	0.51	0.50	0.32	0.24	**0.52**	0.52	0.19	0.22	0.15	single epithelial cell size
x^6	0.98	0.98	0.62	0.48	0.52	**1.00**	0.39	0.61	0.27	bare nuclei
x^7	0.38	0.37	0.27	0.15	0.19	0.39	**0.39**	0.24	0.10	bland chromatin
x^8	0.60	0.60	0.35	0.28	0.22	0.61	0.24	**0.61**	0.15	normal nucleoli
x^9	0.26	0.26	0.17	0.14	0.15	0.27	0.10	0.15	**0.27**	mitoses

Table 5.2: Breast cancer data example: Posterior pairwise and marginal (on diagonal) model inclusion probabilities for the nine predictors.

predictive relevance, which – given that one or a few may be truly predictive – is not surprising in view of the collinearity observed (Recall that the correlation between the predictors ranges from 0.34 to 0.91.). The table provides the Monte Carlo estimates of posterior co-inclusion probabilities for pairs of variables as well as the marginal probabilities of inclusion for each.

5.4 Prediction and cross-validation

5.4.1 Prediction

The predictive probability of a new observation y^* associated with $\boldsymbol{x}^*$ being malign is computed by

$$P(y^* = 1|\boldsymbol{y}) \quad = \quad \int_{\mathsf{T}} P(y^* = 1|\boldsymbol{y}, \mathsf{T}) p(\mathsf{T}|\boldsymbol{y}) d\mathsf{T} \tag{5.3}$$

Given the posterior tree samples T^i, the integral in equation (5.3) is approximated by the sum

$$\sum_{i=1}^{N} P(y^* = 1|\boldsymbol{y}, \mathsf{T}^i) p(\mathsf{T}^i|\boldsymbol{y}) \tag{5.4}$$

where the probability $P(y^* = 1|\boldsymbol{y}, \mathsf{T}^i)$ is computed by

$$\begin{aligned} P(y^* = 1|\boldsymbol{y}, \mathsf{T}^i) &= P(y^* = 1|\{y\}_u, \mathsf{T}^i) \\ &= \int_{\theta_u} P(y^* = 1|\theta_u, \{y\}_u, \mathsf{T}^i) p(\theta_u|\{y\}_u, \mathsf{T}^i) d\theta_u \\ &= \int_{\theta_u} \theta_u p(\theta_u|\{y\}_u, \mathsf{T}^i) d\theta_u \\ &= \mathrm{E}(\theta_u|\{y\}_u, \mathsf{T}^i) \end{aligned} \tag{5.5}$$

where u is the leaf node the new observation falls into. From equation (5.5), we know that given the posterior tree sample, the predictive probability of the new observation being malign is the posterior mean of θ in the leaf that the new observation falls into.

We still need to show why the first step in equation (5.5) holds. The first equation means that the probability of a new observation being 1 given the data and the tree sample is equal to the probability of this observation being 1 given only part of the data and the tree sample, that is, we only need to look at the observations that fall into the same leaf as the new observation. Similar to the notation $\{y\}_u$, $\{y\}_{-u}$ is the set of the observation falling into the other leaves than leaf node u. We first show that given $\{y\}_u$ and T^i, y^* and $\{y\}_{-u}$ are independent. This follows from

$$\begin{aligned} & P(y^*, \{y\}_{-u}|\{y\}_u, \mathsf{T}^i) \\ = & \int P(y^*, \{y\}_{-u}|\{y\}_u, \boldsymbol{\theta}, \mathsf{T}^i) p(\boldsymbol{\theta}|\{y\}_u, \mathsf{T}^i) d\boldsymbol{\theta} \\ = & \int P(y^*, \{y\}_{-u}|\{y\}_u, \boldsymbol{\theta}_u, \boldsymbol{\theta}_{-u}, \mathsf{T}^i) p(\boldsymbol{\theta}_u|\{y\}_u, \mathsf{T}^i) p(\boldsymbol{\theta}_{-u}|\mathsf{T}^i) d\boldsymbol{\theta} \\ = & \int P(y^*|\{y\}_u, \boldsymbol{\theta}_u, \mathsf{T}^i) P(\{y\}_{-u}|\{y\}_u, \boldsymbol{\theta}_{-u}\mathsf{T}^i) p(\boldsymbol{\theta}_u|\{y\}_u, \mathsf{T}^i) p(\boldsymbol{\theta}_{-u}|\mathsf{T}^i) d\boldsymbol{\theta} \\ = & P(y^*|\{y\}_u, \mathsf{T}^i) P(\{y\}_{-u}|\{y\}_u, \mathsf{T}^i). \end{aligned}$$

With this conditional independence, we have

$$\begin{aligned} & P(y^* = 1|\boldsymbol{y}, \mathsf{T}^i) \\ = \; & P(y^* = 1|\{y\}_u, \{y\}_{-u}\mathsf{T}^i) \\ = \; & P(y^* = 1, \{y\}_{-u}|\{y\}_u\mathsf{T}^i)/P(\{y\}_{-u}|\{y\}_u\mathsf{T}^i) \\ = \; & P(y^* = 1|\{y\}_u, \mathsf{T}^i)P(\{y\}_{-u}|\{y\}_u, \mathsf{T}^i)/P(\{y\}_{-u}|\{y\}_u\mathsf{T}^i) \\ = \; & P(y^* = 1|\{y\}_u, \mathsf{T}^i), \end{aligned} \tag{5.6}$$

so that the first equal sign in equation (5.5) holds.

Turning now to prediction, the entire analysis was repeated using a randomly selected 342 observations as training data, and producing $10,000$ posterior tree samples. The posterior was used to make out-of-sample predictions on the remaining 341 test observations. Figure 5.8 displays the results. With a simple threshold at 0.5 on the Monte Carlo estimates of the implied predictive probabilities, averaged over all $10,000$ trees, the raw prediction error is 13 out of 341.

By comparison, use of the greedy algorithm rpart in R (with default parameter settings) led to 23 misclassifications in this 50:50 hold-out prediction assessment. Modifying rpart to permit a smaller *cp* value in splitting the tree led to 18 misclassified cases.

5.4.2 Cross-validation

To further assess prediction validity we also ran a full ten-fold cross-validation analysis, each time randomly dividing the data set into ten parts with each part consisting of ten percent of the benign observations and ten percent of the malign observations. We reserve one part of the data, use the remaining observations to generate posterior tree samples, and then predict the held-out samples. We repeated this ten-fold cross-validation ten times, which resulted in an average

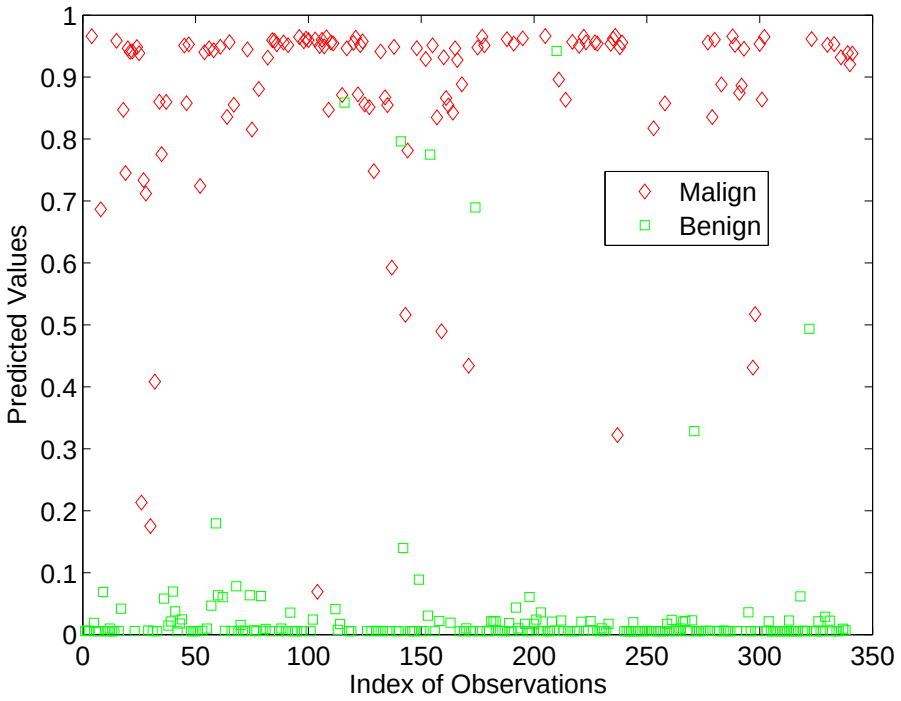


Figure 5.8: Breast cancer example: Predictive probabilities for the 50% sample test-set held out to assess predictive accuracy of the tree model fitted to the 50% training sample. The squares and diamonds represent subjects with benign and malign recurrence, respectively.

misclassification rate of 3.9%, calculated by

$$\begin{aligned} M &= \sum_{i=1}^{10} M(\boldsymbol{y}_{H_i}|\boldsymbol{y}_{-H_i})/10 \\ &= \sum_{i=1}^{10} \int M(\boldsymbol{y}_{H_i}|\boldsymbol{y}_{-H_i}, \mathsf{T}) p(\mathsf{T}|\boldsymbol{y}_{-H_i})/10 d\mathsf{T} \end{aligned} \tag{5.7}$$

where the function $M(\cdot)$ is the misclassification rate and H_i, $i = 1, 2, \ldots, 10$ index the ten portions of the data set. The integral in equation (5.7) is approximated by MCMC estimation. We need to run 10 MCMC chains in order to evaluate equation (5.7). This evaluation will be computationally expensive if we want to do leave one out cross-validation, as we will have to run n MCMC, where n is the number of observations in the data set.

In leave-one-out cross-validation, we are interested in computing

$$P(Y_i|\boldsymbol{y}_{-i}) = \int P(Y_i|\boldsymbol{y}_{-i}, \mathsf{T}) p(\mathsf{T}|\boldsymbol{y}_{-i}) d\mathsf{T} \tag{5.8}$$

where $\boldsymbol{y}_{-i}$ is the set of observations excluding the ith observation. Note that this notation is different from $\{y\}_u$ and $\{y\}_{-u}$. Instead of running MCMC for every i, we will use the idea from the *Importance Sampling* method. The Importance Sampling method requires independent samples, where in MCMC context this requirement cannot be satisfied; however, the convergence of average is still much similar.

Suppose the quantity of interest is

$$\int_{\mathcal{X}} h(x) f(x) dx \tag{5.9}$$

One alternative to direct sampling from f for the evaluation of equation (5.9) is to use importance sampling, defined as

$$\int_{\mathcal{X}} h(x) \frac{f(x)}{g(x)} g(x) dx \tag{5.10}$$

which is approximated by

$$\frac{1}{m}\sum_{j=1}^{m}\frac{f(X_j)}{g(X_j)}h(X_j) \tag{5.11}$$

where $X_1, X_2, \ldots, X_m$ are sampled from a given instrumental distribution g, and the integral in equation (5.9) is approximated by a weighted sum. Function h is evaluated at the points sampled from g and then weighted by $w = f/g$. This estimator will converge to the quantity shown in equation (5.9) as long as the support of g contains that of f. An alternative to approximating equation (5.10) is

$$\frac{\sum_{j=1}^{m} h(X_j)w(X_j)}{\sum_{j=1}^{m} w(X_j)}. \tag{5.12}$$

This estimator also converges to $\int h(x)f(x)dx$ by the Strong Law of Large Numbers and it has an advantage that we only need to calculate the weight w up to a constant; that is, instead of calculating the exact weight

$$w(X_j) = \frac{f(X_j)}{g(X_j)}$$

we can calculate

$$\tilde{w}(X_j) = cw(X_j) \propto \frac{f(X_j)}{g(X_j)}.$$

This method is attractive because the same sample from g can be used repeatedly. Suppose now we have n integrals to evaluate:

$$\begin{array}{c} \int_{\mathcal{X}} h_1(x)f_1(x)dx, \\ \int_{\mathcal{X}} h_2(x)f_2(x)dx, \\ \vdots \\ \int_{\mathcal{X}} h_n(x)f_n(x)dx. \end{array} \tag{5.13}$$

We can then use the same instrumental distribution g for approximating all the above integrals as long as the support of this function g contains those of functions $f_1, f_2, \ldots, f_n$. Therefore we need to generate samples from g only once. This method is efficient when $f_1, f_2, \ldots, f_n$ are close to g.

Now we consider applying the Importance Sampling method to leave one out cross-validation. We first show a general case of equation (5.5), where the leaf node distribution is not limited to binomial and the quantity of interest can be some other function. We have

$$\begin{aligned} P(Y|\boldsymbol{y}, \mathsf{T}) &= P(Y|\{y\}_u, \mathsf{T}) \\ &= \int_{\boldsymbol{\theta}_u} P(Y|\{y\}_u, \boldsymbol{\theta}_u, \mathsf{T}) p(\boldsymbol{\theta}_u|\{y\}_u, \mathsf{T}) d\boldsymbol{\theta}_u \\ &= \int_{\boldsymbol{\theta}_u} P(Y|\boldsymbol{\theta}_u, \mathsf{T}) p(\boldsymbol{\theta}_u|\{y\}_u, \mathsf{T}) d\boldsymbol{\theta}_u \\ &= \int_{\boldsymbol{\theta}_u} P(Y|\boldsymbol{\theta}_u, \mathsf{T}) \frac{p(\boldsymbol{\theta}_u|\mathsf{T}) p(\{y\}_u|\boldsymbol{\theta}_u, \mathsf{T})}{p(\{y\}_u|\mathsf{T})} d\boldsymbol{\theta}_u. \end{aligned} \tag{5.14}$$

The first equal sign holds because for any u the distribution of $\{y\}_u$, falling into leaf node u, given $\boldsymbol{\theta}_u$ and T does not depend on the other parameters $\boldsymbol{\theta}_{-u}$; that is

$$P(\{y\}_u|\mathsf{T}, \boldsymbol{\theta}) = P(\{y\}_u|\mathsf{T}, \boldsymbol{\theta}_u). \tag{5.15}$$

The proof of equation (5.15) is similar to equation (5.6). Replacing Y and $\boldsymbol{y}$ in equation (5.14) with Y_i an $\boldsymbol{y}_{-i}$, we have

$$P(Y_i|\boldsymbol{y}_{-i}, \mathsf{T}) = \int_{\boldsymbol{\theta}_u} P(Y_i|\boldsymbol{\theta}_u, \mathsf{T}) \frac{p(\boldsymbol{\theta}_u|\mathsf{T}) p(\{y\}_{u,-i}|\boldsymbol{\theta}_u, \mathsf{T})}{p(\{y\}_{u,-i}|\mathsf{T})} d\boldsymbol{\theta}_u. \tag{5.16}$$

where $\{y\}_{u,-i}$ is the set of observations falling into leaf node u except the ith observation.

We are interested in evaluating equation (5.8) for $i = 1, 2, \ldots, n$. Since sampling from $p(\mathsf{T}|\boldsymbol{y}_{-i})$ is computationally expensive, the standard Monte Carlo esti-

mation of equation (5.8) is not preferred. In importance sampling, our instrumental distribution is $p(\mathsf{T}|\boldsymbol{y})$, so the importance sampling representation of equation (5.8) is

$$P(Y_i|\boldsymbol{y}_{-i}) = \int P(Y_i|\boldsymbol{y}_{-i},\mathsf{T})\frac{p(\mathsf{T}|\boldsymbol{y}_{-i})}{p(\mathsf{T}|\boldsymbol{y})}p(\mathsf{T}|\boldsymbol{y})d\mathsf{T} \tag{5.17}$$

Thus we only need to run the MCMC algorithm for the whole data set without leaving out any observation and generate samples to evaluate $P(Y_i|\boldsymbol{y}_{-i})$. It is reasonable to believe that the distribution of $\mathsf{T}|\boldsymbol{y}$ is close to the distribution of $\mathsf{T}|\boldsymbol{y}_{-i}$ for any i, especially when the number of observations is relatively large and thus leaving out one observation will have a small effect on the posterior distribution.

We generate samples $\mathsf{T}_1, \mathsf{T}_2, \ldots, \mathsf{T}_N$ from $p(\mathsf{T}|\boldsymbol{y})$ only once and will be able to evaluate n quantities $P(Y_i|\boldsymbol{y}_{-i})$, $i = 1, 2, \ldots, n$. For each i, we calculate

$$\begin{aligned} h(\mathsf{T}_j) &= P(Y_i|\boldsymbol{y}_{-i},\mathsf{T}_j) \\ &= \int_{\boldsymbol{\theta}_u} P(Y_i|\boldsymbol{\theta}_u,\mathsf{T}_j)\frac{p(\boldsymbol{\theta}_u|\mathsf{T}_j)p(\{y\}_{u,-i}|\boldsymbol{\theta}_u,\mathsf{T}_j)}{p(\{y\}_{u,-i}|\mathsf{T}_j)}d\boldsymbol{\theta}_u \end{aligned} \tag{5.18}$$

for $j = 1, 2, \ldots, N$. The second line is given by equation (5.16). The weight $w(\mathsf{T}_j) = \dfrac{f(\mathsf{T}_j)}{g(\mathsf{T}_j)} = \dfrac{p(\mathsf{T}_j|\boldsymbol{y}_{-i})}{p(\mathsf{T}_j|\boldsymbol{y})}$ is computed by

$$\begin{aligned} w(\mathsf{T}_j) &= \frac{p(\mathsf{T}_j|\boldsymbol{y}_{-i})}{p(\mathsf{T}_j|\boldsymbol{y})} \\ &= \frac{p(\boldsymbol{y}_{-i}|\mathsf{T}_j)p(\mathsf{T}_j)/p(\boldsymbol{y}_{-i})}{p(\boldsymbol{y}|\mathsf{T}_j)p(\mathsf{T}_j)/p(\boldsymbol{y})} \\ &= \frac{p(\boldsymbol{y}_{-i}|\mathsf{T}_j)}{p(\boldsymbol{y}|\mathsf{T}_j)}\frac{p(\boldsymbol{y})}{p(\boldsymbol{y}_{-i})} \\ &\propto \frac{p(\{y\}_{u,-i}|\mathsf{T}_j)}{p(\{y\}_u|\mathsf{T}_j)}. \end{aligned} \tag{5.19}$$

The last step holds because first, from equation (5.15), we have the conditional

independence of the marginal likelihood given as

$$\begin{aligned} p(\boldsymbol{y}|\mathsf{T}) &= \int p(\boldsymbol{y}|\mathsf{T},\boldsymbol{\theta})p(\boldsymbol{\theta}|\mathsf{T})d\boldsymbol{\theta} \\ &= \int p(\{y\}_u,\{y\}_{-u}|\mathsf{T},\boldsymbol{\theta}_u,\boldsymbol{\theta}_{-u})p(\boldsymbol{\theta}_u,\boldsymbol{\theta}_{-u}|\mathsf{T})d\boldsymbol{\theta} \\ &= \int p(\{y\}_u|\mathsf{T},\boldsymbol{\theta}_u)p(\boldsymbol{\theta}_u|\mathsf{T})d\boldsymbol{\theta}_u \int p(\{y\}_{-u}|\mathsf{T},\boldsymbol{\theta}_{-u})p(\boldsymbol{\theta}_{-u}|\mathsf{T})d\boldsymbol{\theta}_{-u} \\ &= p(\{y\}_u|\mathsf{T})p(\{y\}_{-u}|\mathsf{T}). \end{aligned} \tag{5.20}$$

Then

$$\begin{aligned} \frac{p(\boldsymbol{y}_{-i}|\mathsf{T}_j)}{p(\boldsymbol{y}|\mathsf{T}_j)} &= \frac{p(\{y\}_{-u}|\mathsf{T}_j)p(\{y\}_{u,-i}|\mathsf{T}_j)}{p(\{y\}_{-u}|\mathsf{T}_j)p(\{y\}_u|\mathsf{T}_j)} \\ &= \frac{p(\{y\}_{u,-i}|\mathsf{T}_j)}{p(\{y\}_u|\mathsf{T}_j)}. \end{aligned} \tag{5.21}$$

The weight for posterior sample T_j is proportional to the ratio of the marginal likelihood of the observations in leaf node u, where the omitted observation should be, over the marginal likelihood of all the observations, including the held out observation, in leaf node u. Note that this ratio can not be further simplified to $1/p(y_i|\mathsf{T}_j)$ because the observations in the same leaf are not marginally independent given the tree structure T.

Note that the calculation of the weight in equation (5.21) may be questionable when the i^{th} observation falls into a leaf with only one observation, y_i. In this case $\{y\}_{u,-i}$ is actually an empty set. In our analysis, when this happens we set $p(\{y\}_{u,-i}|\mathsf{T}_j)$ to zero. This assertion is reasonable because when given such a tree model T_j, we observe as if only one data point. It makes no sense to predict y_i with the "other" data, which actually is an empty set. Therefore, the weight for this case is zero. An alternative is to specify a minimum leaf size (> 1) in our model, so that in every leaf node in the tree, there are at least two observations.

Now we have $h(\mathsf{T}_j)$ given in equation (5.18) and weights $w(\mathsf{T}_j)$ given in equa-

tion (5.19). The approximation of $P(Y_i|\boldsymbol{y}_{-i})$ is given by

$$
\begin{aligned}
& \frac{\sum_{j=1}^{N} h(\mathsf{T}_j) w(\mathsf{T}_j)}{\sum_{j=1}^{N} w(\mathsf{T}_j)} \\
= & \frac{\sum_{j=1}^{N} \int_{\boldsymbol{\theta}_u} P(Y_i|\boldsymbol{\theta}_u, \mathsf{T}_j) \frac{p(\boldsymbol{\theta}_u|\mathsf{T}_j) p(\{y\}_{u,-i}|\boldsymbol{\theta}_u, \mathsf{T}_j)}{p(\{y\}_{u,-i}|\mathsf{T}_j)} d\boldsymbol{\theta}_u \frac{p(\{y\}_{u,-i}|\mathsf{T}_j)}{p(\{y\}_u|\mathsf{T}_j)}}{\sum_{j=1}^{N} \frac{p(\{y\}_{u,-i}|\mathsf{T}_j)}{p(\{y\}_u|\mathsf{T}_j)}}
\end{aligned}
\tag{5.22}
$$

Consider the leave-one-out cross-validation in the breast cancer example. Each time we want to hold out the i^{th} observation and compute $P(Y_i = 1|\boldsymbol{y}_{-i})$. Similar to equation (5.5), function $h(\mathsf{T}_j)$ is

$$
E(\theta_u|\{y\}_{u,-i}, \mathsf{T}_j). \tag{5.23}
$$

From the marginal likelihood given in equation (2.18), the weight is

$$
\frac{\mathrm{B}\left(\sum_{k \in \mathcal{N}_{\mathsf{T}_j}(u,\mathcal{I}), k \neq i} y_k + a, n_u - 1 - \sum_{k \in \mathcal{N}_{\mathsf{T}_j}(u,\mathcal{I}), k \neq i} y_k + b\right)}{\mathrm{B}\left(\sum_{k \in \mathcal{N}_{\mathsf{T}_j}(u,\mathcal{I})} y_k + a, n_u - \sum_{k \in \mathcal{N}_{\mathsf{T}_j}(u,\mathcal{I})} y_k + b\right)} \tag{5.24}
$$

where n_u is the number of observations, including the ith observation, in leaf node u.

Turning now to the breast cancer data, the entire analysis was done using all the observations (683) observations as the data, producing $25,000$ posterior tree samples. These trees were repeatedly used to calculate $P(Y_i = 1|\boldsymbol{y}_{-i})$, $j = 1, 2, \ldots, 683$. Figure 5.9 displays the results. With a simple threshold at 0.5 on the Monte Carlo estimates of the implied predictive probabilities, averaged over all $25,000$ trees, the raw prediction error is 16 out of 683 (2.34%).

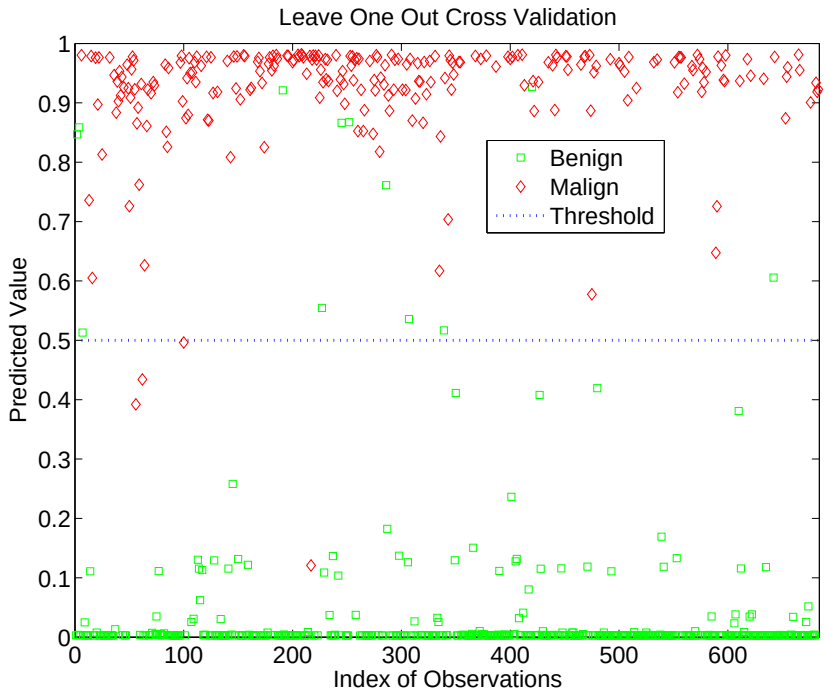


Figure 5.9: Breast cancer example: The results for leave-one-out cross-validation. For each observation i, $P(Y_i = 1|\boldsymbol{y}_{-i})$ is shown. The squares and diamonds represent subjects with benign and malign recurrence, respectively. The simple threshold at 0.5 is overlaid.

Computationally, this leave-one-out cross-validation by importance sampling is much faster than the naive method[2]. In fact, the leave-one-out cross-validation is supposed to be more computationally intensive than the ten-fold cross-validation analysis, yet it took less time to run the leave-one-out analysis. One may want to try this importance sampling method for the ten-fold cross-validation; however, this analysis can be problematic. As we explained in the text next to equation (5.21), given a tree sample T_j, if the left-out data fall into the same leaf, where there are no other observations, the weight for the tree sample is 0. If we apply the importance sampling method for ten-fold cross-validation, this is more likely to happen. It does not make much sense to specify a very large leaf size so that there are at least one tenth of the observations in any leaf node.

5.5 Discussion

In this chapter, by studying the breast cancer data, we discussed several aspects of Bayesian analysis in tree models. We first discussed the convergence problem. The K-S procedure was introduced to attempt to explore the convergence problem. As stated in the text, this was not a sufficient tool. We are not bold enough to say through this procedure the output of the chain from our algorithm had converged. Instead, particularly for this data example, we claimed the algorithm with the restructure move explored more tree space than the one with only basic moves and its mixing property seemed to be better. However, although the improvement of restructure move is obvious, we still can not find a stopping criterion to guide the chain. Some other nonparametric methods may be tried to handle this problem

[2]With our implementation in late 2005, the cross-validation for this data set after the posterior tree samples ($25,000$ samples) were produced took about 3 minutes on an Intel Xeon $3.0Ghz$ with $2Gb$ memory.

in the future.

The second part of this chapter focused on the application of importance sampling on leave-one-out cross-validation analysis. We avoided regenerating posterior samples for each held-out data set. Believing that the posterior distribution given the held-out data set is close to the one given the whole data set, we used the latter as the instrumental distribution in importance sampling. Detailed calculations were given for the binomial case; however, we have not yet discussed what to do when this method is not applicable. This is left to be discussed in Chapter 7.

Chapter 6

Example III: Proteomics Data and Resampling

In this chapter, we discuss another data set from cancer proteomic analysis. The response is also a binary variable as in the breast cancer data discussed in Chapter 5. At first sight, there is nothing new; however, we present this example here for two purposes.

First, we want to show that in some cases tree models can not be directly applied to the data. In Chapter 5, we showed how to make predictions and do cross-validation analysis. The prediction for breast cancer data using our tree model is better than with other available tree models. In studying the proteomics data with our tree model, a primitive analysis will give a very good result in terms of prediction; however, it is less encouraging when we look at the data closely. Therefore, a specific algorithm is needed for such data sets.

Second, we find that the predictive validity is questionable because each patient has multiple samples in the data. In primitive analysis, samples from the same patient tend to fall into the same leaf; this reduces the credibility of leave-one-out

cross-validations. We extend the tree model using resampling method to address this issue. New sampling schemes are introduced to deal with the within-patient structure.

6.1 Data description

This proteomic breast data contains 554 samples, which are sampled from the tumor or lymph nodes on each patient and generated as mass spectra traces using the Matrix Assisted Laser Desorption/Ionization Time-of-Flight (MALDI-TOF) technology (Franzen, 1997; Campa *et al.*, 2003a). The range of mass per charge is divided into $p = 268$ intervals and intensity is measured in each interval. For each sample, we have 268 measured values, denoted as $(x^1, x^2, \ldots, x^{268})$ for this sample. In mass spectrometry analysis, the location of the peak is important and informative. By exploring the relationship between covariates and clinical outcomes, we aim to study the predictive power of such mass spectrum data. We have three different responses, which are ER (estrogen receptor status), HER2 (related hormone receptor status), and LN (lymph node status), each of them a binary outcome (0 and 1). Some values are missing and the corresponding response are assigned -9.

An interesting structure in this data set is that multiple samples can come from the same patient. The 554 samples come from only 44 patients. We will explore this structure by incorporating resampling methods, introduced in later sections, into the Bayesian tree model.

The mass spectrum traces for patients with id 9 and 16, marked with blue solid lines and red dashed lines correspondingly, are shown in Figure 6.1. The ER of patient 9 is 0 while the ER of patient 16 is 1. Obviously the intensity

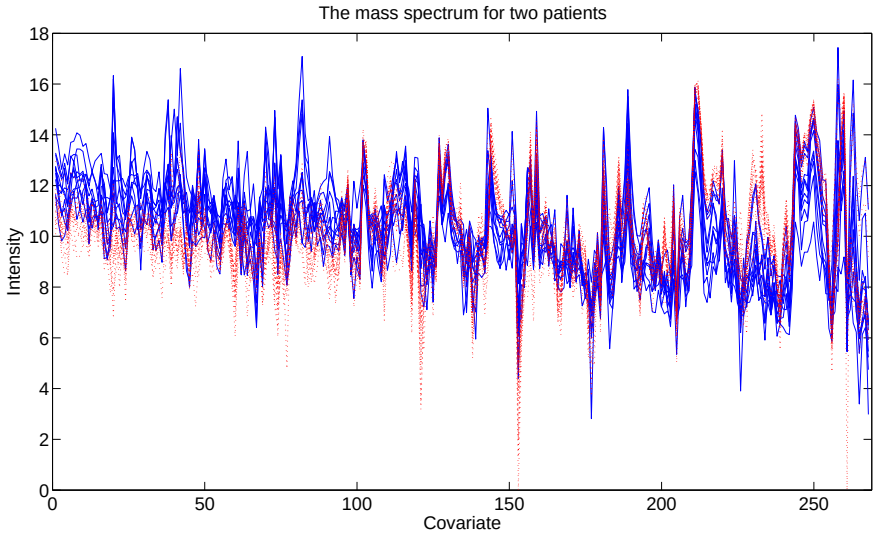


Figure 6.1: The mass spectra traces for patients with id 9 (blue sold lines) and 16 (red dashed lines).

of $x^1, x^2, \ldots, x^{100}$ for patient 9 is much higher than patient 16. Meanwhile, the intensities of the remaining covariates for these two patients share similar structure. This may indicate that some of the first 100 covariates may be informative in predicting ER level.

A simple generalized linear model analysis shows that there may be some predictive information in the regressors with respect to each of the responses. The tree model is used for the following two reasons. First, in mass spectrum analysis, whether a peak occurs at certain location is very informative. The indication of a peak corresponds directly to a splitting rule in the tree model and the splitting rules in the tree model can be easily interpreted in this context. Second, the covariates are highly correlated. We aim to model the dependence structure of the covariates using tree models.

6.2 Analysis

In this section, we will first briefly describe the model specification and then give results on the importance of predictors and the predictions.

For each of the three responses, we specify a model that is very similar to the one in Section 4.2. For T, the pinball prior has $\alpha(m) = 1 + \text{Pois}(m-1; 10)$, where $m = 10$ for this data and $\beta(i|m) = 1+\text{Bin}(i-1; m-2, 0.5)$. The splitting variables are chosen uniformly via $\gamma(i) = \frac{1}{268}$, and the splitting thresholds come, for all k, from $\delta_k(\cdot) = \text{Unif}[0, 1]$ independently (all the covariates have been transformed according to equation (2.8)). The within-leaf sampling model is bernoulli with probability of ER(HER2, LNPOS) positive being θ_u in leaf u having independent uniform prior, as specified in equation (2.17) taking $a = b = 1$. In this data, there are 135 missing values found in ER and there are 117 missing values found in each of HER2 and LNPOS. In the following text, if ER is 1, we call it *ER positive*. If ER is 0, we call it *ER negative*, though this is somehow ambiguous. Otherwise, we call it *ER missing*. Similar terms are adopted for HER2 and LNPOS. In running the MCMC algorithm, we will first ignore the missing values. Later, we will make prediction for those samples with the posterior tree samples.

We run an MCMC using the algorithm with change, grow/prune, swap and restructure moves and produce $25,000$ trees. The histograms of the tree size, log integrated likelihood and log posterior probability of the tree samples are displayed in Figure 6.2. The upper, middle and lower panels correspond to the tree samples from the posterior distribution given ER, HER2 and LNPOS respectively. The size of the trees corresponding to ER and LNPOS ranges from 10 to 20 while the size of the trees corresponding to HER2 ranges from 15 to 25. The priors for tree

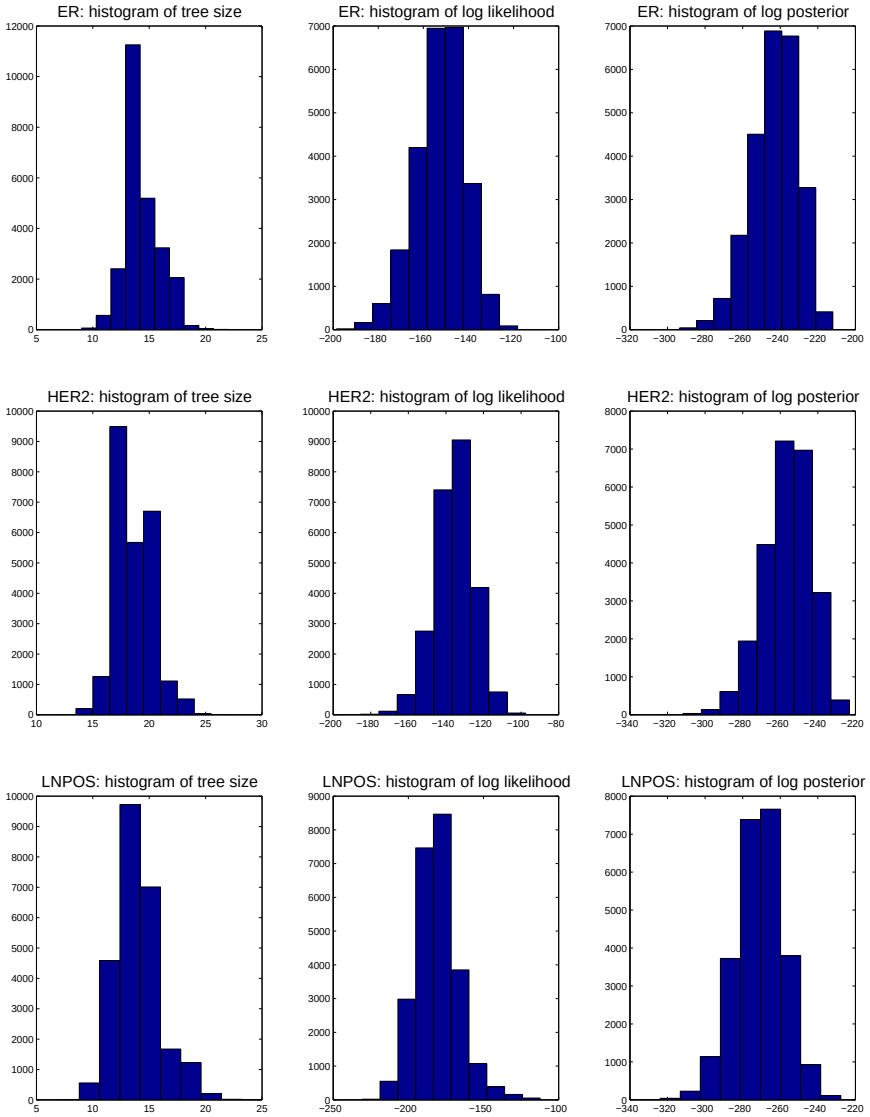


Figure 6.2: The histrogram of tree size, log integrated likelihood and log posterior prbability of the posterior tree samples. The upper, middle and lower panels represents the results for ER, HER2, and LNPOS respectively.

Rank	ER	HER2	LNPOS
1	87 (1.00)	156 (1.00)	87 (1.00)
2	95 (1.00)	157 (1.00)	225 (1.00)
3	146 (1.00)	184 (1.00)	199 (1.00)
4	81 (0.96)	199 (1.00)	149 (1.00)
5	100 (0.96)	116 (0.98)	192 (0.82)
6	256 (0.94)	173 (0.79)	76 (0.71)
7	241 (0.85)	147 (0.76)	248 (0.70)
8	30 (0.79)	31 (0.68)	38 (0.55)
9	159 (0.44)	81 (0.61)	77 (0.48)
10	163 (0.42)	159 (0.60)	47 (0.25)

Table 6.1: The ten most frequently used predictors in the posterior tree samples given each of three responses: ER, HER2 and LNPOS.

size all have a mean at 10. We also tried different priors for the tree size and found that the results are insensitive to this specification. The histograms of log integrated likelihood and log posterior probability also show that a wide range of tree model have been explored.

In this analysis we aim to find which predictors, corresponding to the intensity at certain locations, are significant in predicting the responses. For each predictor, we calculate the posterior probability that this predictor is used (at least once) as a splitting variable. The top ten most frequently used predictors are listed in Table 6.1. Interesting information can be mined out of this table. For example, the $87th$ predictor ranks highest in posterior probability for both responses ER and LNPOS. We take a closer look at Figure 6.1. The enlarged mass spectrum between 81 and 93 is displayed in Figure 6.3. We can see that there is a clear cut at the $87th$ predictor, which is marked with green line, between patient id 9 and 16. Furthermore, the intensity at this location is relatively low, compared to the peak at the location of the $83th$ predictor. This might indicate that there is hidden information, possibly a peptide, at this location. We can also see that

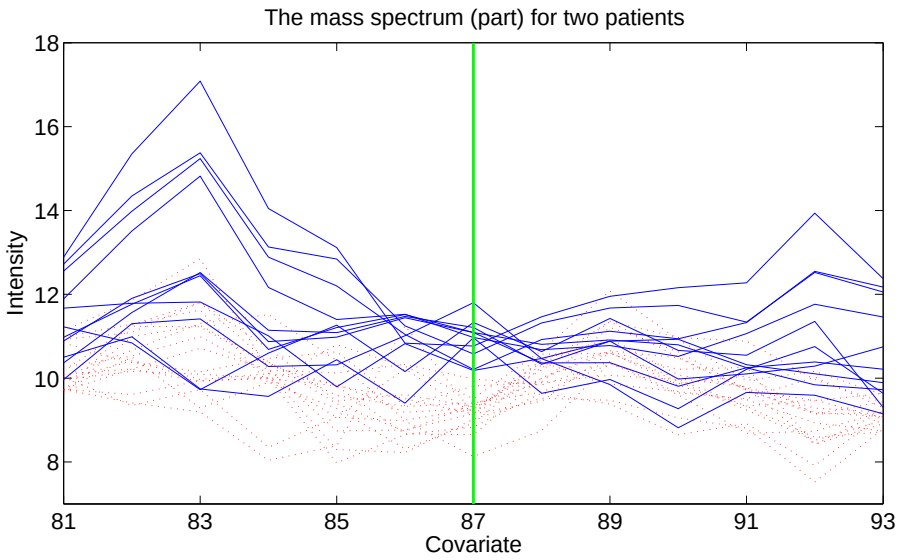


Figure 6.3: The mass spectra traces for patients with id 9 (blue solid lines) and 16 (red dashed lines). Only the part between 81 and 93 is shown.

199th predictor appears in top-ten list for both HER2 and LNPOS. Furthermore, considering the particular structure of this data, the 146th predictor shown in ER column, the 147th predictor shown in HER2 column, and the 149th in $LNPOS$ column may give the same important signal because they are close to each other and hence may represent the same peak. Such information can be sent back to the biologists and further studied from a proteomic point of view.

To assess prediction validity we ran a leave-one-out cross-validation. We are interested in calculating $P(Y_i = 1|\boldsymbol{y}_{-i})$ and comparing this predictive probability to some threshold, which is the number of positive values in each data set over the total number of non-missing values in this analysis. If $P(Y_i = 1|\boldsymbol{y}_{-i})$ is higher than the threshold, this sample is predicted to be positive and otherwise predicted to be negative. We consider the fraction of the samples where the sample status differs from the predicted status. This fraction is the misclassification rate in the leave-one-out cross-validation analysis. Figure 6.4 displays the predicted

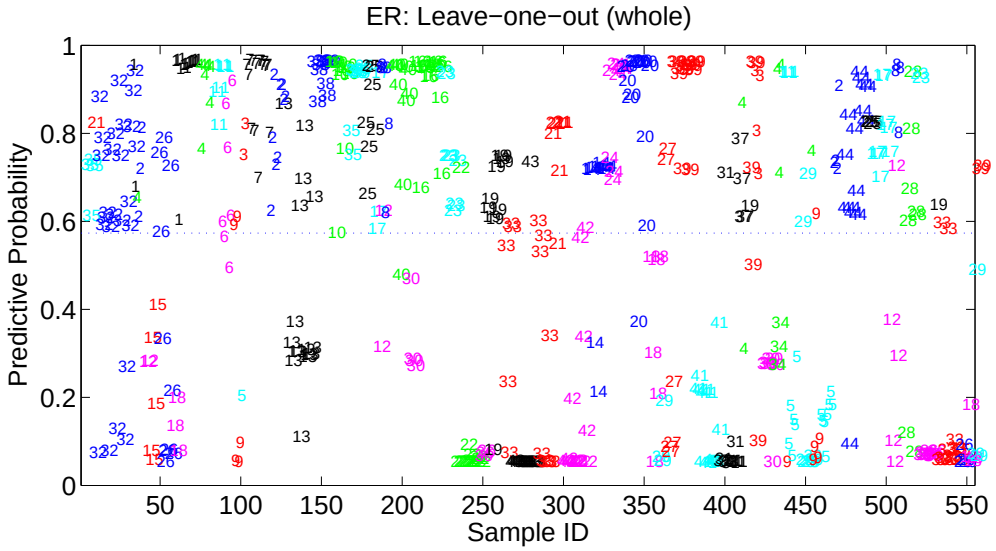


Figure 6.4: The predicted probability $P(Y_i = 1|\boldsymbol{y}_{-i})$ in leave-one-out cross-validation. The dashed line indicates the threshold, which is the number of positive values in each data set over the total number of non-missing values in this analysis.

probabilities for ER response. In this figure, each sample is marked with its patient id. Obviously we do not have a clear separation between predicted ER positive and predicted ER negative. Quite a few predicted values are very close to the threshold. However, the misclassification rate is only 0.066.

We also draw the boxplot of predicted values grouped by the patient id. Figure 6.5 displays the boxplots for ER positive, ER negative and ER missing. The upper panel shows the boxplots of the predicted probabilities for ER positive. The mean predicted probabilities for each patient are all greater than the threshold, although some of them are very close to the threshold value. The middle panel shows boxplots of the predicted probabilities for ER negative. It is obvious that the variance of the predicted values within patient are generally greater than those in ER positive. The predicted probabilities shown in these two panels are calculated by leave-one-out cross-validation. We also calculate the predicted prob-

ability $P(Y_i^*|\boldsymbol{y})$ for those samples with ER missing response. This is computed by averaging over all the trees generated from the posterior distribution given the whole set of observations. The lower panel in Figure 6.5 displays these predictive probabilities for ER missing. The variances are even greater.

At first sight, these results seems to be very encouraging and one might therefore think if the Markov chain can be assumed to have converged, these results are ready for discussion with the collaborative biologists. However, before we can make such a claim, we need to step back and take a closer look at the posterior samples. Figure 6.6 shows a randomly picked tree from the posterior samples given the data with ER response. For most of the leaf nodes, the number of observations in the node is quite small. It is quite possible that most of the leaf node contains only the sample IDs from the same patient. A closer look at the leaf node supports our conjecture. We know that the prediction given the tree sample is related to the observations falling into same leaf node. So for example, the samples of patient A into the left-most leaf node in the tree. We hold out one sample i and try to compute $P(Y_i = 1|\boldsymbol{y}_{-i})$. Remember we can use importance sampling to compute this probability without regenerating the posterior samples according to equation (5.22). If the samples falling into the same leaf as the i^{th} sample are mostly ER positive, the predicted probability of Y_i being 1 is high and close to 1. Otherwise, the predicted probability is low and close to 0. Note that we observe that the samples from the same patient tend to fall into the same leaf and it is likely that this leaf will contain only one patient because the number of observations in each leaf node is relatively small. The prediction is therefore very "accurate". However, this is not the structure we want to explore. We aim to explore the similarity in the mass spectrum between patients, not within patients.

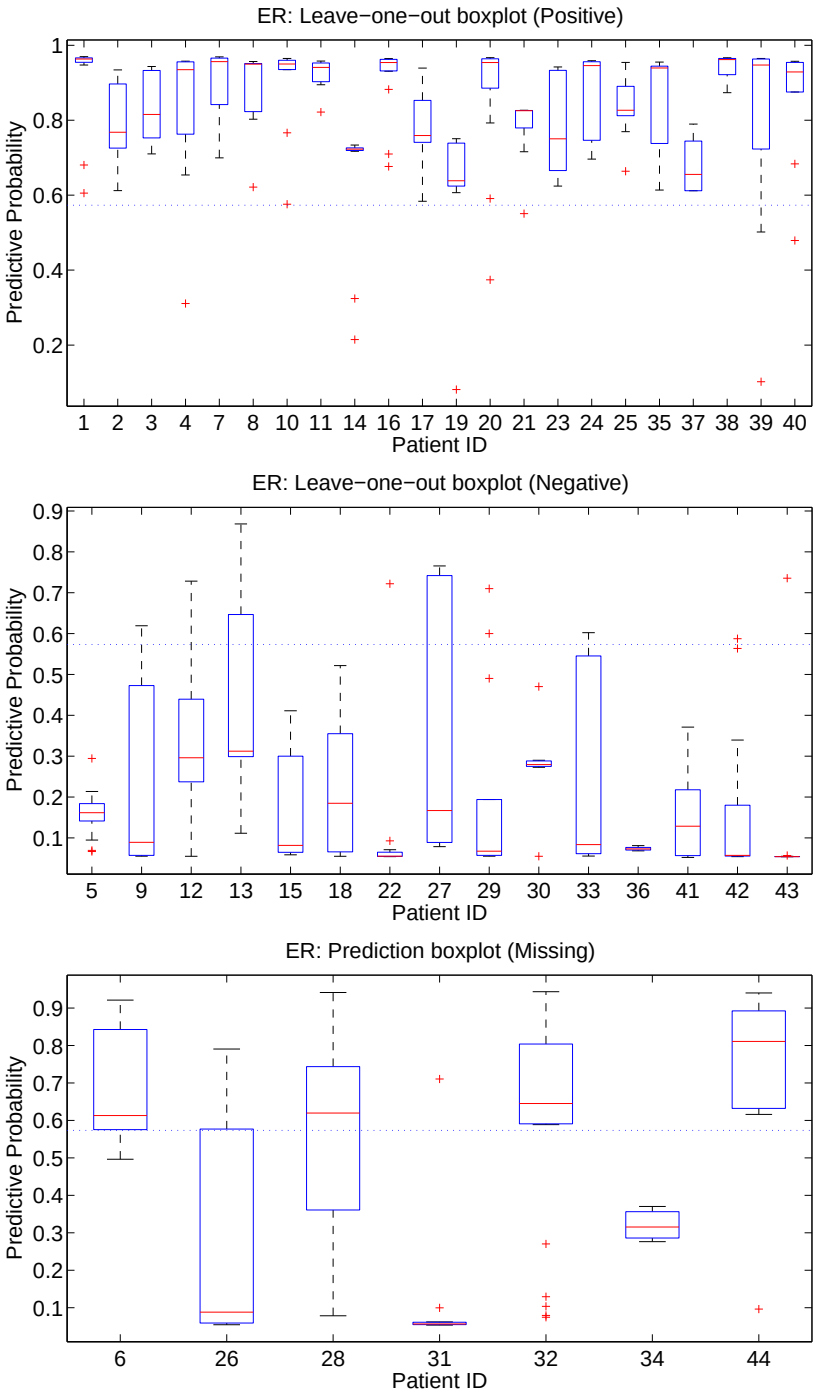


Figure 6.5: The boxplot of predicted values for ER response grouped by the patient id. The upper panel and the middle panel display the predicted probabilities for ER positive patients and ER negative patients respectively. The lower panel displays the predicted probabilities for ER missing.

We can enforce a more restricted limit on the leaf size, as described in equation (3.2), to have more patients into one leaf. However, this circumvention is rather ad hoc and we do not have a clear idea what this limit should be. Another way to assess the prediction validity is to do hold-one-patient-out cross-validation. We did try that and the results do show some problems with this data; however, this cross-validation is computationally expensive. In doing the cross-validation, we hold out one patient, which means tens of samples, at each time. As we discussed in Section 5.4.2, the importance sampling method is not feasible in this situation. Therefore, we need to regenerate the tree samples when we hold out a patient, which dramatically increases the computation. We will introduce a resampling idea in next section to deal with this particular problem and this dataset.

The above discussion is based on the leave-one-out cross-validation analysis for ER response. We also did the same analysis for HER2 and LNPOS, which are shown in Figure 6.7 and Figure 6.8 respectively.

6.3 Resampling

In this dataset, we have multiple measurements for each patient. We denote these measurements as $\boldsymbol{X}_i = \{\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^{r_i}\}$, where r_i is the number of samples for patient i. In the resampling model, instead of directly using these measurements as the covariates, we treat them as noisy representatives. Suppose these r_i measurements are a random sample from some unknown, patient specific distribution $F_i(x)$. Assume a Dirichlet process prior with parameter α on $F_i(\cdot)$. That is for every k and all measurable partitions $(B_1, \ldots, B_k)$ of $\mathcal{X}$, the distribution of $(P(B_1), \ldots, P(B_k)) = (F_i(x \in B_1), \ldots, F_i(x \in B_k))$ is Dirichlet $\mathcal{D}(\alpha(B_1), \ldots, \alpha(B_2))$. Now $\boldsymbol{X}_i$ is a sample from $F_i(\cdot)$. As discussed in Fergu-

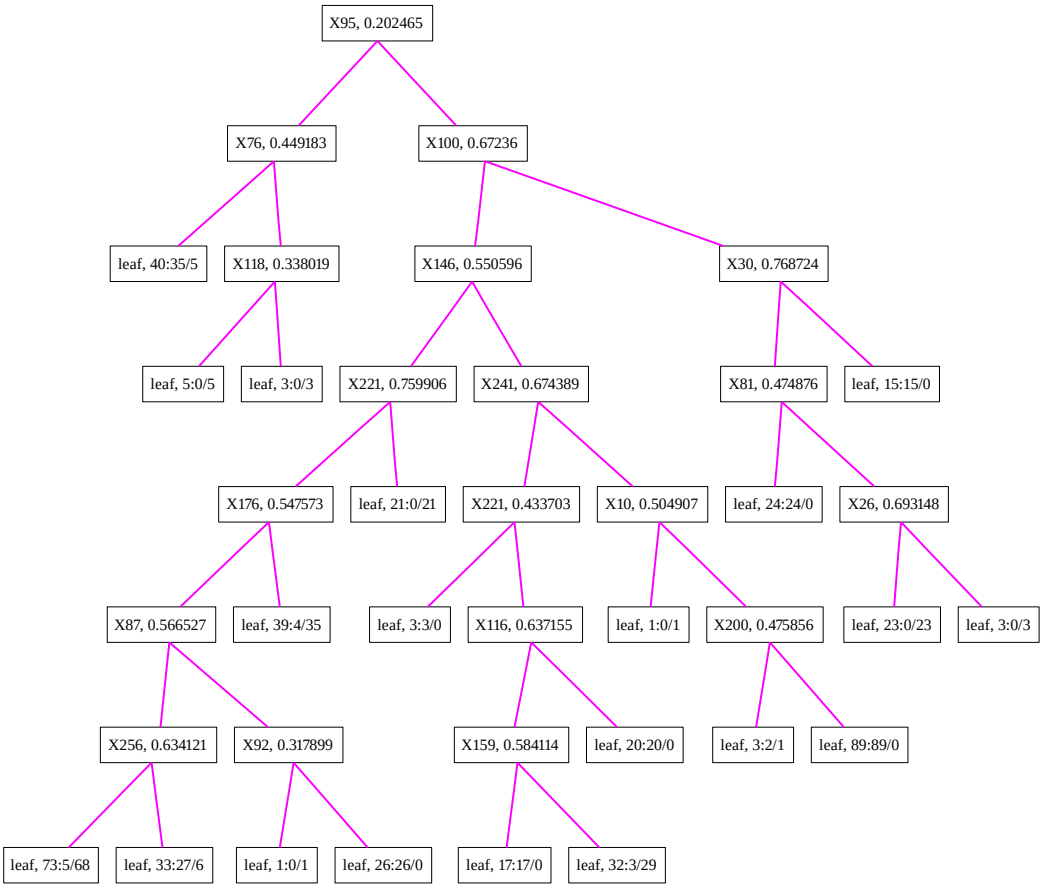


Figure 6.6: A randomly picked tree sample from the posterior tree samples given the data with ER response.

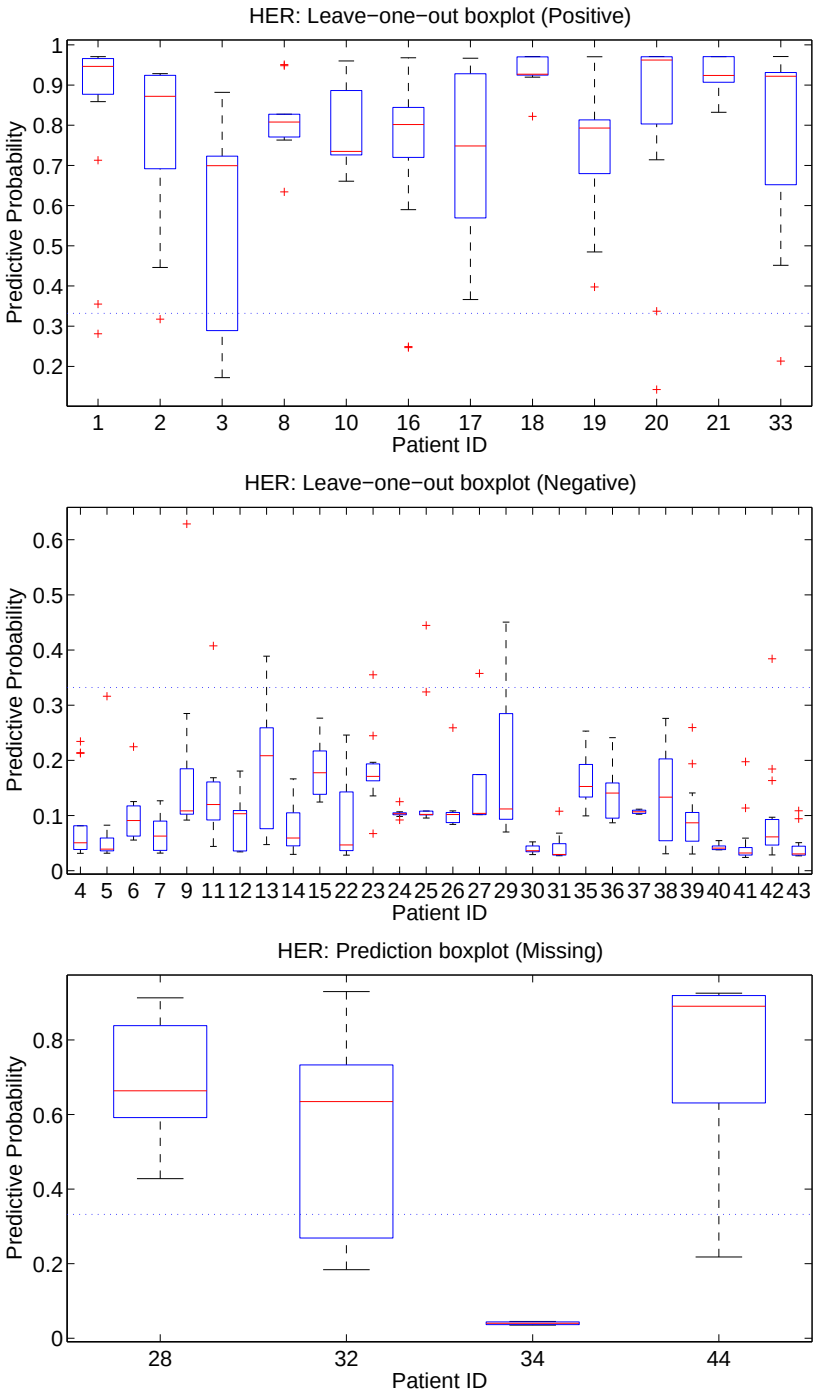


Figure 6.7: The boxplot of predicted values for HER2 response grouped by the patient id. The upper panel and the middle panel display the predicted probabilities for HER2 positive patients and HER2 negative patients respectively. The lower panel displays the predicted probabilities for HER2 missing.

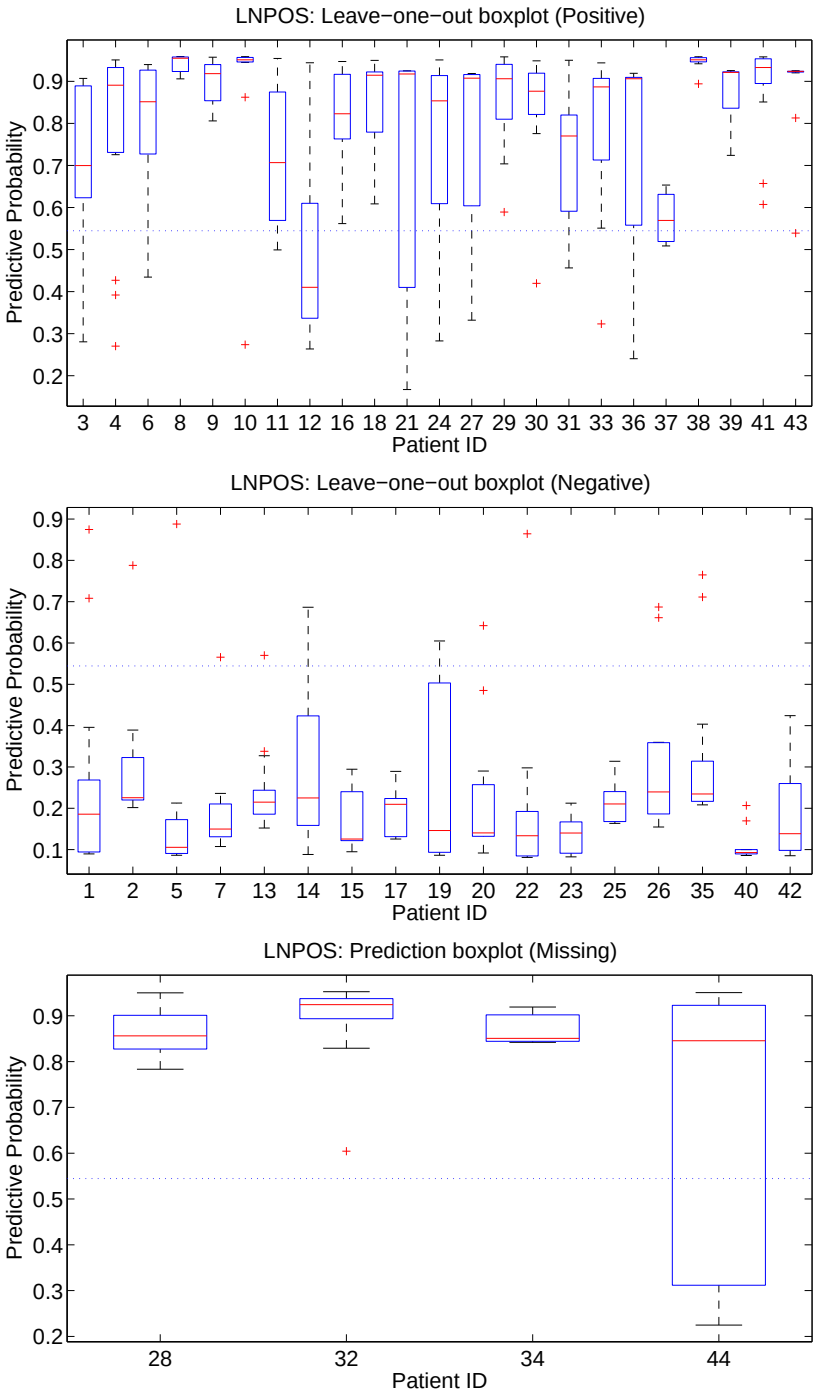


Figure 6.8: The boxplot of predicted values for LNPOS response grouped by the patient id. The upper panel and the middle panel display the predicted probabilities for LNPOS positive patients and LNPOS negative patients respectively. The lower panel displays the predicted probabilities for LNPOS missing.

son (1973), the posterior distribution of $(P(B_1), \ldots, P(B_k))$ for all measurable partitions $(B_1, \ldots, B_k)$ of $\mathcal{X}$ given the data is

$$\begin{aligned} & f(P(B_1), \ldots, P(B_k) | \boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^{r_i}) \\ \propto \; & f(\boldsymbol{X}_i | P(B_1), \ldots, P(B_k)) f(P(B_1), \ldots, P(B_k)) \\ = \; & \prod_{l=1}^{k} \prod_{j=1}^{r_i} P(B_l)^{\delta_{\boldsymbol{x}_i^j}(B_l)} \prod_{l=1}^{k} P(B_l)^{\alpha(B_l)} \\ = \; & \prod_{l=1}^{k} P(B_l)^{\sum_{j=1}^{r_i} \delta_{\boldsymbol{x}_i^j}(B_l) + \alpha(B_l)}. \end{aligned} \tag{6.1}$$

Therefore the posterior distribution of $F_i(\cdots)$ is a Dirichlet process with parameter $\alpha + \sum_{j=1}^{r_i} \delta_{\boldsymbol{x}_i^j}$. The predictive distribution is therefore

$$F_i(t | \boldsymbol{X}_i) \;\; = \;\; \frac{\alpha((-\infty, t]) + \sum_{j=1}^{r_i} \delta_{\boldsymbol{x}_i^j}((-\infty, t])}{\alpha(\mathcal{X}) + r_i}. \tag{6.2}$$

Thus the predictive distribution is a mixture of prior guesses on $F_i(\cdot)$ and the empirical CDF. In the limiting case, when α gets small, the predictive distribution is just the empirical CDF

$$F_i(t | \boldsymbol{X}_u) \;\; \rightarrow \;\; \frac{1}{r_i} \sum_{j=1}^{r_i} \delta_{\boldsymbol{x}_i^j}((-\infty, t]) \tag{6.3}$$

and so t is just uniformly distributed across $\boldsymbol{x}_i^j$. When α is not 0, we allow t to take values other than $\boldsymbol{x}_i^j$, $j = 1, 2, \ldots, r_i$. This case may be of interest when there are just a few measurements in $\boldsymbol{x}_i$. Prior guess on $F_i(\cdot)$ explores the other values than $\boldsymbol{x}_i$ that could provide a better prediction $p(\boldsymbol{y} | \boldsymbol{x}, \mathsf{T})$.

Now we introduce the resampling method for our Bayesian tree model. Suppose patient i has outcome y_i and independent variable x_i. The model is

$$p(y_i | \boldsymbol{x}_i, \mathsf{T}). \tag{6.4}$$

Now we have multiple noisy measurements $\boldsymbol{X}_i = \{\boldsymbol{x}_i^1, \boldsymbol{x}_i^2, \ldots, \boldsymbol{x}_i^{r_i}\}$ on $\boldsymbol{x}_i$. In our data, some patients have as many as 26 and some have just 3 measurements. For each patient i, these measurements are assumed to be random samples from $F_i(\cdot)$. We have already shown that if we assume Dirichlet process priors on $F_i(\cdot)$, $\boldsymbol{x}_i$ is uniformly distributed. However, the conditional distribution of $\boldsymbol{x}_i$ is no longer uniform given the observed data.

$$\begin{aligned} p(\boldsymbol{x}_i|y_i, \boldsymbol{X}_i, \mathsf{T}) &\propto p(\boldsymbol{x}_i|\boldsymbol{X}_i)p(y_i|\boldsymbol{x}_i, \mathsf{T}) \\ &= \textstyle\sum_{j=1}^{r_i} q_i^j \delta_{\boldsymbol{x}_i^j} \end{aligned} \tag{6.5}$$

where q_i^j, $j = 1, 2, \ldots, r_i$ sum to 1, and

$$q_i^j \propto p(y_i|\boldsymbol{x}_i^j, \mathsf{T}). \tag{6.6}$$

That is, the conditional probability $p(\boldsymbol{x}_i^j|y_i, \boldsymbol{X}_i, \mathsf{T})$ is proportional to the likelihood given that the j^{th} covariate in all the samples from the i^{th} patient is chosen as the representative.

We can now formally incorporate the within-patient information into the MCMC tree model analysis by resampling the covariates. Let p_i denote the index of measurement for patient i. The implied sampling algorithm is then:

1. For each i, set $\boldsymbol{x}_i^{p_i}$ as the covariate. The p_i are all initialized to be 1. The data set is $(y_i, \boldsymbol{x}_i^{p_i})$, $i = 1, 2, \ldots, m$, where m is the number of patients.

2. Sample from $p(\mathsf{T}|(y_i, \boldsymbol{x}_i^{p_i}), i = 1, 2, \ldots, m)$ using the Metropolis-Hastings algorithm with all four types of proposals. Repeat this step long enough.

3. For each i, compute $q_i^j = p(y_i|\boldsymbol{x}_i^j, \mathsf{T})$. Set $\boldsymbol{x}_i^{p_j} = \boldsymbol{x}_i^j$ with probability $\dfrac{q_i^j}{\sum_{l=1}^{r_i} q_i^l}$.

4. Go to step 2.

This resampling algorithm basically means that if the likelihood is in favor of one specific covariate, then the probability that this covariate is sampled to be the predictor is higher than the others.

Once we have the posterior samples, we can make prediction about the new observation by averaging over the tree samples, i.e.

$$p(\boldsymbol{y}_*|\boldsymbol{y}, \boldsymbol{x}, \boldsymbol{X}_*) \approx \frac{1}{N}\sum_{i=1}^{N} p(\boldsymbol{y}_*|\boldsymbol{X}_*, \mathsf{T}_i) \tag{6.7}$$

where $\boldsymbol{X}_* = \{\boldsymbol{x}_*^1, \boldsymbol{x}_*^2, \ldots, \boldsymbol{x}_*^{r^*}\}$ is the multiple measurements of mass spectrum for this new patient and T_i, $i = 1, 2, \ldots, N$ are samples from $p(\mathsf{T}_i|\boldsymbol{y}, \boldsymbol{x})$. The distribution $p(\boldsymbol{y}_*|\boldsymbol{X}_*, \mathsf{T}_i)$ is computed by averaging over all the measurements

$$\int p(\boldsymbol{y}_*|\boldsymbol{x}_*, \mathsf{T}) P(\boldsymbol{x}_*|\boldsymbol{X}_*) d\boldsymbol{x}_* \approx \frac{1}{r^*}\sum_{l=1}^{r^*} p(\boldsymbol{y}_*|\boldsymbol{x}_*^l, \mathsf{T}) \tag{6.8}$$

Similar to equation (6.3), $\boldsymbol{x}_*$ is just uniformly distributed across $\boldsymbol{X}_*$.

We then run a leave-one-out cross-validation, but we are holding out one patient at a time. Note that the importance sampling method discussed in Section 5.4.2, which greatly reduces computation cost, is still applicable here. Therefore, we can generate the posterior tree samples only once. Figure 6.9 displays the predicted probabilities that ER is 1 for each patient held out. The upper panel and lower panel show the predicted values for ER positive patients and ER negative patients respectively. It is obvious that the variance within each patient is very large and the mean values are all close to the threshold. Compared to Figure 6.5, Figure 6.9 suggest that it may be very difficult to make predictions using the information from the other patients. Figure 6.5 looks good because the covariates within one patient are similar to each other, which does indicate the utility of the

tree model for evaluating the degree of within patient similarity as opposed to prediction.

6.4 Discussion

In this chapter, we explored tree modeling in analysis of a clinically interesting proteomic dataset. This is a very interesting example because of its samples-within-patient structure. Ignoring this structure, our tree model gave very good predictions; however, the tree model seemed to classify the samples according to the patient, which was not informative for cross patient predictions. If all the samples are known to sample from similar tumor locations within each patient and the number of samples for each patient are the same, we can consider all the samples from each patient as one single observation. As this was not the case, we introduced a resampling method, which assumes that the samples are from an unknown patient-specific distribution. The revised analysis and results by the MCMC algorithm were discussed.

This resampling method is an interesting idea to model data that have structure that is similar to this proteomic data. But it fails to make good predictions in cross-validation analysis. This failure suggests we step back and analyze the data again, though this resampling method still helps to diagnose some problems in the analysis discussed in Section 6.2. I also exploits the importance sampling method introduced in Section 5.4.2 to reduce the computation cost for leave-one-patient-out cross-validation.

This resampling example is a rather incomplete analysis. For example, we have not yet studied the posterior probability of each sample being used in the tree. Furthermore, we can explore how to improve the leave-one-out cross-validation

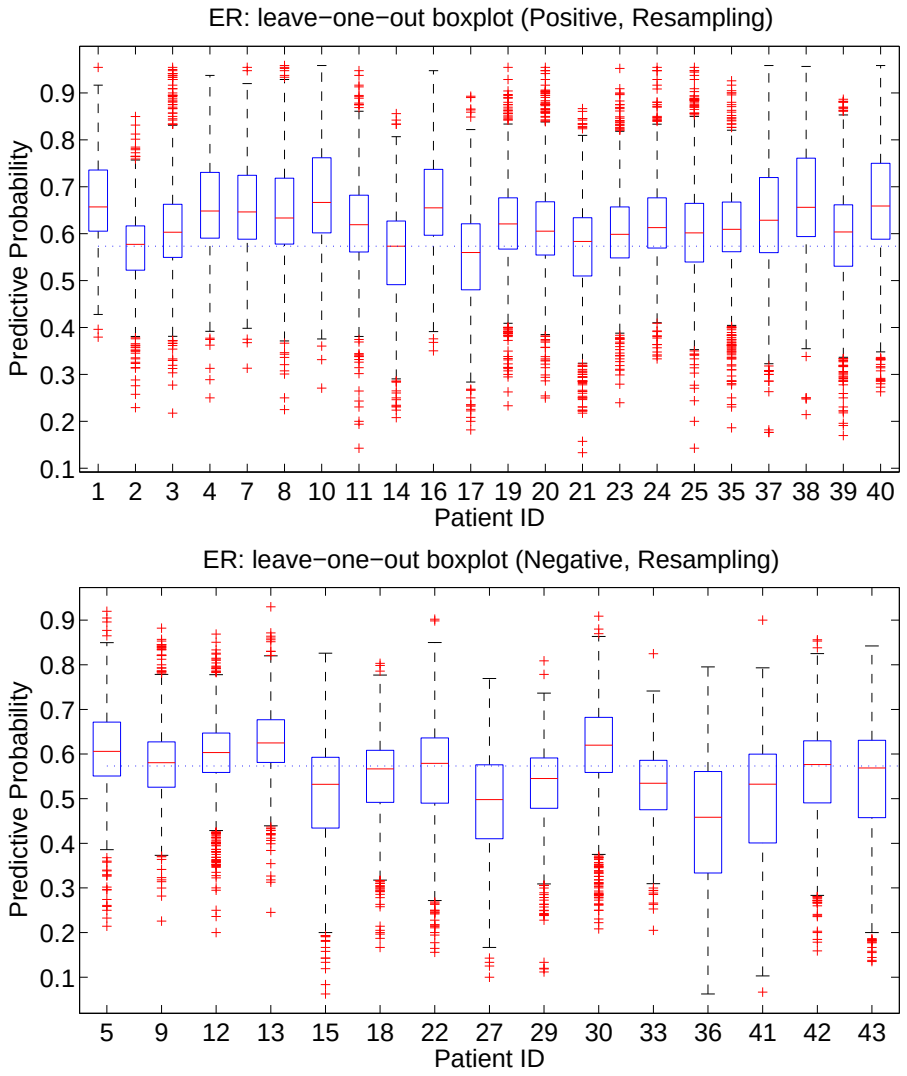


Figure 6.9: The boxplot of predicted values in leave-one-out cross-validation analysis for ER response grouped by the patient ID. The upper panel and the lower panel display the predicted probabilities for ER positive patients and ER negative patients respectively.

with our Bayesian tree model in the future.

Chapter 7

Random Threshold

One major feature of tree models is the sharp partitioning of data into separate, unrelated subgroups. With a single predictor variable this induces a "step-function" regression predictor, and the inherent discontinuity can sometimes be regarded as undesirable. Here we introduce the novel idea of "random thresholds" to begin to address this issue.

In the first part of this chapter, we explore a simple example with only a single threshold to motivate and explain the ideas of random threshold modeling. Bayesian analysis is given for this simple example. Bayesian tree models with random thresholds are then described in some generality. We discuss aspects of MCMC analysis in this framework, and also comparisons of predictions between the random threshold tree and regular tree using a simulated example.

7.1 Illustrative example

7.1.1 Model specification

In the tree model, we recursively partition the covariate space into sub-regions. In each region, a simple distribution is assumed. Consider a simple example data as shown in the left panel of Figure 7.1. In this data, the mean value of y given x is discontinuous at $x = 0$. In other words, if we make a split at $x = 0$, which is marked with black dashed line, the observations in each region can be modeled with simple distribution. To fit this data, we can use

$$y \sim \begin{cases} \mathrm{N}(\mu_1, \sigma^2), & x \leq \tau, \\ \mathrm{N}(\mu_2, \sigma^2), & x > \tau. \end{cases} \tag{7.1}$$

This corresponds to a tree model with one splitting variable and normal distributions in the leaves; however, this data is less realistic than that displayed in the right panel of Figure 7.1, where a clear cut at $x = 0$ does not exist. If we still use a tree model to fit this data, we will need to make more than one split. The regression function of y given x may look like the (green) curve in this graph. As there are more splitting variables, the tree size may get very large and thus it is hard to capture and analyze the true structure.

Before we introduce the idea of random threshold, we explain why this could happen. A first cause is the error in the variable. In this case, the observations are indeed distributed with different mean values across the regions and the regions are defined by the splitting variable z, but we do not observe this variable or the values of z are noisy. Suppose that we actually observe x as the true underlying z plus noise ϵ, where $\epsilon \sim \mathrm{N}(0, \gamma^2)$. In the true underlying model, the splitting threshold is $z = 0$. If x is far less than 0, the true underlying splitting variable

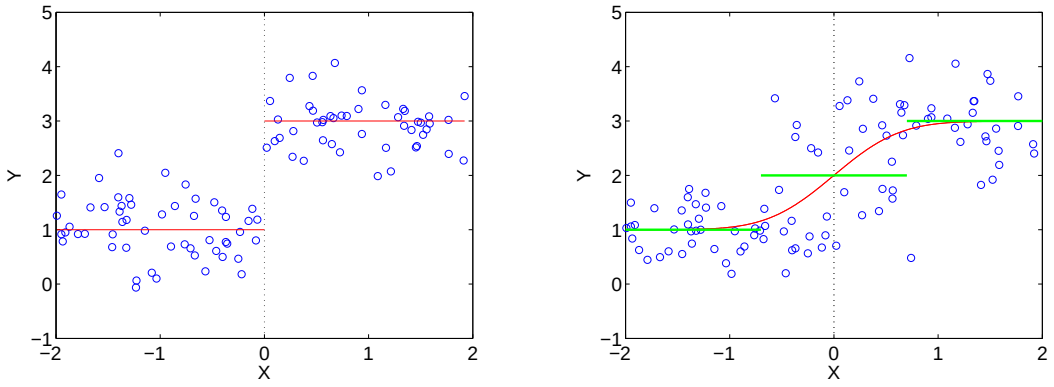


Figure 7.1: Illustrative example. Left panel: the mean value of y given x is disjointed. Right: the mean value of y given x is continuous and smooth.

z is also less than 0 with high probability. In this case, the distribution of y is similar to the one defined in region $\{z : z \leq 0\}$. The result is similar if x is far greater than 0; however, if x is close to 0, the underlying z can be either less or greater than 0. For example, if x is 0, the probability that z is less than 0 is 0.5. Therefore, if we observe only x, the distribution of y is a mixture of the distributions specified in the adjacent regions. Thus given only x, there is no clear cut for y at $x = 0$.

Assume

$$x = z + \nu \tag{7.2}$$

where $\nu \sim \mathrm{N}(0, \gamma^2)$ and

$$y = \mu_1 1(z \leq \tau) + \mu_2 1(z > \tau) + \epsilon \tag{7.3}$$

where $\epsilon \sim \mathrm{N}(0, \sigma^2)$. Given x, the probability density function of y can be com-

puted by integrating out z to give

$$\begin{aligned} p(y|x) &= \int p(y|x,z)p(z|x)dz \\ &= \int_{-\infty}^{\tau} p(y|z)p(z|x)dz + \int_{\tau}^{\infty} p(y|z)p(z|x)dz \\ &= \phi(y;\mu_1,\sigma^2)(1-\Phi(x;\tau,\gamma^2)) + \phi(y;\mu_2,\sigma^2)\Phi(x;\tau,\gamma^2) \end{aligned} \tag{7.4}$$

where $\phi(x;\mu,\sigma^2)$ and $\Phi(x;\mu,\sigma^2)$ are Normal probability density function and cumulative distribution function respectively. Then it is easy to show that

$$\mathrm{E}(y|x) = \mu_1(1-\Phi(x;\tau,\gamma^2)) + \mu_2\Phi(x;\tau,\gamma^2). \tag{7.5}$$

That is, the mean of y given x is a continuous function of x, favoring smooth expected mean as shown in the right panel of Figure 7.1.

Another interpretation is via randomness in the threshold as analogous to errors in splitting variables. In equation (7.1), the distribution of each observation y_i is decided by (x_i,τ). That is, for all x_i, τ is the same. A subtler model introduces randomness in the threshold value. For example, for each observation the distribution of y_i is decided by (x_i,τ_i) and τ_i has a normal distribution with mean τ_0 and variance γ^2, with τ_i, $i = 1,2,\ldots,n$ being unobservable. We only know the distribution of τ by its mean and variance. This specification will lead to the same probability density function for y as in equation (7.4). We call this idea, which induces smoothness of the regression of y on x across the borders of the sub-regions, the *Random Threshold Tree Model*.

In both approaches, the variance parameter γ^2 is a smoothness parameter. Figure 7.2 displays the mean of y given x, as specified by equation (7.3), for different γ^2. As γ^2 gets smaller, the mean is less smooth and similar to the one given by equation (7.1), so that the random threshold tree approaches the standard model.

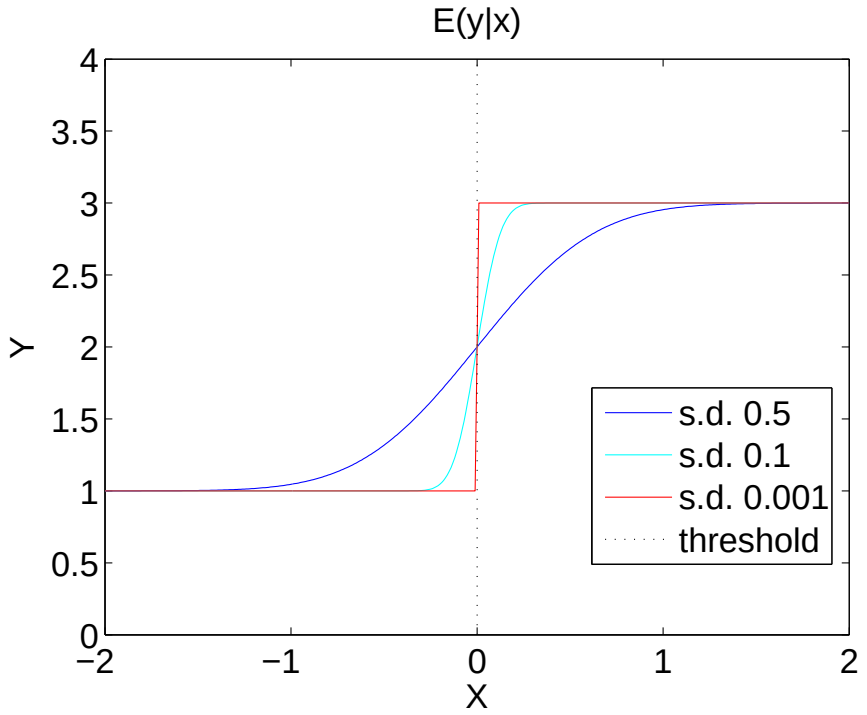


Figure 7.2: The expected mean of y given x for different γ^2.

7.1.2 Bayesian analysis

A natural prior choice for the parameters $(\mu_1, \mu_2, \sigma^2, \gamma^2, \tau)$ in equation (7.3) is

$$\begin{array}{lcl} \mu_1|\sigma^2 & \sim & \mathrm{N}(\mu_0, \sigma^2) \\ \mu_2|\sigma^2 & \sim & \mathrm{N}(\mu_0, \sigma^2) \\ \sigma^2 & \sim & \mathrm{IG}(a, b) \\ \gamma^2 & \sim & \mathrm{IG}(a, b) \\ \tau & \sim & \mathrm{Unif}(L, U) \end{array} \tag{7.6}$$

where (μ_0, a, b, L, U) are the pre-specified hyper parameters. There is no conjugate prior for τ; we choose a uniform distribution.

Given the observations (x_i, y_i), $i = 1, 2, \ldots, n$, the implied full posterior distribution may be sampled via MCMC using the following sequence of conditionals of each iteration:

1. For each i, sample $z_i|x_i, \gamma^2 \sim N(x_i, \gamma^2)$.

2. Sample $\sigma^2|\tau, \mathcal{D} \sim \text{IG}(\frac{n}{2} + a, \frac{1}{\frac{1}{4}\sum_{i=1}^{n}(y_i - \mu_0)^2 + \frac{1}{b}})$.

3. Sample $\mu_1|\tau, \sigma^2, \mathcal{D} \sim \text{N}(\frac{\mu_0 + \sum_{i=1}^{n} y_i 1(z_i \leq \tau)}{n+1}, \frac{\sigma^2}{n+1})$

   and $\mu_2|\tau, \sigma, \mathcal{D} \sim \text{N}(\frac{\mu_0 + \sum_{i=1}^{n} y_i 1(z_i > \tau)}{n+1}, \frac{\sigma^2}{n+1})$.

4. Sample $\tau|\mu_1, \mu_2, \sigma^2, \mathcal{D}$.

5. Sample $\gamma^2|\mu_1, \mu_2, \sigma^2, \mathcal{D}$.

6. Output $(\mu_1, \mu_2, \sigma^2, \gamma^2, \tau)$.

7. Go to step 1.

where $\mathcal{D} = (z_i, y_i)_{i=1}^n$. Note that it is hard to sample directly from the conditional distribution in step 4 and 5, so in these steps, we run a Metropolis-Hastings step with proposals drawn from the priors for each of the τ and γ^2.

For the data shown in the right panel of Figure 7.1, we choose $\mu_0 = 0$, $a = b = 1$, $L = -2$ and $U = 2$. Figure 7.3 displays the histograms of posterior samples of τ, μ_1 and μ_2. The posterior mean and corresponding Monte Carlo standard error for τ, μ_1 and μ_2 are 0.11(0.16), 1.13(0.15) and 2.95(0.14) respectively.

For any new x^*, we compute the predicted mean of y^* as

$$\begin{aligned} & E(y^*|x^*, (x_i, y_i), i = 1, 2, \ldots, n) \\ = & \int_\Theta y^* p(y^*|x^*, \Theta) p(\Theta|(x_i, y_i), i = 1, 2, \ldots, n) d\Theta \end{aligned} \tag{7.7}$$

where Θ is $(\mu_1, \mu_2, \sigma^2, \gamma^2, \tau)$. Given the posterior samples of Θ, the predicted mean is approximated by Monte Carlo integral. Figure 7.4 displays the predicted mean

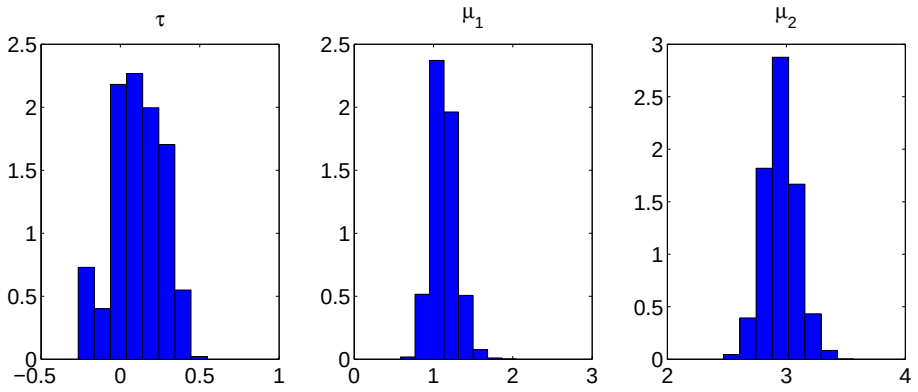


Figure 7.3: The histograms of posterior samples of τ, μ_1 and μ_2.

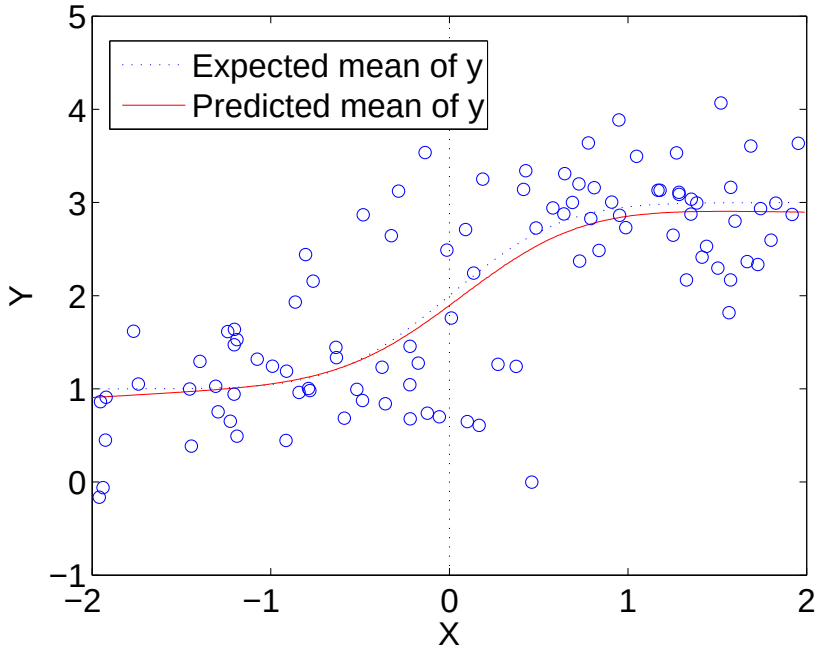


Figure 7.4: The predicted mean of y, shown as the red solid line. The blue dashed line is the expected mean level from the model.

level (red solid line), which is very close the expected mean level (blue dotted line) from the model in this simple example.

7.2 Tree models with random thresholds

7.2.1 Model specification and posterior

We now introduce random thresholds in tree models. As discussed in Section 7.1.1, we assume that the splitting thresholds are random in order to get smooth transitions in predictor space across the sub-regions. Remember a tree model T is specified by the triplet $(T, \boldsymbol{k}_T, \boldsymbol{\tau}_T)$. For each splitting threshold τ_u, $u \in a(T)$, we assume that

$$\tau_u \quad = \quad \rho_u + \nu_u \tag{7.8}$$

where ρ_u is called the "true" splitting threshold for node u. Therefore a complete tree model specification with random thresholds is $\mathsf{T}_R = (T, \boldsymbol{k}_T, \boldsymbol{\tau}_T, \boldsymbol{\rho}_T)$, where $\boldsymbol{\rho}_T = \{\rho_u, u \in a(T)\}$. The overall idea is that the ν_u terms will be small, random perturbations to the underlying thresholds, thus adding little "shocks" to the left/right specifications for each observation separately. It is key that the random threshold will have tight priors around the ρ values. The prior for T_R is specified by

$$\pi(\mathsf{T}_R) \quad = \quad \pi(\boldsymbol{\rho}_T|\boldsymbol{\tau}_T, \boldsymbol{k}_T, T)\pi(\boldsymbol{\tau}_T|\boldsymbol{k}_T, T)\pi(\boldsymbol{k}_T|T)\pi(T) \tag{7.9}$$

We adopt the same priors for $\pi(\mathsf{T}) = \pi(\boldsymbol{\tau}_T|\boldsymbol{k}_T, T)\pi(\boldsymbol{k}_T|T)\pi(T)$ as discussed in Chapter 2 and $\pi(\boldsymbol{\rho}_T|\boldsymbol{\tau}_T, \boldsymbol{k}_T, T)$ is specified by equation (7.8). Then the likelihood is

$$f(\boldsymbol{y}|\boldsymbol{\rho}_t, \boldsymbol{\tau}_T, \boldsymbol{k}_T, \mathsf{T}). \tag{7.10}$$

Note that $\boldsymbol{y}$ given $\boldsymbol{\rho}_T$, $\boldsymbol{k}_T$ and T are independent of $\boldsymbol{\tau}_T$, thus we can compute the posterior of the tree model with random threshold as

$$\begin{aligned} \pi(\mathsf{T}_R|\boldsymbol{y}) &\propto f(\boldsymbol{y}|\mathsf{T}_R)\pi(\mathsf{T}_R) \\ &= f(\boldsymbol{y}|\boldsymbol{\rho}_t, \boldsymbol{k}_T, \mathsf{T})\pi(\boldsymbol{\rho}_T|\boldsymbol{\tau}_T, \boldsymbol{k}_T, T)\pi(\boldsymbol{\tau}_T|\boldsymbol{k}_T, T)\pi(\boldsymbol{k}_T|T)\pi(T). \end{aligned} \tag{7.11}$$

Remember that random thresholds are introduced to achieve smoothing of the conditional distribution of y given x. We are less concerned with finding the exact partition of the feature space than with sampling from $\pi(\boldsymbol{\tau}_T, \boldsymbol{k}_T, T|\boldsymbol{y})$. This is obtained by integrating out $\boldsymbol{\rho}_T$ as

$$\pi(\boldsymbol{\tau}_T, \boldsymbol{k}_T, T|\boldsymbol{y}) = \int_{\boldsymbol{\rho}_T} \pi(\boldsymbol{\rho}_T, \boldsymbol{\tau}_T, \boldsymbol{k}_T, T|\boldsymbol{y}) d\boldsymbol{\rho}_T \tag{7.12}$$

The resulting posterior MCMC method, similar to the one in Section 7.1.2, is then straightforward, as follows:

1. Sample T and $\boldsymbol{\rho}_T|\mathsf{T}$ from their prior.

2. Propose a move in T (change, grow/prune, swap, restructure) to get a new tree model T'.

3. For each internal node u in T', sample $\rho_u \sim \mathrm{N}(\tau_u, \gamma^2)$. Thus we have $\mathsf{T}'_R = (\mathsf{T}', \boldsymbol{\rho}_T)$.

4. Compute the acceptance ratio $r(\mathsf{T}_R, \mathsf{T}'_R)$.

5. Take $\mathsf{T}_R = \mathsf{T}'_R$ with probability $\min\{r(\mathsf{T}_R, \mathsf{T}'_R), 1\}$.

6. Output T and go to step 2.

However this sampling scheme is not favored because: on one hand, we need to revise the codes for all the proposals; on the other hand, practically the new

proposed tree T'_R is less likely to be accepted due to the updating of $\boldsymbol{\rho}_T$. Rather, we can achieve a more effective MCMC utilizing the existing code and method by approximately integrating over the random threshold variables. In the practicable context of a fairly precise prior for the thresholds, this can be done as below and leads to the following modifications of the "regular" MCMC.

Notice that the posterior in equation (7.12) can be written

$$\int_{\boldsymbol{\rho}_T} f(\boldsymbol{y}|\boldsymbol{\rho}_t, \boldsymbol{k}_T, \mathsf{T})\pi(\boldsymbol{\rho}_T|\boldsymbol{\tau}_T, \boldsymbol{k}_T, T)\pi(\boldsymbol{\tau}_T|\boldsymbol{k}_T, T)\pi(\boldsymbol{k}_T|T)\pi(T)d\boldsymbol{\rho}_T. \tag{7.13}$$

Noting that the prior distribution $\pi(\boldsymbol{\tau}_T, \boldsymbol{k}_T, T)$ is independent of $\boldsymbol{\rho}_T$, the corresponding terms can be taken out of the integral and then the posterior is

$$\pi(\boldsymbol{\tau}_T|\boldsymbol{k}_T, T)\pi(\boldsymbol{k}_T|T)\pi(T)\int_{\boldsymbol{\rho}_T} f(\boldsymbol{y}|\boldsymbol{\rho}_t, \boldsymbol{k}_T, \mathsf{T})\pi(\boldsymbol{\rho}_T|\boldsymbol{\tau}_T, \boldsymbol{k}_T, T)d\boldsymbol{\rho}_T. \tag{7.14}$$

We can now directly use the Metropolis-Hastings algorithm discussed in Chapter 3 and the only difference is the computation of the likelihood function. The marginalized likelihood $f(\boldsymbol{y}|\boldsymbol{\tau}_T, \boldsymbol{k}_T, \mathsf{T})$ is approximated by

$$\frac{1}{M}\sum_{j=1}^{M} f(\boldsymbol{y}|\boldsymbol{\rho}_T^j, \boldsymbol{k}_T, \mathsf{T}) \tag{7.15}$$

where $\boldsymbol{\rho}_T^j$, $j = 1, 2, \ldots, M$ are sampled from its prior $\pi(\boldsymbol{\rho}_T|\boldsymbol{\tau}_T, \boldsymbol{k}_T, T)$. In this sampling scheme, we can directly use the current code without revising proposals and the only change is to define a new class for the computation of the likelihood.

Furthermore, given the posterior draws T_i, $i = 1, 2, \ldots, M$, it is not difficult to design an algorithm to sample from $\boldsymbol{\rho}|\boldsymbol{y}, \mathsf{T}_i$ if it is of interest.

7.2.2 Prediction and cross-validation

Given the posterior samples of the tree model, prediction is straightforward, via

$$p(y^*|\boldsymbol{y}) \;=\; \int_{\mathsf{T}} p(y^*|\mathsf{T},\boldsymbol{y})p(\mathsf{T}|\boldsymbol{y})d\mathsf{T}. \tag{7.16}$$

We also assess prediction validity by running leave-one-out cross-validation. In Section 5.4.2, we discussed how to use importance sampling to reduce the computational cost. However, in the tree model with random thresholds, the importance sampling method is not directly feasible.

The equation (5.20) is the key in the importance sampling method. More specifically, we need to have

$$p(\boldsymbol{y}|\mathsf{T}) \;=\; \textstyle\prod_{u\in b(T)} p(\{y\}_u|\mathsf{T}), \tag{7.17}$$

i.e., that the observations in different leaves are independent of each other given T. However, in the tree model with random thresholds the observations fall to the leaf node with certain probability given only T. We no longer have explicit definition of $\{y\}_u$, $u \in b(T)$ and equation (7.17) does not hold.

If we want to run leave-one-out cross-validation, the importance sampling method does not help to reduce the computational cost. Each time we hold out an observation, we need to rerun the algorithm and produce new samples.

7.3 Simulated example

We simulate data according to

$$y \;\sim\; \begin{cases} \mathrm{N}(1,1) & x_2 \le \rho_2, x_1 \le \rho_1, \\ \mathrm{N}(3,1) & x_2 > \rho_2, x_1 \le \rho_1, \\ \mathrm{N}(5,1) & x_3 \le \rho_3, x_1 > \rho_1, \\ \mathrm{N}(7,1) & x_3 > \rho_3, x_1 > \rho_1, \end{cases} \tag{7.18}$$

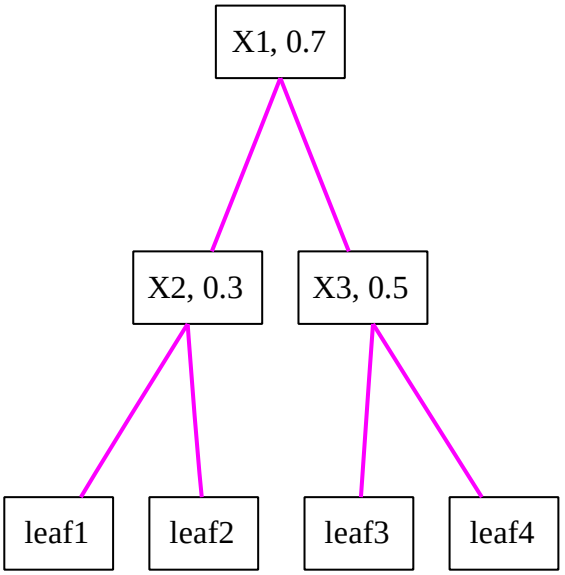


Figure 7.5: The tree structure for simulating the data.

where $\rho_1 \sim N(0.7, 0.01)$, $\rho_2 \sim N(0.3, 0.01)$ and $\rho_3 \sim N(0.5, 0.01)$. The corresponding tree structure is displayed in Figure 7.5. The simulated data set contains 162 observations. When we generate the data, we keep track of the "true" splitting threshold $\boldsymbol{\rho}$ for each sample. Therefore we know which of the four distribution it is drawn from. The data is shown in Figure 7.6. Four different points correspond to four different mean levels; however, they do not correspond to four leaf nodes in Figure 7.5, because by our model, each sample can fall into any leaf node as a result of the sample-specific randomly perturbed thresholds.

We ran the Metropolis-Hastings algorithm with change, grow/prune, swap and restructure proposals and produce $8,000$ tree samples. Given the posterior tree samples, we computed the expected mean of each sample, displayed in the left panel of Figure 7.7. As a comparison, the results for the regular tree model without random thresholds are also shown. First, it is not a surprise that the posterior means of the samples that are marked with cyan square are less than 7. For this

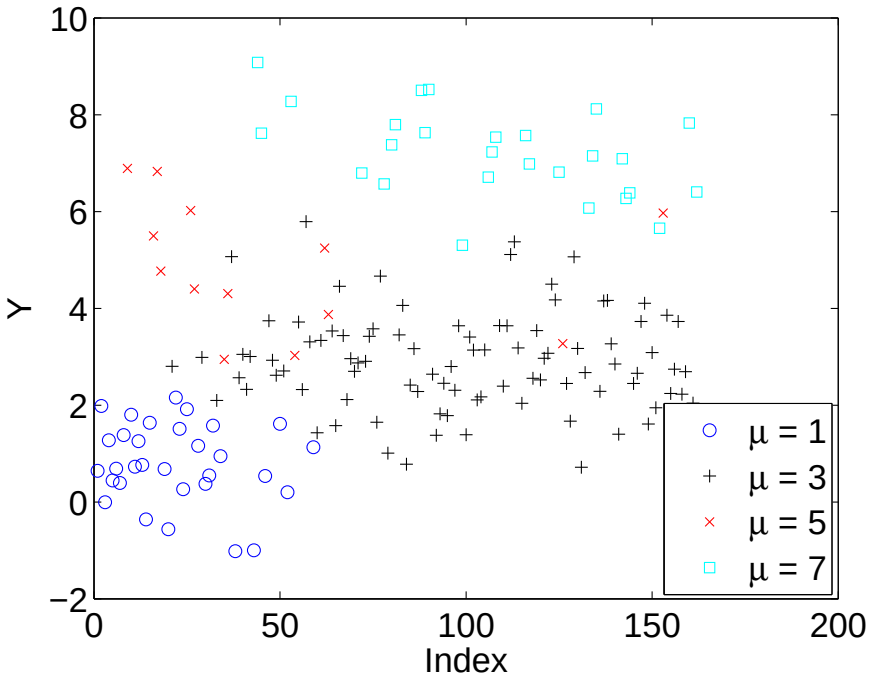


Figure 7.6: The data generated from the model specified in equation (7.18).

specific data, the within-leaf variance is relatively large and thus we do not expect to have a clear partition. It is very possible that those cyan square samples are classified with red cross and black plus samples and therefore the posterior means of cyan square samples are less than 7. Second, the posterior means are computed by averaging over all tree samples; however, as we can see in the right panel of Figure 7.7, most of posterior means of the black plus samples are the same. This does not reflect the fact that these samples could fall into the other leaves. In the left panel of Figure 7.7, the posterior means of blue circle samples and black plus samples are close to each other, indicating the probability for these samples to fall into the other leaf is high. This is not a surprise since their corresponding sub-regions are immediately adjacent and belong to the same region $\{x_1 \leq \rho_1\}$, so indicating the relevance and utility of the new random threshold idea.

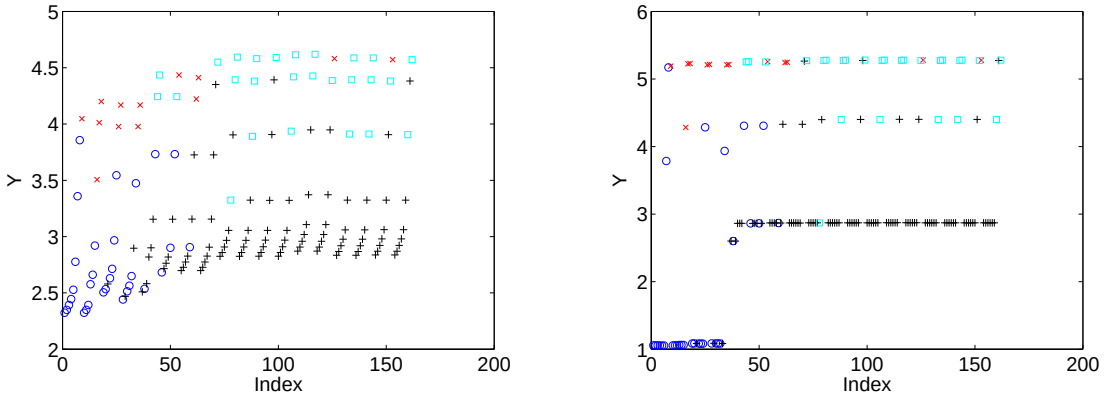


Figure 7.7: The posterior mean of each sample. The left panel corresponds to the tree model with random thresholds. The right panel corresponds to the regular tree model.

We also compare the tree samples from the tree model with random thresholds to those from the tree model without random thresholds. Figure 7.8 displays two tree samples. The tree in the left panel corresponds to the tree model with random thresholds and the one in the right panel corresponds to the regular tree model. They both have the largest log integrated likelihood in their analysis. The size of tree on the right is much larger as expected because in the regular tree model more splits are needed to achieve a good fit; this is also illustrated by the example in the right panel of Figure 7.1. Exploring other tree samples confirmed this result, that the random threshold model generates smaller and simpler trees.

7.4 Discussion

In this chapter, we introduced random threshold modeling ideas and incorporate this into our Bayesian tree model. This idea represents and error in variables concept, i.e., the uncertainty of the splitting thresholds. For each splitting rule in the tree model, although an extra variable is defined, the implementation of the

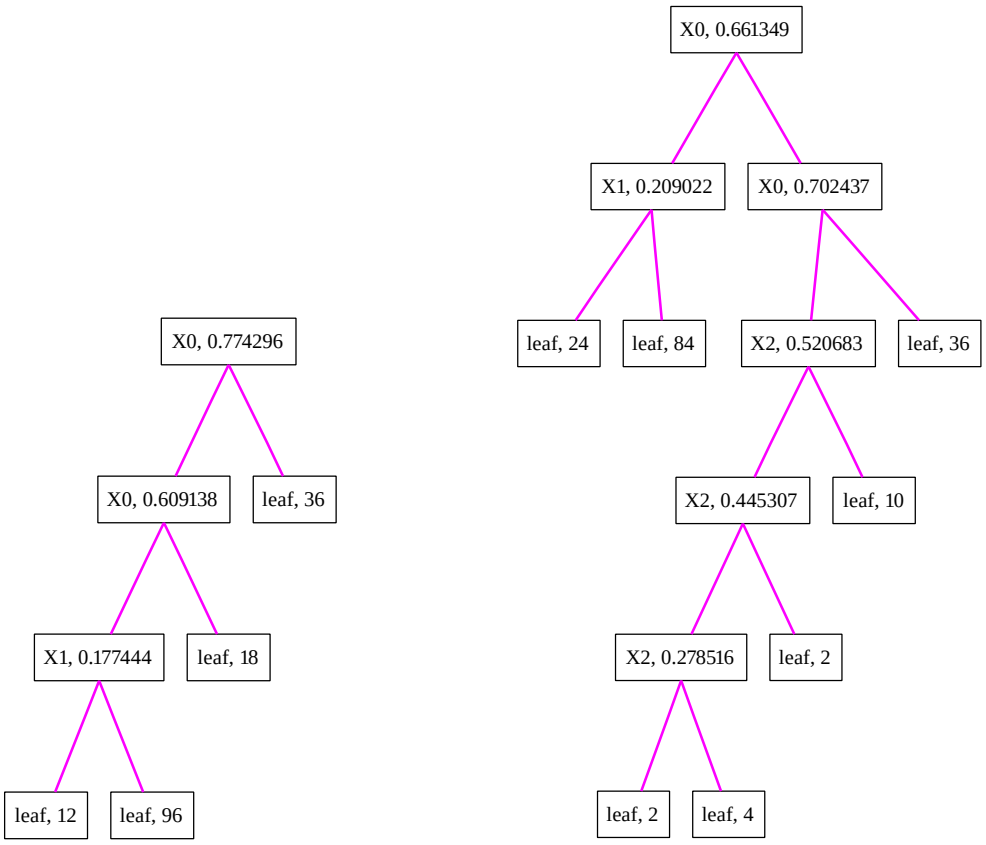


Figure 7.8: Tree samples. The left panel corresponds to the tree model with random threshold. The right panel corresponds to the regular tree model.

corresponding Metropolis-Hastings algorithm is no more complex. We only need to compute the induced marginal likelihood by averaging over random thresholds.

The basic idea of a random threshold is simple. If an observation is close to the border of the region, then it is less independent of the other observations. Furthermore, the posterior probabilities of the observation falling into each leaf may infer the distance between leaves, which corresponds to the distance between sub-regions in the partition. The distance between leaves is an interesting area for future research. In a conversation with Fabio Rigat, we have discussed the possibility of specifying a prior for such a distance. Our tree model with the random thresholds now suggests a quite novel approach.

In this present analysis, inference on random thresholds is still vague. In the posterior tree samples, different tree structure are visited and we cannot directly compare the thresholds across the trees. The focus is on computing the smooth estimates of the resulting regressions rather than finding out about the underlying "true" splitting thresholds.

As we will see in Chapter 8, this random threshold can be applied to the other tree models than the regression tree.

Chapter 8

AR Tree Models

Nonlinear time series models have been widely studied in the literature. One of the common techniques to extend linear autoregressive models to handle nonlinear time series data is thresholding. Motivated by the threshold autoregressive model (Tong, 1990), we now explore Bayesian tree models in time series.

In development of this autoregressive tree model, we found the work of Meek *et al.* (2002) of interest. Although the model specification is similar, key difference exists. Meek *et al.* (2002) conduct a Bayesian analysis for a collection of alternative model structures (trees) having unknown model parameters. However, the construction of these structures are not clear and instead of running a MCMC algorithm, Meek *et al.* (2002) evaluate the posterior probabilities for each structure. The problem is then changed to that of model selection over finite number of models. Our approach is quite different. Our autoregressive tree models focus on exploring the posterior space with the MCMC algorithm discussed in the previous chapters, and aggregating over trees for prediction. There are no pre-specified model structures. Instead we specify a prior over tree models to represent structure uncertainty.

We first review several thresholding autoregressive models and then discuss the motivation for our autoregressive tree model in details. After describing model specifications, we study two data examples. The first one is a synthetic data example, which gives some guidelines for autoregressive tree analysis. The second one is the Canadian lynx data example (Priestley, 1988). The extension to autoregressive tree models with random thresholds is discussed in the last section.

8.1 Motivation

An autoregressive model can be written as

$$y_t = \theta_0 + \sum_{i=1}^{k} \theta_i y_{t-i} + \epsilon_t \tag{8.1}$$

where $\epsilon_t \sim \mathrm{N}(0, \sigma^2)$. This is the standard AR(k) model for the $\{y_t\}$ process. A class of nonlinear time series models extending equation (8.1) is called transition autoregressive models (Tong, 1990; Chan and Tong, 1986; van Dijk *et al.*, 2002). Generally the current y_t does not simply regress on the linear combination of the past values, but rather on a mixture of two (or more) linear combinations. The way of mixing is controlled by the transition function as in

$$y_t = \theta_{01} + \sum_{i=1}^{k} \theta_{i1} y_{t-i} + (\theta_{02} + \sum_{i=1}^{k} \theta_{i2} y_{t-i}) \pi(\gamma, c, s_t) + \epsilon_t \tag{8.2}$$

where $\epsilon_t \sim \mathrm{N}(0, \sigma^2)$. Here the model for y_t is actually a mixture of two AR(k) models, specified by $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2$, where $\boldsymbol{\theta}_1 = (\theta_{01}, \theta_{11}, \ldots, \theta_{k1})$ and $\boldsymbol{\theta}_2 = (\theta_{02}, \theta_{12}, \ldots, \theta_{k2})$ respectively. $\pi(\gamma, c, s_t)$ is the transition function, where s_t includes the past values of y_t and possibly some other inputs, γ and c are the parameters to be specified. To simplify the notation, we write equation (8.2) in

matrix form, namely

$$y_t = x_t'\boldsymbol{\theta}_1 + \pi(\gamma, c, s_t)x_t'\boldsymbol{\theta}_2 + \epsilon_t \tag{8.3}$$

where $x_t' = (1, y_{t-1}, y_{t-2}, \ldots, y_{t-k})$.

Specific to different problems and models, the transition function $\pi(\gamma, c, s_t)$ has different forms. Tong (1978), Tong and Lim (1980), Tsay (1989), and Tong (1990) proposed and discussed a threshold autoregressive (TAR) model, namely

$$y_t = x_t'\boldsymbol{\theta}_1 + I[s_t > c]x_t'\boldsymbol{\theta}_2 + \epsilon_t \tag{8.4}$$

where the transition function is an indicator function which divides the space into two regimes and in each regime y_t is "locally" an AR process. s_t in the indicator function can be some other inputs than the time series data itself. If we restrict s_t to the past values of the time series data, we have the self-exciting threshold autoregressive (SETAR) model (Petruccell and Woolford, 1984; Chen and Tsay, 1991; Wong and Li, 1998), namely

$$y_t = x_t'\boldsymbol{\theta}_1 + I[y_{t-d} > c]x_t'\boldsymbol{\theta}_2 + \epsilon_t \tag{8.5}$$

where d is the delay parameter. From the view of tree models, equation (8.5) is an explicit expression of the tree model with only one split and the splitting rule is $\{y_{t-d} > c\}$. An immediate extension of this SETAR model is to admit more than two regimes. For example, one possible extension could be

$$y_t = x_t'\boldsymbol{\theta}_1 + I[y_{t-d_1} > c_1, y_{t-d_2} \leq c_2]x_t'\boldsymbol{\theta}_2 + I[y_{t-d_1} > c_1, y_{t-d_2} > c_2]x_t'\boldsymbol{\theta}_2 + \epsilon_t \tag{8.6}$$

Theoretically, we can still use the procedure described in Tsay (1989) to find the estimate for the delay parameters and the thresholds. But in that case, one

question is left unanswered. How many thresholds do we need to put into this extended SETAR model? A possible approach to proceed is to enumerate all possible ways of thresholding and compare these models according to some criterion (e.g. AIC, BIC) after the best estimate for delay parameters and thresholds are found. This approach is computationally expensive and even infeasible for the data with a complicated thresholding structure. A formal and appropriate approach is, of course, to build a thresholding structure on top of the AR model. A natural choice for such structure is the tree model. For example (8.6) can be easily described by a tree model of size 3 (two splits). In the remaining sections of this chapter, we illustrate how to synthesize Bayesian tree models with AR models in the context of non-linear time series data analysis. This model, combining tree and autoregressive model, is called autoregressive tree (ART).

In both equation (8.4) and (8.5), the transition function is discontinuous. The introduction of smoothness into the transition function leads to a model called smooth transition autoregressive model (STAR). One can refer to Bacon and Watts (1971), Chan and Tong (1986), Luukkonen *et al.* (1988), Granger and Teräsvirta (1993), Teräsvirta (1994), Wong and Li (1998), Leybourne *et al.* (1998) and van Dijk *et al.* (2002) for detailed discussions. Some examples of smooth transition functions are:

- Logistic transition function: $\pi(\gamma, c, y_{t-d}) = (1 + exp(-\gamma(y_{t-d} - c)))^{-1}$.
- Exponential transition function: $\pi(\gamma, c, y_{t-d}) = 1 - exp(-\gamma(y_{t-d} - c)^2)$.
- Second-order logistic transition function: $\pi(\gamma, c_1, c_2, y_{t-d}) = (1+exp(-\gamma(y_{t-d}-c_1)(y_{t-d} - c_2)))^{-1}$, where $c_1 < c_2$.

The smoothness parameter γ in transition functions above is positive. No matter

what the (smooth) transition function is, the basic idea of STAR and TAR models is to replace the linear combination with a non-linear function. That is, all these models can generalized as

$$y_t = f(\{y_{t-1}, y_{t-2}, \ldots\}, \boldsymbol{\theta}) + \epsilon_t. \tag{8.7}$$

In other words, the non-linearity is introduced in the expected mean of y_t given the past values, while normality still holds. Sometimes we would like to relax this restriction. In SETAR model, the mean of y_t is determined by either $\boldsymbol{\theta}_1$ or $(\boldsymbol{\theta}_1, \boldsymbol{\theta}_2)$, depending on y_{t-d}. STAR models relax this to be a weighted sum and the weight is a function of the past value. This is how the smoothness in the mean value is introduced. To further relax the restriction of normality, we assume there is an underlying random variable z_{t-d} which depends on y_{t-d} and possibly some other parameters. This z_{t-d} can be considered as the "true" transition variable. In this case, the distribution of y_t is determined by this representative z_{t-d}. If normal distribution is assumed in each region, then the distribution y_t is a normal mixture. Therefore, not only is the mean of y_t a nonlinear function of the past values, but also the distribution of y_t is non-normal. This is analogous to the random threshold introduced in Chapter 7. Therefore, if we use tree models to define a thresholding structure for a SETAR model, this extension from SETAR and STAR model is very natural in the tree model context and will be introduced in a later section.

8.2 Model specification

We will illustrate the relationship between the SETAR model and the ART model by looking at a more complicated example. Consider the following model

$$\begin{aligned} y_t &= I(y_{t-d_1} < c_1)x_t'\boldsymbol{\theta}_1 + I(y_{t-d_1} \geq c_1, y_{t-d_2} < c_2)x_t'\boldsymbol{\theta}_2 \\ &\quad + I(y_{t-d_1} \geq c_1, y_{t-d_2} \geq c_2)x_t'\boldsymbol{\theta}_3 + \epsilon_t. \end{aligned} \tag{8.8}$$

Unlike equation (8.5), where the first item $x_t'\boldsymbol{\theta}_1$ is like a base for the other regimes, equation (8.8) is specified in the way that the "AR process" in each regime is distinct from and independent of the others. This specification is introduced only to simplify the notation. At time t, given the past data $\{y_{t-1}, y_{t-2}, \ldots\}$, the space is divided into three regions ($\Omega = \{y_{t-d_1} < c_1\} \cup \{y_{t-d_1} \geq c_1, y_{t-d_2} < c_2\} \cup \{y_{t-d_1} \geq c_1, y_{t-d_2} \geq c_2\}$). This is similar to the partitioning of the feature space in the tree model. It is easy to check that the thresholding structure in equation (8.8) corresponds to the tree shown in Figure 8.1. The tree shown in Figure 8.1 is different from the regression tree or classification tree in several perspectives.

- In CART models, a predictor set is used to form the splitting rules in the tree and hence make the partition of the feature space. Here in ART models, the past data $\{y_{t-1}, y_{t-2}, \ldots\}$ serves as the predictor variables.

- The splitting rules are changing as time changes. We specify the delay parameter d_i instead of a particular variable in the splitting rules. For example, in node 0 in the tree shown in Figure 8.1, the delay parameter is d_1. So at time t, the splitting variable to be used is y_{t-d_1}.

- In the leaf node u, the specified distribution is written as AR$(k; \boldsymbol{\theta}_u)$, meaning that if y_t falls into this leaf, it has a form of $y_t = x_t'\boldsymbol{\theta}_u + \epsilon_t$. $\{y_t\}_u$, the set of

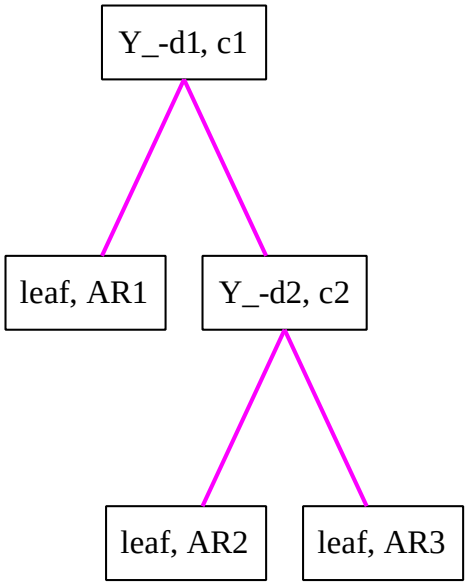


Figure 8.1: An ART example. AR1, AR2 and AR3 in the leaf node stand for AR$(k;\boldsymbol{\theta}_1)$, AR$(k;\boldsymbol{\theta}_2)$, and AR$(k;\boldsymbol{\theta}_3)$ respectively.

observations falling into leaf node u, is not necessarily an AR$(k;\boldsymbol{\theta}_u)$ process because the time index in $\{y_t\}_u$ is not sequential.

Define

$$\boldsymbol{y} = \begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_T \end{bmatrix} \tag{8.9}$$

and

$$\boldsymbol{x} = (x^1, x^2, \ldots, x^p) = \begin{bmatrix} y_k & y_{k-1} & \cdots & y_{k+1-p} \\ y_{k+1} & y_k & \cdots & y_{k+2-p} \\ \vdots & \vdots & \ddots & \vdots \\ y_{T-1} & y_{T-2} & \cdots & y_{T-p} \end{bmatrix} \tag{8.10}$$

where p is the pre-specified maximum value for the delay parameter. We hold out the first k observations, to be used as the input. Then we can directly apply the model specification for the regression tree to $(\boldsymbol{y}, \boldsymbol{x})$ with the distribution in leaf

node u specified as

$$y_t \sim \mathrm{N}(X_t'\boldsymbol{\theta}_u, \sigma^2) \tag{8.11}$$

for every (y_t, X_t), $t \in \mathcal{N}_{\mathsf{T}}(u, \mathcal{I})$, $u \in b(T)$ and $X_t = (1, y_{t-1}, y_{t-2}, \ldots, y_{t-k})$. This is a special case of equation (1.18). The likelihood function for leaf node u is

$$L(\boldsymbol{\theta}_u, \sigma^2; \boldsymbol{y}_u, \boldsymbol{X}_u) = \frac{1}{(2\pi)^{\frac{n_u}{2}}|\Sigma|^{\frac{1}{2}}}\exp(-\frac{1}{2}(\boldsymbol{y}_u - \boldsymbol{X}'_u\boldsymbol{\theta}_u)'\Sigma^{-1}(\boldsymbol{y}_u - \boldsymbol{X}'_u\boldsymbol{\theta}_u)) \tag{8.12}$$

where n_u is the number of observations in the leaf and $\Sigma = \sigma^2 I_{n_u \times n_u}$ and $(\boldsymbol{y}_u, \boldsymbol{X}_u)$ denotes all the observations falling into this leaf.

As in the other cases, we need to compute the marginal likelihood on any given tree structure. Suppose now we have M leaves, then

$$\begin{array}{rcl} \boldsymbol{y}_1 &=& \boldsymbol{X}'_1\boldsymbol{\theta}_1 + \epsilon_1, \\ \boldsymbol{y}_2 &=& \boldsymbol{X}'_2\boldsymbol{\theta}_2 + \epsilon_2, \\ &\vdots& \\ \boldsymbol{y}_M &=& \boldsymbol{X}'_M\boldsymbol{\theta}_M + \epsilon_M, \end{array} \tag{8.13}$$

where $\epsilon_u \sim \mathrm{N}(0, \sigma^2 I_{n_u \times n_u})$ independently. Let $\boldsymbol{y} = (\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_M)'$ and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_M)'$. A natural choice of the prior for $\boldsymbol{\theta}_u, u = 1, 2, \ldots, M$ and σ^2 is

$$\begin{array}{rcl} \boldsymbol{\theta}_u &\sim& \mathrm{N}(0, \frac{\sigma^2}{m} I_{q\times q}), \\ \sigma^2 &\sim& \mathrm{IG}(a, b), \end{array} \tag{8.14}$$

where m is a pre-specified hyper parameter to control the variance of $\boldsymbol{\theta}_u$ and $q = k + 1$. It is easy to show $\boldsymbol{y}|\sigma^2$ is still multivariate normal with

$$\begin{array}{rcllcl} \mathrm{E}(\boldsymbol{y}|\sigma^2) &=& \mathrm{E}(\mathrm{E}(\boldsymbol{y}|\boldsymbol{\theta}, \sigma^2)) &=& \boldsymbol{0}, \\ \mathrm{Var}(\boldsymbol{y}|\sigma^2) &=& \mathrm{E}(\mathrm{Var}(\boldsymbol{y}|\boldsymbol{\theta}, \sigma^2)) + \mathrm{Var}(\mathrm{E}(\boldsymbol{y}|\boldsymbol{\theta}, \sigma^2)) &=& \sigma^2\boldsymbol{C} \end{array} \tag{8.15}$$

where the covariance matrix is

$$\boldsymbol{C} = \mathrm{Diag}(\frac{1}{m}\boldsymbol{X}'_1\boldsymbol{X}_1 + I_{n_1\times n_1}, \frac{1}{m}\boldsymbol{X}'_2\boldsymbol{X}_2 + I_{n_2\times n_2}, \ldots, \frac{1}{m}\boldsymbol{X}'_M\boldsymbol{X}_M + I_{n_M\times n_M}) \tag{8.16}$$

From the normal/gamma theory, we know that $\boldsymbol{y}$ is marginally a multivariate t distribution in $n = \sum_{u=1}^{M} n_u$ dimensions, with $2a$ degrees of freedom, mode $\mathbf{0}$ and scale matrix $\boldsymbol{R} = \boldsymbol{C}/(ab)$ with density

$$p(\boldsymbol{y}) \quad \propto \quad (2a + \boldsymbol{y}'\boldsymbol{R}^{-1}\boldsymbol{y})^{-(2a+n)/2}. \tag{8.17}$$

In order to compute the marginal likelihood we now only need to evaluate the probability density function for observation $\boldsymbol{y}$.

This model can be relaxed by assuming different σ^2 in each leaf node. That is the distribution in leaf node u specified as

$$y_t \quad \sim \quad \mathrm{N}(X_t'\boldsymbol{\theta}_u, \sigma_u^2). \tag{8.18}$$

In this case, for example, equation (8.8) will be changed to

$$\begin{array}{rcl} y_t & = & I(y_{t-d_1} < c_1)(x_t'\boldsymbol{\theta}_1 + \epsilon_{t1}) + I(y_{t-d_1} \geq c_1, y_{t-d_2} < c_2)(x_t'\boldsymbol{\theta}_2 + \epsilon_{t2}), \\ & & +I(y_{t-d_1} \geq c_1, y_{t-d_2} \geq c_2)(x_t'\boldsymbol{\theta}_3 + \epsilon_{t3}). \end{array} \tag{8.19}$$

where ϵ_{ti}, $i = 1, 2, 3$ are normal random variables with different variance.

Similar to the prior specification in equation (8.14), the prior for $(\boldsymbol{\theta}_u, \sigma_u^2)$, $u = 1, 2, \ldots, M$ is

$$\begin{array}{rcl} \boldsymbol{\theta}_u & \sim & \mathrm{N}(0, \frac{\sigma_u^2}{m} I_{q\times q}) \\ \sigma_u^2 & \sim & \mathrm{IG}(a, b) \end{array} \tag{8.20}$$

The calculation of marginal likelihood is similar, except that in this case the marginal likelihood in each leaf can be computed independently. Similarly, $\boldsymbol{y}_u$ is marginally a multivariate t distribution in n_u dimensions, with $2a$ degrees of freedom, mode $\mathbf{0}$ and scale matrix $\boldsymbol{R}_u = \boldsymbol{C}_u/(ab)$, where $\boldsymbol{C}_u = \frac{1}{m} X_u' X_u + I_{n_u \times n_u}$. We compute the marginal likelihood $p(\boldsymbol{y}_u|\mathsf{T})$ by evaluating the corresponding probability density function of multivariate t distribution. Then the marginal

likelihood for all the observations is computed by $p(\boldsymbol{y}|\mathsf{T}) = \prod_{u=1}^{M} p(\boldsymbol{y}_u|\mathsf{T})$. In the following text, we will mainly study the case of equation (8.8).

We conclude this section by discussing forecasting. Given the data and posterior tree samples, one-step forecasting is very straightforward based on

$$p(y_{t+1}|y_{1:t}) = \int_{\mathsf{T}} p(y_{t+1}|\mathsf{T}, y_1, y_2, \ldots, y_t) p(\mathsf{T}|y_{1:t}) d\mathsf{T}. \tag{8.21}$$

However, when we want to further predict y_{t+2}, it is not that easy. There are two possible approaches. The first approach assumes y_{t+1} is unknown yet. Then the predictive density of y_{t+2} given $y_{1:t}$ is

$$p(y_{t+2}|y_{1:t}) = \int_{\mathsf{T}} p(y_{t+2}|\mathsf{T}, y_1, y_2, \ldots, y_t) p(\mathsf{T}|y_{1:t}) d\mathsf{T} \tag{8.22}$$

but the predictive distribution of y_{t+2} given $\mathsf{T}, y_{1:t}$ is unknown. In linear AR models, this is a normal distribution; however, in ART models, without the normal assumption, we know nothing about $p(y_{t+2}|\mathsf{T}, y_{1:t})$ without immense additional computations involving imputation of y_{t+1}. This relates to the second approach that assumes y_{t+1} is known. Then we have

$$\begin{aligned} & p(y_{t+2}|y_{1:t+1}) \\ = & \int_{\mathsf{T}} p(y_{t+2}|\mathsf{T}, y_{1:t+1}) p(\mathsf{T}|y_{1:t+1}) d\mathsf{T} \end{aligned} \tag{8.23}$$

In this case, the predictive distribution of y_{t+2} given $\mathsf{T}, y_{1:t+1}$ is known. However, the posterior samples T given $y_{1:t+1}$ are not available. We need to regenerate the tree samples at each step of forecasting, which is computationally expensive. In practice, for near future forecasting, we still use the tree samples from $\mathsf{T}|y_{1:t}$, simply assuming that this represents an adequate approximation to the theoretically exact $p(\mathsf{T}|y_{1:t+1})$.

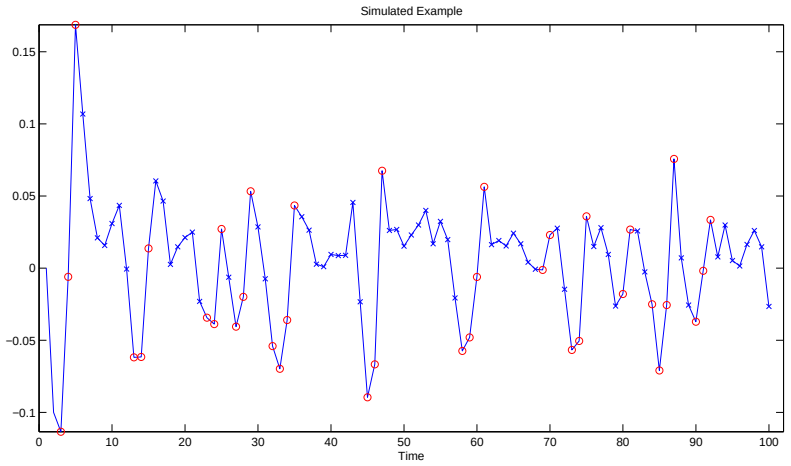


Figure 8.2: A simulated data from TAR model

8.3 Synthetic data analysis

We study two synthetic datasets. The first is simulated from a TAR model. We show how to deal with TAR models in the ART framework. The second dataset is more complicated, involving more than one threshold.

We simulate the first example dataset from the following model

$$y_t = I(y_{t-1} \leq 0)(1.2y_{t-1} - 1.5y_{t-2}) + I(y_{t-1} > 0)(0.5y_{t-1} - 0.06y_{t-2}) + \epsilon_t \tag{8.24}$$

where $\epsilon_t \sim \mathrm{N}(0, 0.02^2)$ and the initial values are $y_1 = 0$, $y_2 = -0.1$. The data is shown in Figure 8.2. The data points marked with red circle are simulated from $1.2y_{t-1} - 1.5y_{t-2} + \epsilon_t$ and those with blue cross are from $0.5y_{t-1} - 0.06y_{t-2} + \epsilon_t$.

When we analyze this data with ART models, we know that there is only one threshold. Therefore in running the MCMC algorithm, we can start from a tree with only one splitting variable (two leaves). Since the size of the tree will be kept the same, we do not need to run the grow/prune move. Swap move is not feasible

and restructure move and change move are equivalent in this case. So, basically, we just change the thresholding rule at every step. In the posterior samples, we see 1479 out of 1500 trees with the delay parameter 1. The MCMC mean and standard variance of the threshold are 0.00086 and 0.0107 respectively. We also try to run the MCMC algorithm with grow/prune move and restructure move and small trees are favored in the posterior samples.

The above analysis is very straightforward and can be easily done even without the tree structure. However, when we move to a more complicated model, it is not so obvious. We consider the following model

$$\begin{aligned} y_t &= I(y_{t-1} < -100)(-767 - 1.5y_{t-1}) + I(-100 \le y_{t-1} < 0)(1.5 - 0.5y_{t-1}) \\ &\quad + I(y_{t-1} \ge 0)(-300 + 0.5y_{t-1}) + \epsilon_t \end{aligned} \tag{8.25}$$

where $\epsilon \sim \mathrm{N}(0, 100^2)$. A simulation of 200 samples is shown in Figure 8.3. We also show a scatter plot of y_t versus y_{t-1} in Figure 8.4. It is obvious y_t is not linear in y_{t-1}. The different regimes are marked with different symbols. In this case, a simple AR model can not fit the data well. We thus use ART to explore the thresholding structure; however, p specified in equation (8.10) is usually unknown. In other words, we do not know how many steps we need to look back. Some exploratory analysis may be helpful. Figure 8.5 shows the scatter plot of y_{t-2} versus y_t. Unlike the scatter plot in Figure 8.4, it is hard to distinguish the three regions marked with different symbols. In this case, p is likely to be 1 and then we conduct the TAR analysis. However, in ART analysis we specify a large enough number for p, e.g. $p = 3$ for this data. Then we run the MCMC algorithm and check what structure is preferred in the posterior samples.

For this data, the prior for the leaf node distribution is $\boldsymbol{\theta}_u \sim \mathrm{N}(0, \sigma^2 I_{q\times q})$

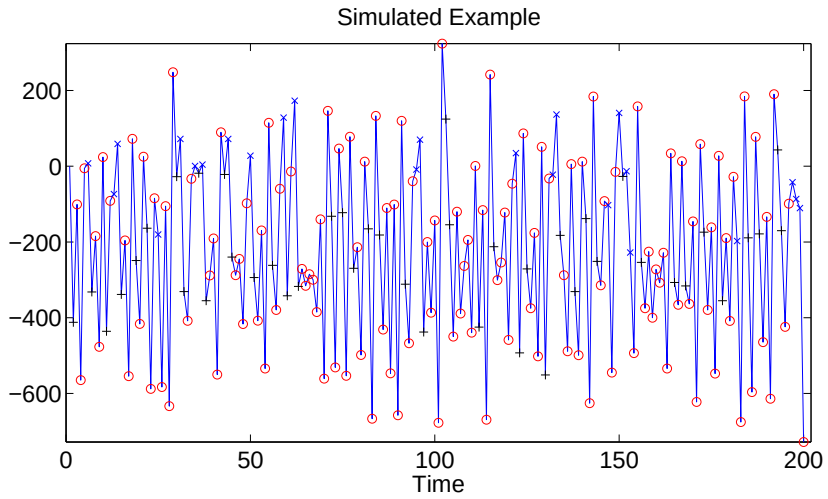


Figure 8.3: A simulation of 200 samples from model specified in equation (8.25)

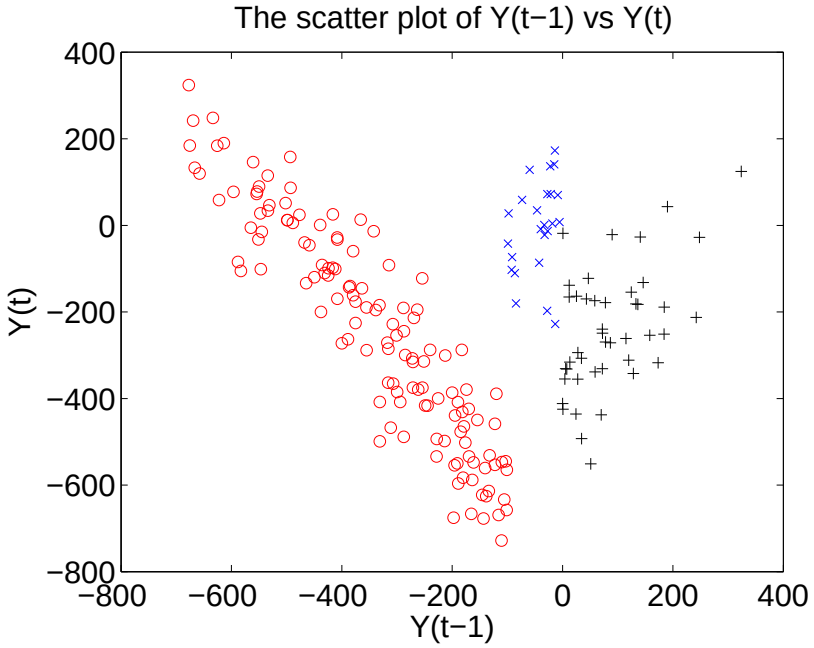


Figure 8.4: The scatter plot of y_{t-1} vs y_t. Nonlinearity is observed.

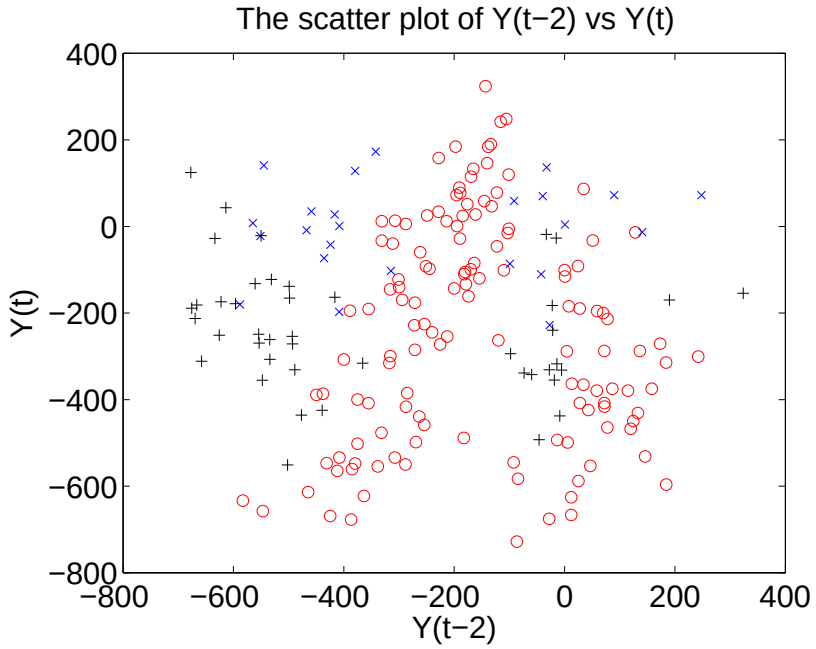


Figure 8.5: The scatter plot of y_{t-2} vs y_t. No obvious thresholding can be seen from this scatter plot.

and $\sigma^2 \sim \text{IG}(0.01, 0.01)$. The pinball prior has $\alpha(m) = 1 + \text{Pois}(m-1; 3)$ and $\beta(i; m) = 1 + \text{Bin}(i-1; m-2, \frac{1}{2})$. The splitting variables (delay parameters) are chosen uniformly via $\gamma(i) = \frac{1}{p}$, and splitting thresholds come, for all k, from $\delta_k(\cdot) = \text{Unif}(-723, 324)$, where -723 and 324 are the minimum and maximum values of y_t respectively.

We ran the MCMC algorithm using the grow/prune, swap, change and basic restructure proposals for 1000 iterations. Each iteration includes a series of 25 change moves, 25 grow/prune moves, 25 swap moves and 1 restructure move. The acceptance ratio of grow/prune and change move is about 15.5% and 18.4% respectively. The acceptance ratio of swap move is 0% and that of basic restructure move is almost 1. This is not a surprise. When we take a close look at the posterior tree samples, we find many trees similar to the two shown in Figure 8.6.

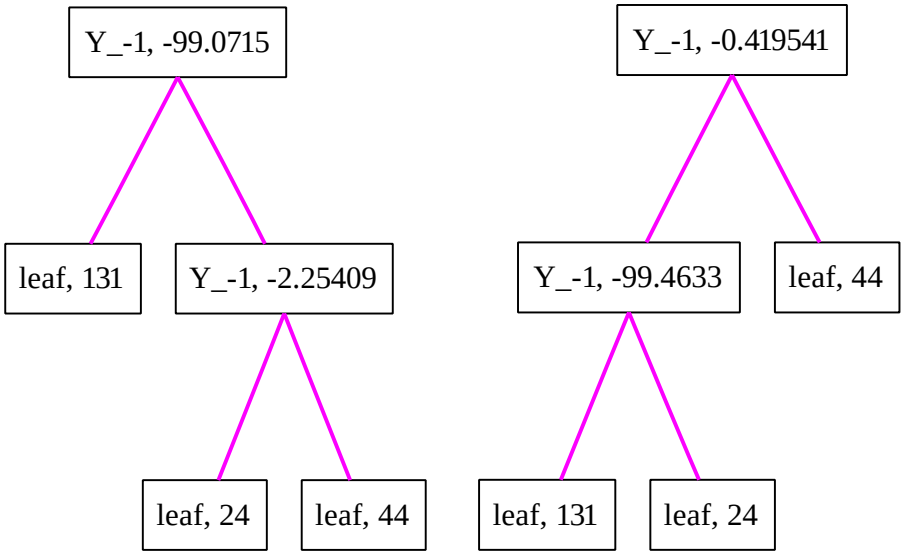


Figure 8.6: Two posterior tree samples. These are visited with almost equal probability.

It is obvious that if, for example, the tree on the left panel of Figure 8.6 is the current one, swap move will not be accepted because it will propose a new tree with an empty leaf node. As for the basic restructure move, it will propose trees between the two shown in Figure 8.6. We can see that the two tree structures in the figure are actually equivalent in terms of the thresholding structure and our basic restructure move is able to jump between them. Meanwhile, our MCMC algorithm finds some other trees as well, but most of the trees we see in the posterior samples have the structure as in Figure 8.6 and in most of the trees, although we allow delay parameter up to 3, only y_{t-1} is used, indicating that y_{t-2} and y_{t-3} are not necessary to form the thresholding structure.

Figure 8.7 shows the histogram of the log integrated likelihood for the posterior tree samples. The red dashed line indicates the log integrated likelihood of the true model. The histogram shows two modes. One is centered at the true value and the other one is slightly smaller. This indicates that our algorithm has visited

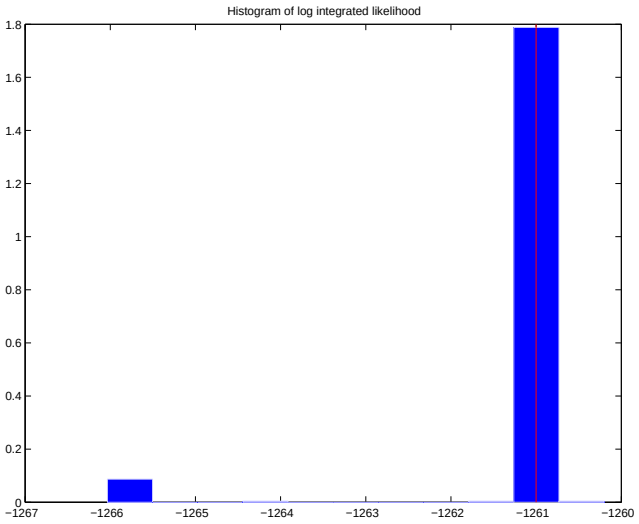


Figure 8.7: The histogram of the log integrated likelihood with the one from the true model indicated.

the other trees while the tree model corresponding to equation (8.25) is strongly preferred.

8.4 Lynx data

Now we consider a real data example. We want to model the annual number of lynx trappings in the Mackenzie River District of North-west Canada for the period 1821 to 1934 (Priestley, 1988). The logarithm with base 10 of the data is shown in Figure 8.8. We first look at the scatter plot of y_t versus y_{t-1} to check if there is any obvious structure. The scatter plot is shown in Figure 8.9. It is obvious that y_t is generally linear in y_{t-1}. The non-linearity lies in the asymmetry in the rising period (i.e. from "valley" to "peak", $y_{t-1} \geq y_{t-2}$) and the falling period (i.e. from "peak" to "valley", $y_{t-1} > y_{t-2}$). In Figure 8.9, these two periods are marked with red circle and blue cross respectively. It can be seen that

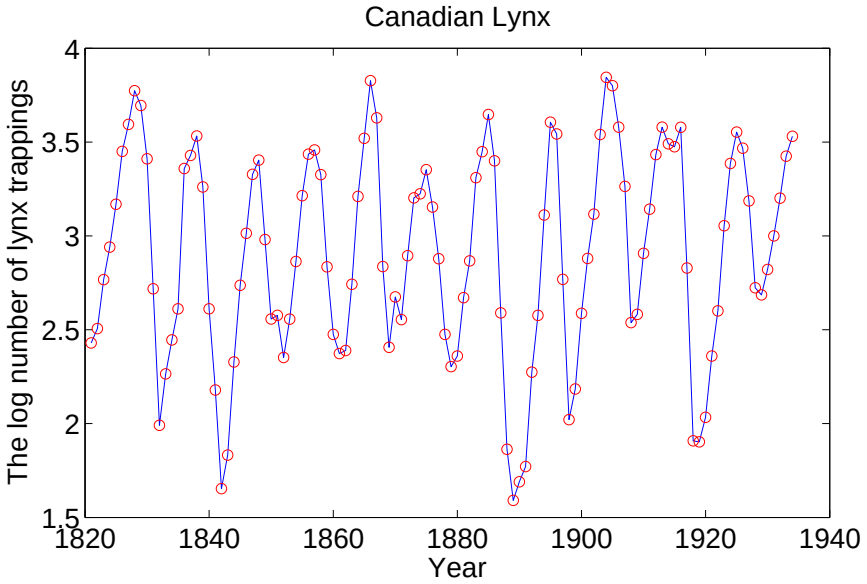


Figure 8.8: The log number of the Canadian lynx trappings for the period 1821 to 1934.

the variance of y_t given y_{t-1} in the rising period is slightly larger than that in the falling period; the time spent on the rising period is slightly higher than the time spent on the falling side.

In this section, we aim to explore this feature with our ART model. Chapter 7 in Tong (1990) is recommended for a comprehensive study of this data. In our analysis, we set $k = p = 2$, that is, at any time t we will look back up to 2 steps to find an appropriate splitting variable and in each leaf, an $AR(2)$ model is assumed. Observing the significance of $y_{t-1} - y_{t-2}$, we will also include this variable in the splitting variable set. In this case, equation (8.10) is

$$\boldsymbol{x} \;=\; (x^1, x^2, \ldots, x^p, x^1 - x^2) \;=\; \begin{bmatrix} y_k & y_{k-1} & y_k - y_{k-1} \\ y_{k+1} & y_k & y_{k+1} - y_k \\ \vdots & \vdots & \vdots \\ y_{T-1} & y_{T-2} & y_{T-1} - y_{t-2} \end{bmatrix}. \qquad (8.26)$$

We ran the MCMC algorithm and produced 1000 posterior tree samples. Fig-

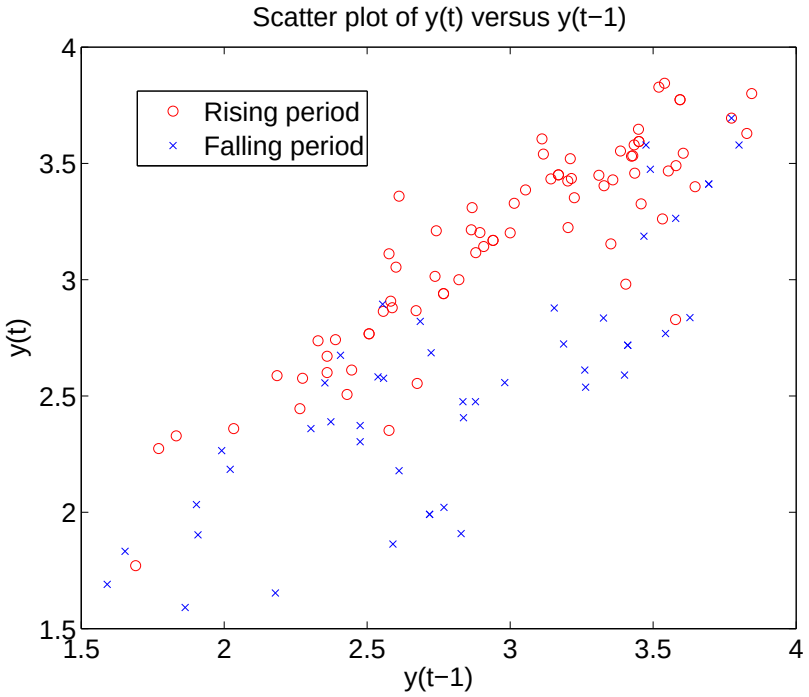


Figure 8.9: The scatter plot of y_t versus y_{t-1}. The rising period and falling period are marked with red circle and blue cross respectively.

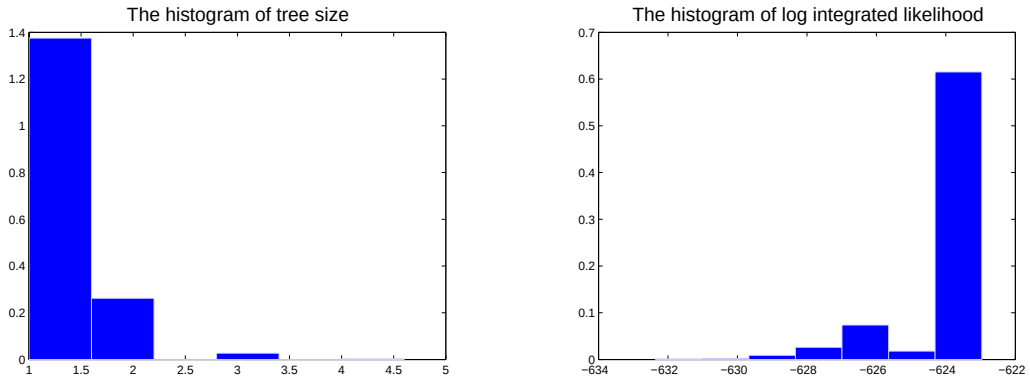


Figure 8.10: The posterior histograms of tree size and log integrated likelihood.

ure 8.10 displays the histogram of tree size and log integrated likelihood. Surprisingly, 825 out of 1000 tree samples have only one single node, which means that in our ART analysis AR(2) model is preferred in fitting the data. This partly supports what we have seen in Figure 8.9, where a strong linear relation between y_t and y_{t-1} was shown. However, non-linearity is evident as now discussed.

Taking a closer look at the posterior tree samples, we find that some more complicated models have also been visited. Two tree samples are shown in Figure 8.11. They are both trees with only one splitting variable. In the left panel, the splitting variable is y_{t-3}, which is $y_{t-1} - y_{t-2}$ in our specification. The resulting two subset of observations are very close to the division between rising period and falling period. The tree displayed in the right panel has a different splitting variable, which is y_{t-2}.

This ART analysis in this example illustrates the use of the approach as an exploratory tool to study non-linearity in time series data. There are many question left unanswered and this first extension of Bayesian tree models to ART modeling is just an initial, opening step. For example, have we run the posterior MCMC algorithm long enough? How do we choose k and p? These questions will be

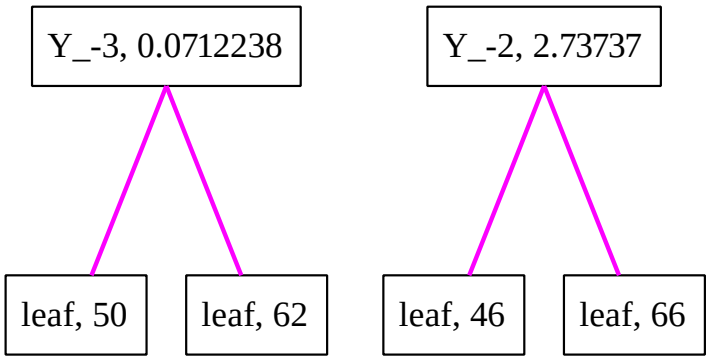


Figure 8.11: Two tree samples

studied as we further develop this model.

8.5 Extension

We briefly discuss one extension of ART models, that of random thresholds. The observation and the splitting variable are specified as equation (8.9) and (8.10). In addition to that, we have a set of underlying variable $\boldsymbol{z} = (z^1, z^2, \ldots, z^p)$. As in the random threshold model, the given splitting variable set is the "noisy" representation of $\boldsymbol{z}$ and the true splitting variables are $\boldsymbol{z}$. Similar to the discussion in Chapter 7, the only change to the implementation of ART model is the calculation of the likelihood. We integrate out all the underlying variables in order to calculate the marginal likelihood.

Consider a simple example specified in equation (8.5). Suppose y_{t-d} is a noisy splitting variable. It follows a normal distribution with mean y^0_{t-d} and variance

γ^2. The marginal likelihood is

$$\begin{aligned} & p(y_t|x_t, y_{t-d}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, c, \sigma^2, \gamma^2) \\ = & \int_{y^0_{t-d}} p(y_t|x_t, y^0_{t-d}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, c, \sigma^2, \gamma^2) p(y^0_{t-d}|y_{t-d}) dy^0_{t-d} \\ = & \phi(y_t; x_t'(\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2), \sigma^2)(1 - \Phi(y_{t-d}|c, \gamma^2)) + \phi(y_t; x_t'\boldsymbol{\theta}_2, \sigma^2)\Phi(y_{t-d}|c, \gamma^2) \end{aligned} \tag{8.27}$$

where $\phi(x; \mu, \sigma^2)$ and $\Phi(x; \mu, \sigma^2)$ are the normal probability density function and the normal cumulative distribution function respectively. Thus the mean $\mathrm{E}(y_t|x_t, y_{t-d}, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, c, \sigma^2, \gamma^2)$ is

$$x_t'(\boldsymbol{\theta}_1 + \boldsymbol{\theta}_2)(1 - \Phi(y_{t-d}|c, \gamma^2)) + x_t'\boldsymbol{\theta}_2\Phi(y_{t-d}|c, \gamma^2) \tag{8.28}$$

which can be simplified to

$$x_t'\boldsymbol{\theta}_1(1 - \Phi(y_{t-d}|c, \gamma^2)) + x_t'\boldsymbol{\theta}_2. \tag{8.29}$$

This mean is a specific case of the mean in equation (8.3). The transition function is in the form of normal CDF. It is easy to check that different transition functions, e.g. exponential transition function and logistic transition function, correspond to different distribution assumptions for the underlying splitting variable. The key difference between ART with random threshold and STAR is that the normal assumption in STAR does not hold in our model. Therefore in analyzing nonlinear time series data, the normal assumption can be relaxed with ART model with random threshold.

Appendix A

SimTree Manual

In this appendix, we describe the implementation of our Bayesian tree model in C++ and demonstrate the usage of the code SimTree. SimTree is a collection of C++ code with object oriented design. With this design, it is easy to write new code for new applications. For example, we need to include just one new class for survival tree.

A.1 Classes

In this section, we describe the several key classes: *Node*, *Model*, *Proposal*, *MCMC* and *Diagnose*. Figure A.1 displays the complete class hierarchy in the implementation of SimTree.

A.1.1 Node

Node class defines a node in the tree. It has the following member variables.

- left, right: the pointers to the left child node and right child node.
- obs: the pointer to the observation class, which stores the data.

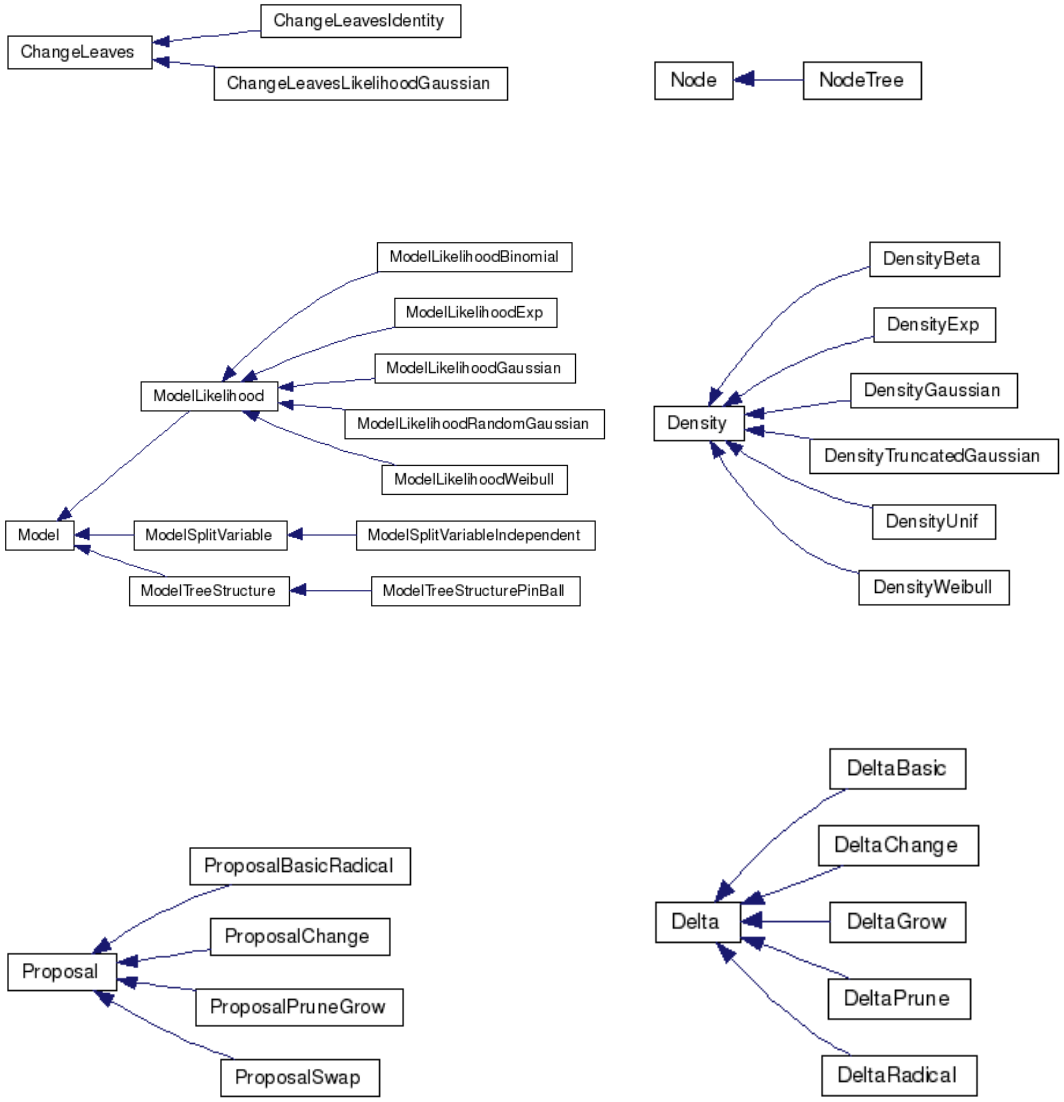


Figure A.1: The class hierarchy in the implementation of SimTree.

- splitVariable, splitLevel: splitting rules assigned to this node. If splitVariable is -1, this node is a leaf node.
- SubjectList: the index of observations classified to this node.

Several key member functions includes

- DumpDotFile: Output the subtree starting from this node to a dot file, which will be used to draw a tree with *dot* program.
- UpdateSubjectList: if the splitting rules is changes, update the subject list.

NodeTree is a derived class of Node. It represents the whole tree/subtree. A key member function is Pot. Given the model for tree structure, splitting variables and leaf node distribution, it calculates the negative log posterior probability.

A.1.2 Model

Model class is designed to describe the model for tree structure, splitting variables and leaf node distribution. Key member functions include

- Potential: a virtual function. It calculates the negative log prior probability.
- PotentialDifference: a virtual function. It calculates the change of negative log prior probability given the changes to the tree.

ModelLikelihood, ModelSplittingVariable and ModelTreeStructure are derived classes of Model. ModelLikelihood describes the leaf node distribution. In the current development, we implement ModelLikelihoodBinomial, ModelLikelihoodExp, ModelLikelihoodGaussian, ModelLikelihoodRandomGaussian and ModelLikelihoodWeibull, corresponding to Binomial distribution, Exponential distribution, Nor-

mal distribution, Normal distribution with random threshold and Weibull distribution in the leaf node. When a new model is developed, we only need to implement a new class derived from ModelLikelihood and implement those virtual functions calculating the negative log marginal likelihood.

ModelSplitVariable is a derived class of Model to describe the prior for the splitting rules. In the current development, we have the implementation of ModelSplitVariableIndependent, corresponding to the independent prior specifications for splitting rules.

ModelTreeStructure is a derived class of Model to describe the prior for the tree structure. In the current development, we have the implementation of ModelTreeStructurePinball, corresponding to the pinball prior for tree structure described in section 2.1.

A.1.3 Proposal

Proposal class is designed to describe several proposals in the Metropolis-Hastings algorithm in chapter 3. It has only one member function ProposeChange, which proposes changes to the tree and return a Delta class (describing the change).

ProposalBasicRadical, ProposalChange, ProposalPrunegrow and ProposalSwap are the derived classes corresponding to restructure move, change move, grow/prune move and swap move.

A.1.4 MCMC and diagnosis

MCMC class runs the Metropolis-Hastings algorithm given the input of proposals and models for the tree. Acceptance ratio are recorded and return. MCMC itself does not records all the trees in the chain. So after each iteration, we need to

store the tree in a vector.

Diagnose class provides a collection of functions, calculating the importance of splitting variables, negative log posterior probability, the predicted values in leave one out cross validation, etc., given the input of posterior tree samples.

A.2 Usage of SimTree

The code is successfully compiled and run with g++ 4.0 and Visual C++ 8.0. An example of running SimTree is

Example

```
./SimTree seed DataFile OutputPrefix
```

The running arguments are

- seed: seed number for random number generator.
- DataFile: the input of data. The first line specifies the type of distribution in the leaf node, defined in the header file Observation.h. The second line specifies the number of observations and the third line specifies the number of predictors. From the fourth line, each line is an observation in the format of $x_1, x_2, \ldots, x_p, y$.
- OutputPrefix: the prefix for a collection of output files. The output files includes the importance table of predictors, log marginal likelihood for each tree, predicted values of leave one out cross validation, etc.. Detailed output can be found in Diagnose::Scoring.

The software is available for download on `http://www.isds.duke.edu/~casper/`.

datainput.txt

```
0
5
2
0.1 0.2 0.12
0.2 0.1 0.14
0.4 0.4 0.50
0.1 0.1 0.13
0.2 0.4 0.30
```

Table A.1: A sample data input file.

Bibliography

Bacon, D. W. and Watts, D. G. (1971). Estimating the transition between two intersecting straight lines. *Biometrika* **58**, 525–534.

Breiman, L., Friedman, J., Olshen, R., and Stone, C. (1984). *Classification and Regression Trees*. The Wadsworth Statistics/Probability series, Belmont CA.

Brooks, S. P. and Roberts, C. P. (1998). Convergence assessment techniques for Markov chain Monte Carlo. *Statistics and Computing* **8**, 319–335.

Buntine, W. (1992). Learning classification tree. In *Statistics and Computing*, vol. 2, 63–73. Springer Netherlands.

Campa, M. J., Fitzgerald, M. C., and Patz, Jr., E. F. (2003a). Exploring the proteome with MALDI-TOF (editorial). *Proteomics* **3**, 9, 1659–1660.

Chan, K. S. and Tong, H. (1986). On estimating thresholds in autoregressive models. *Journal of Time Series Analysis* **7**, 179–190.

Chen, R. and Tsay, R. S. (1991). On the ergodicity of TAR(1) processes. *The Annals of Applied Probability* **1**, 4, 613.

Chipman, H., George, E., and McCulloch, R. (1998). Bayesian CART model search (with discussion). *J. Am. Statist. Ass.* **93**, 935–960.

Chipman, H., George, E., and McCulloch, R. (2002). Bayesian treed models. *Machine Learning* **48**, 299–320.

Conover, W. (1971). *Practical Nonparametric Statistics*. John Wiley & Sons, New York.

Cowles, M. K. and Carlin, B. P. (1996). Markov chain Monte Carlo convergence diagnostics: A comparative review. *J. Am. Statist. Ass.* **91**, 883–904.

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

Faraway, J. J. (2005). *Extending Linear Model With R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. Routledge.

Ferguson, T. S. (1973). A Bayesian analysis of some nonparametric problems. *Annals of Statistics* **1**, 209–230.

Franzen, J. (1997). Improved resolution for maldi-tof mass spectrometers: A mathematical study. *International Journal of Mass Spectrometry and Ion Processes* **164**, 1, 19–34.

Granger, C. W. and Teräsvirta, T. (1993). *Modeling Nonlinear Economic Relationships*. Oxford: Oxford University Press.

Green, P. J. (1995). Reversible jump Markov chain Monte Carlo computation and Bayesian model determination. *Biometrika* **82**, 4, 711–732.

Leybourne, S., Newbold, P., and Vougas, D. (1998). Unit roots and smooth transitions. *Journal of Time Series Analysis* **19**, 1, 83–97.

Luukkonen, R., Saikkonen, P., and Teräsvirta, T. (1988). Testing linearity against smooth transition autoregressive models. *Biometrika* **75**, 491–499.

Meek, C., Chickering, D., and Heckerman, D. (2002). Autoregressive tree models for time-series analysis. In *Proceedings of the Second International SIAM Conference on Data Mining*, Arlington, VA, SIAM.

Petruccell, J. D. and Woolford, S. W. (1984). A threshold AR(1) model. *Journal of Applied Probability* **21**, 2, 270–286.

Priestley, M. (1988). *Non-linear and Non-stationary Time Series Analysis*. Academic Press, London.

Teräsvirta, T. (1994). Specification, estimation, and evaluation of smooth transition autoregressive models. *Journal of the American Statistical Association* **89**, 425, 208–218.

Tong, H. (1978). On a threshold model. In C. Cheng, ed., *Pattern recognition and signal processing*, 101–141. Amsterdam: Sijthoff and Noordhoff.

Tong, H. (1990). *Non-linear Time Series: A Dynamic System Approach*. Oxford University Press.

Tong, H. and Lim, K. S. (1980). Threshold autoregression, limit cycles and

cyclical data (with discussions). *Journal of the Royal Statistical Society, Series B (Methodological)* **42**, 3, 245–292.

Tsay, R. S. (1989). Testing and modeling threshold autoregressive process. *Journal of the American Statistical Association* **84**, 431–452.

van Dijk, D., Teräsvirta, T., and Franses, P. H. (2002). Smooth transition autoregressive models - a survey of recent developments. *Econometrics Reviews* **21**, 1–47.

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

Wong, A. C. and Li, W. (1998). A note on the corrected Akaike information criterion for threshold autoregressive models. *Journal of Time Series Analysis* **19**, 113.

Biography

I was born in Guandong, China, on the 21st of August 1979. I was a honorable graduate from Beijing University (aka Peking University) with a bachelor degree in Statistics and a bachelor degree in Economics in 2002. I received a Chancellor fellowship to study for Ph.D. in Statistics at University of California, Los Angeles in 2002. In 2003, I enrolled in the Ph.D. program at ISDS in the summer of the year 2003 and became a Ph.D. candidate in the spring of 2004.