TREE-BASED METHODS FOR SURVIVAL ANALYSIS AND HIGH-DIMENSIONAL DATA

Ruoqing Zhu

A dissertation submitted to the faculty of the University of North Carolina at Chapel Hill in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Biostatistics.

Chapel Hill
2013

Approved by:

Dr. Michael R. Kosorok
Dr. Jianwen Cai
Dr. Jason P. Fine
Dr. Donglin Zeng
Dr. Stephen R. Cole
Dr. Yufeng Liu

Abstract

RUOQING ZHU: Tree-based methods for survival analysis and high-dimensional data
(Under the direction of Dr. Michael R. Kosorok)

Machine learning techniques have garnered significant popularity due to their capacity to handle high dimensional data. Tree-based methods are among the most popular machine learning approaches. My dissertation aims on improving existing tree-based methods and developing statistical framework for understanding the proposed methods. It contains three topics: recursively imputed survival tree, reinforcement learning trees and reinforcement learning trees for right censored survival data. A central idea of my dissertation is focused on increasing the chance of using signaled variables as splitting rule during the tree construction while not loosing the randomness/diversity, hence a more accurate model can be built. However, different methods achieve this by using different approaches. Recursively imputed survival tree recursively impute censored observations and refit the survival tree model. This approach allows better use of the censored observations during the tree construction, it also changes the dynamic of splitting rule selections during the tree construction so that signaled variables can be emphasized more in the refitted model. Reinforcement learning trees takes a direct approach to emphasize signaled variables in the tree construction. An embedded model is fitted at each internal node while searching for splitting rules. The variable with the largest variable importance measure is used as the splitting variable. A new theoretical framework is proposed to show consistency and convergence rate of this new approach. In the third topic, we further extend reinforcement learning trees to right censored survival data. Brier score is utilized to calculate the variable importance measures. We also show a desirable property of the proposed method that can help correct the bias of variable importance measures when correlated variables are present in the model.

I dedicate this dissertation work to my parents,

Dr. Lixing Zhu and Qiushi Tian,

who have loved and supported me throughout my life,

and to my beloved wife,

Xian Cao,

who stood by me through the good times and bad.

Acknowledgments

My graduate experience at University of North Carolina at Chapel Hill has been an amazing journey. I am grateful to a number of people who have guided and supported me throughout the research process, and cheered me during my venture.

My deepest gratitude is to my advisor, Dr. Michael R. Kosorok, for his guidance, support, patience, and also the freedom he gave me to explore on my own. I have been very fortunate to have an advisor like him. And I would not have been able to achieve this accomplishment without him.

I gratefully thank Dr. Donglin Zeng for his tremendous help in my dissertation. His patience and experience helped me overcome many difficult problems.

I would also like to thank my committee members Dr. Jianwen Cai, Dr. Jason P. Fine, Dr. Stephen R. Cole and Dr. Yufeng Liu for their insightful comments and constructive criticisms at different stages of my research. These comments motivated many of my thinking.

I am grateful to Dr. Haibo Zhou who supported me in my first year. The experience of collaboration under his guidance was invaluable.

I would like to thank Dr. Kristen Hassmiller Lich and Elizabeth Holdsworth La. My collaboration with them has been a very enjoyable part of my graduate study.

I am also thankful to Dr. Hongtu Zhu, Dr. Wei Sun, Dr. Fei Zou, Dr. Michael Wu, and all other faulty members, students and staff in the department of biostatistics.

This department provides the ideal environment for learning and doing research, it is the best in the world!

Finally, my family has supported and helped me along the course of this dissertation by giving encouragement and providing emotional support I needed. To them, I am eternally grateful.

Table of Contents

List of Tables

List of Figures

Chapter 1

Literature Review

1.1 Introduction

The decision tree is a predictive model in machine learning. In 1984, Breiman et al. published the classic book "Classification and Regression Trees"(CART) which introduced this idea to the statistics and data mining communities. This nonparametric model recursively partitions the training dataset and builds an accurate, flexible and easy-to-interpret model. Through the work of many researchers, tree-based methods have progressed significantly. During the last decade, after the "bagging" idea was introduced by Breiman (1996), ensemble tree methods have provided much more accurate models. And random forests (Breiman, 2001) has become a state-of-the-art method in machine learning. However, the asymptotic behavior of tree-based methods is still puzzling, resulting in many difficulties in deriving consistency results and prediction error calculations. Meanwhile, adaption of tree-based methods for survival analysis has drawn a lot interests. Much research has focused on tree building and dealing with censoring. In this dissertation, I attempt to answer these previously raised questions and challenges and improve upon existing methods. We will briefly review related works in the following sections. Some details are deferred to later sections when we presenting each specific method.

1.2 Tree-based methods

Tree-based methods originate from decision trees, which are commonly used tools in operation research. Based on initial work by Hunt et al. (1966) and many others, Quinlan (1983) invented the Iterative Dichotomiser 3 (ID3) algorithm to generate decision trees. This method utilizes a binary splitting mechanism and the idea of entropy, or information gain. This algorithm later on led to the trade mark algorithm C4.5, which is one the most famous decision tree learners. An independent work by Breiman et al. (1984) introduced Classification and Regression Trees (CART) to the statistics community. This method immediately drew a lot of interests in the statistics research community because this tree-based method is fully non-parametric, highly predictive, and easy to interpret. There are many versions of tree based methods, their differences primarily lie on splitting rule, pruning mechanism, and the use of ensembles and randomization.

1.2.1 Single tree model

1.2.1.1 Building a tree model

The ideas of the two classic tree-building algorithms, CART and C4.5, are very similar. Suppose we want to predict a class or continuous response Y from input features $(X_1, X_2, ..., X_p)^\top$ where p can be large but finite, we grow a binary tree by recursively splitting the training data. At a root node of the tree, which contains all training samples, a splitting variable X and a splitting value c are created to split all samples into two disjoint subsets by identifying the indicator function $I(X < c)$ for each sample. The two resultant subsets are called daughter nodes of the current node. This process is recursively applied to each of the two daughter nodes and their subsequent nodes until the sample size of a node is sufficiently small. This method eventually creates disjoint subsets (terminal nodes) in the predictor feature space $\mathcal{X}$, and predictions can be uniquely determined by identifying which terminal node the test sample belongs to. The prediction for regression modeling is obtained by averaging the training samples in that terminal node. In classification, however, the most prevalent class label is used. A graphic demonstration of the fitted model looks like an upside-down growing tree, by which the method was named. From a practical prospective, this type of model is easy to interpret because of its categorizing nature that each terminal node assigns a single prediction value to a subspace of the feature space. Moreover, since it is fully non-parametric, trees generally require fewer assumptions than classical methods and handle a wide variety of data structures.

1.2.1.2 Tree pruning

However, problems also arise even as the method looks appealing. When noise is large comparing to the true signal, or when there are unmeasured factors (Mingers, 1989), the simple tree method is likely to produce false splitting rules near the terminal node, and the entire tree tends to be large and complicated. Hence over-fitting often occurs and leads to large prediction errors. A natural solution to this is to delete all subsequent nodes from an internal node, and use all data within that branch as a single terminal node. Tree pruning procedures were thus introduced (Quinlan, 1993; Breiman et al., 1984) to reduce the size of the tree, diminish over-fitting and reduce prediction error. There are two ways to serve the purpose of pruning, one is a stopping criteria which prevents some nodes from being split further, and the other one removes nodes after the decision tree is produced. Most authors prefer the latter since it allows potential interaction structures being built before deciding whether a branch of the tree is worth keeping (Kohavi and Quinlan, 2002).

There are several pruning methods in the literature, including cost-complexity pruning, critical value pruning, pessimistic pruning, MDL pruning and many others. Most of the pruning algorithms aim at reducing the misclassification error in a classification problem. Breiman et al. (1984) proposed cost-complexity pruning, which, as its name suggested, takes both the cost (prediction error) and the complexity (size) of the tree into consideration and gives an overall score for further selection; Critical value pruning (Mingers, 1987) calculates a goodness-of-split measure, where the value of the measure reflects how well the chosen attribute splits the data. By setting a critical value on this measurement, any branch in the tree that does not reach the critical value will be pruned and become a terminal node. Pessimistic error pruning proposed by Quinlan (1986) is a different type of pruning procedure that does not require a test data-set. It utilizes a binomial distribution to obtain an estimate of the misclassification rate. Although the statistical justification of this method is dubious, it does have some advantages over other methods in the early history of the development of tree methods. Minimum Description Length (MDL) pruning was proposed by Mehta et al. (1995). They define a measurement which involves the code length, or the stochastic complexity, which has specific optimality properties (Rissanen, 1996). LeBlanc and Tibshirani (1998) also suggested to use lasso on CART which leads to both shrinking of the node estimates and pruning of branches in the tree. This method performs better than cost-complexity pruning in some problems. For a comparison of many different tree punning methods and also many other issues on this topic, please refer to Mingers (1989); Niblett (1987); Quinlan (1987).

1.2.2 Ensemble methods

A single tree-model can be easily interpreted, however, it suffers in terms of accuracy. It can be viewed as a histogram estimator where the variables and their bandwidth are locally and adaptively chosen. In early 1990's, the research community found that learning and combining multiple versions of the model can substantially improve classification error rate as compared to the error rate obtained by learning a single model of the data (Ali and Pazzani, 1996; Kwok and Carter, 1990). Breiman (1996) proposed a bootstrap aggregating procedure called "bagging". This bagging predictor takes multiple versions of the bootstrap sample (Efron and Tibshirani, 1993) from the training dataset and fits an unpruned single tree model to each bootstrap sample. The final predictor is obtained by averaging over different versions of the model. This procedure works surprisingly well and out performs its competitors in most situations. Some intuitive explanations of how and why this works were given in Breiman (1998): "Some classification and regression methods are unstable in the sense that small perturbations in their training sets or in construction may result in large changes in the constructed predictor ... ," however, "unstable methods can have their accuracy improved by perturbing and combining, that is, generating multiple versions of the predictor ... ." This idea has motivated much subsequent work including random forests (Breiman, 2001), a general framework for tree ensembles.

1.2.2.1 Splitting rules and variant of random forests

In the original CART or bagging approach, the splitting point is searched throughout the entire possible range of all variables to produce the most distinct daughter nodes. However, this may lead to a model that strongly leans to the training data set and the prediction error could be large due to overfitting. Pruning is one way to solve this, however, introducing randomization into the splitting criteria can solve the problem from another angle (see Bauer and Kohavi (1999) for a survey). Ali and Pazzani (1996) replaced using the best split by randomly selecting from the best ones with probability proportional to the test scores. Another similar idea was proposed by Dietterich (2000), which randomly selects from K best splits. Ho (1998) proposed a random subspace method which selects a random subspace of the feature space to grow the tree. Amit and Geman (1997) search over a random selection of features for the best split at each node. Breiman's random forests was largely influenced by these works, especially Amit and Geman (1997). In random forests, a random subset (*mtry*) of features is selected at each internal node, then the best split, which produces the best score, is selected as the splitting rule. In regression modeling, variance reduction is used to calculate the score, while in classification modeling, Gini index is commonly used.

Variants of random forests differ in their choices of splitting rules. Geurts et al. (2006) proposed to use a different cutting point generating method which leads to computational advantages. In their proposed extremely randomized trees, a random cutting point is generated for each selected feature, and the splitting rule is decided by choosing the best among them. Comparing to searching the best cutting point for each feature, this method substantially reduces computational intensity. Recent methods (Ishwaran et al., 2008) further extended this idea by generating multiple cutting points (*nsplit*) for each feature and comparing different splits. In our proposed methods, randomized splitting rule is implemented due to its computational advantages.

When sample size in an internal node is sufficiently small (less than a pre-specific number *nmin*), splitting will stop and conclude the current node as a terminal node. An interesting idea was proposed by Cutler and Zhao (2001). In their paper, each individual tree classifier is a perfectly-fit with only one sample in each terminal node. Although each tree apparently overfits the training data, forest averaging diminishes this drawback. From this point of view, tree pruning procedure becomes less important, however, some forest pruning methods such as Martínez-Muñoz and Suárez (2006) and Caruana et al. (2004) can save computational cost. An other variant of random forests proposed by Chipman et al. (2010) select prior information to optimize the tree fitting result. A sequence of MCMC draws of single trees is averaged to obtain the posterior inference.

1.2.2.2 On randomization

It is now generally acknowledged by the research community that a certain level of randomization along with constructing ensembles in tree-based methods can substantially improve performance. A concern addressed by many researchers was the instability of each unpruned tree (Hothorn et al., 2004). However, the idea of randomization relieves this concern: As enlighten by Breiman (1998), perturbing single trees and taking averages over forests can substantially increase performance. It is actually the independence between each tree that helps diminish the over all averaged instability. Simulation results from Cutler and Zhao (2001) reveal that one reason why their proposed method, PERT, works well although individual tree classifiers are extremely weak. The reasoning behind the extremely randomized trees (ERT) method is almost the same: although the entire sample is used to fit each tree, the dependence between any two individual trees are very weak since the splitting value is drawn at random.

Randomization can be achieved by different manners in a tree fitting procedure. Bootstrap aggregation, random selection of features and random cutting point generation all increase the randomness or reduce the dependence of fitted trees. On the other hand, greediness (pursuing signal) is equally important since over-randomized trees (such as purely random forests proposed by Breiman (2004)) is inefficient in detecting signals. Hence an accurate ensemble tree model has a good blend of randomness and greediness.

1.2.2.3 Tuning parameters

The aforementioned tuning parameters such as *mtry*, *nmin* and *nsplit* play an important role in the performance of tree-based models. *mtry* largely controls the diversity of trees. A large *mtry* compares almost all features and, with a high probability, uses the same variable to construct splitting rules in the early stage of a tree. *nmin* controls the depth of each tree. Selecting a small *nmin* is oftentimes beneficial, however, theoretical results show that *nmin* should also grow with sample size n. *nsplit* is also related to diversity since a large *nsplit* is equivalent to an exhaustive search for cutting points. In our simulation studies, we always use the same parameter tuning so that the results are comparable. However, we also demonstrate that the advantage of reinforcement learning trees is beyond the reach of parameter tuning.

1.3 Theoretical justification

One of the major challenges of tree-based methods is analyzing its asymptotic property, especially for the original random forests proposed by Breiman (2001). The greedy splitting rule selection (due to the exhaustive search of cutting point) causes trouble in formalizing the constructed trees. A much simpler model called purely random forests (Breiman, 2004) is widely considered for analyzing the asymptotic behavior of ensemble tree-based models. Basing on this simple model, Lin and Jeon (2006) established the connection between random forests and nearest neighborhood estimation. They also established a lower bound on the convergence rate of random forests under a special type of tree construction mechanism. Biau et al. (2008) showed consistency of ensemble tree-based methods and some variants of random forests. However, they also gave an interesting counterexample, under which random forests may not be consistent. This counterexample is based on a checkerboard model (also considered by Kim and Loh (2001)), which blinds the marginal signal search in a tree construction.

Following the discussions by Lin and Jeon (2006) and Breiman (2004) on a special type of purely random forest, Biau (2012) proved consistency and showed that the convergence rate only depends on the number of strong variables which, collectively and completely define the true model structure. The proof for the convergence rate result in his paper can serve as a guideline for future analysis of random forests under more general structures. However, behind this celebrated result, two key components require a careful further investigation. First, the probability of using a strong variable to split at an internal node depends on the within-node data (which possibly depends on an independent sample as suggested in Biau (2012)). With rapidly reduced sample sizes toward terminal nodes, this probability is unlikely to behave well for the entire tree. However, a large terminal node size is likely to introduce increased bias which may also harm the error rate. Second, identifying strong variables in a high dimensional surface can still be very tricky. The counterexample of consistency given by Biau et al. (2008) can potentially lead to blinding of the selection criteria so that strong variables may not be chosen. The rationale behind the above argument is that one cannot fully explore a high dimensional surface from a viewpoint which only assesses the marginal effect of each variable. Hence if the marginal effect of a strong variable is behaving like a noise variable, then the selection process may fail.

Our proposed reinforcement learning trees (RLT) takes a step forward toward understanding the asymptotic behavior of tree-based methods and building more accurate models, especially in high-dimensional settings. By fitting an embedded model at each internal node, the variable(s) with the largest variable importance (carries the most signal) will be used to split. This creates an advantage in formalizing the splitting pass from the root node to a terminal node. The simulation studies also show that RLT and its extension to survival data can easily detect an interaction model such as the checkerboard structure.

1.4 Extending tree-based methods to censored survival data

Tree-based survival models offer a more flexible model structure comparing to relatively restricted parametric models such as the popular Cox model or Accelerate failure time model. Ciampi et al. (1988), Segal (1988), and LeBlanc and Crowley (1992) provided early efforts to adapt tree-based methods to right censored survival data. Logrank test statistic is a natural choice for evaluating and comparing possible splitting rules in many of these early methods, although others used criteria such as distance measures between Kaplan-Meier curves or model based statistics. Tree pruning procedure is also a necessary and beneficial procedure to prevent overfitting. However, the performance of these methods are limited by their single tree structure. Ensemble methods significantly improve the prediction accuracy in survival tree modeling. Recent developments in this line including Hothorn et al. (2006) who utilize inverse probability of censoring weights (Van der Laan and Robins, 2003) to avoid censored observations, and Ishwaran et al. (2008) who employ random splitting rules and predict cumulative hazard functions. Our proposed recursively imputed survival trees (Zhu and Kosorok, 2012) updates censored observations to a model-based conditional failure time and refit the model. These extra failure observations help to build deeper trees and also modify the probability of using a strong variable as splitting rule. However, when the model structure is complicated and the dimension is large, the imputation may not always benefit. On the other hand, the proposed reinforcement learning trees survival model suits better in these scenarios.

Chapter 2

Recursively imputed survival trees

2.1 Introduction

My first dissertation topic is recursively imputed survival tree (RIST) regression for right-censored data. This new nonparametric regression procedure uses a novel imputation approach combined with extremely randomized trees that allows significantly better use of censored data then previous tree based methods, yielding improved model fit and reduced prediction error. The proposed method can also be viewed as a type of Monte Carlo EM algorithm which generates extra diversity in the tree-based fitting process. Simulation studies and data analyses demonstrate the superior performance of RIST compared to previous methods. This work is published in *Journal of the American Statistical Association*, 2012. The content of this part remains mostly unchanged from the published version, although some new findings and interpretations are added in the discussion section.

After reviewing some existing methods, an important question we could ask ourselves at this point is: what is the maximum information that can be extracted from censored survival data? We could also ask: is it possible to obtain as much information as is contained in non-censored survival data? And if not, what is the best we can do? These questions motivated us to develop an updating procedure that could extrapolate the information contained in a censored observation so that it could effectively be treated as uncensored. This basic idea is also motivated by the nature of tree model fitting which requires a minimum number of observed failure events in each terminal node. Consequently, censored data is in general hard to utilize, and information carried by censored observations is typically only used to calibrate the risk sets of the log-rank statistics during the splitting process. Motivated by this issue, we have endeavored to develop a method that incorporates the conditional failure times for censored observations into the model fitting procedure to improve accuracy of the model and reduce prediction error. The main difficulty in doing this is that calculation and generation of the conditional failure times requires knowledge of model structure. To address this problem, we propose an imputation procedure that recursively updates the censored observations to the current model-based conditional failure times and refits the model to the updated dataset. The process is repeated several times as needed to arrive at a final model. We refer to the resulting model predictions as recursively imputed survival trees (RIST).

Although imputation for censored data has been mentioned in the non-statistical literature (as, for example, in Hsieh (2007); and Tong et al. (2006), the proposed use of censored observations in RIST to improve tree-based survival prediction is novel. The primary benefits of RIST are three-fold. First, since the censored data is modified to become effectively observed failure time data, more terminal nodes can be produced and more complicated tree-based models can be built. Second, the recursive form can be viewed as a Monte Carlo EM algorithm Wei and Tanner (1990) which allows the model structure and imputed values to be informed by each other. Third, the randomness in the imputation process generates another level of diversity which contribute to the accuracy of the tree-based model. All of these attributes lead to a better model fit and reduced prediction error.

To evaluate the performance of RIST and compare with other popular survival methods, we utilize four forms of prediction error: Integrated absolute difference and supremum absolute difference of the survival functions, integrated Brier score (Graf et al., 1999; Hothorn et al., 2004) and the concordance index (used in Ishwaran et al. (2008)). The first two prediction errors for survival functions can be viewed as L_1 and L_∞ measures of the functional estimation bias. Note that the Cox model uses the hazard function as a link to the effect of covariates, so one can use the hazard function to compare two different subjects. Tree-based survival methods, in contrast, do not enjoy this benefit. To compare the survival of two different subjects and also calculate the concordance index error, we propose to use the area under the survival curve which can be handy in a study that runs for a limited time. Note that this would also be particularly useful for Q-learning applications when calculating the overall reward function based on average survival (Zhao et al., 2009).

The remainder of this part of the dissertation is organized as follows: In Section 2.2, we introduce the data set-up, notation, and model. In Section 2.3, we give the detailed proposed algorithm and some additional rationale behind it. Section 2.4 uses simulation studies to compare our proposed method with existing methods such as Random Survival Forests (Ishwaran et al., 2008), conditional inference Random Forest (Hothorn et al., 2006), and the Cox model with regularization (Friedman et al., 2010), and discusses pros and cons of our method. Section 2.5 applies our method to two cancer datasets and analyzes the performance. The paper ends with a discussion in Section 2.6 of related work, including conclusions and suggestions for future research directions.

2.2 Data set-up and model

The proposed recursively imputed survival tree (RIST) regression applies to right censored survival data. To facilitate exposition, we first introduce the data set-up and notation. Let $X = (X_1, ..., X_p)$ denote a set of p covariates from a feature space $\mathcal{X}$.

The failure time T given $X = x$ is generated from the distribution function $F_x(\cdot)$. For convenience, we denote the survival function as $S_x(\cdot) = 1 - F_x(\cdot)$. The censoring time C given $X = x$ has conditional distribution function $G_x(\cdot)$. The observed data are (Y, δ, X), where $Y = \min(T, C)$ and $\delta = I\{T \leq C\}$. Throughout this article we assume a conditionally independent censoring mechanism which posits that T and C are independent given covariates X. We also assume that there is a maximum length of follow-up time τ. A typical setting where this arises is under progressive type I censoring where survival is measured from study entry, and one observes the true survival times of those patients who fail by the time of analysis and censored times for those who do not. In this case, the censoring time C_i can be viewed as the maximum possible duration in the study for subject i, $i = 1, \ldots, n$. The survival time T_i for this subject follows survival distribution S_{x_i} which is fully determined by $\mathbf{X}_i = (X_{i1}, ..., X_{ip})$. If T_i is less than C_i, then $Y_i = T_i$ and $\delta_i = 1$ is observed; otherwise, $Y_i = C_i$ and $\delta_i = 0$ is observed. Using a random sample of size n, RIST can estimate the effects of covariate X on both the survival function and expectation of T (truncated at τ).

2.3 Proposed Method and Algorithm

2.3.1 Motivation and Algorithm outline

In this section we give a detailed description of our proposed recursively imputed survival tree (RIST) algorithm and demonstrate the unique and important features. One of the important ideas behind this method is an imputation procedure applied to censored observations that more fully utilizes all observations. This extra utilization helps improve the tree structures through a recursive form of model fitting, and it also enables better estimates of survival time and survival function.

The imputation procedure is motivated by a fact about censored data. Specifically, a censored observation will always fall into one of the following categories: The true survival time T is larger than study time τ so that we would not observe it even if the subject started at time 0 and was followed to the end of study; Alternatively, the true survival time T is less than τ so that we would observe the failure if the subject started at time 0 and there was no censoring prior to end of study. However, such a fact is masked whenever a subject is censored. Hence, the key questions are how to classify censored observations and how to impute values for them if they fall into either category.

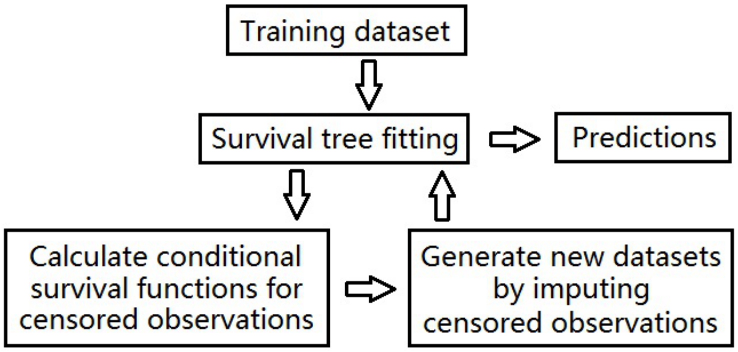

Training dataset
Survival tree fitting
Predictions
Calculate conditional survival functions for censored observations
Generate new datasets by imputing censored observations

Figure 2.1: A Graphical demonstration of RIST

We will begin our algorithm with a graphical view (Figure **??**) followed by a high-level illustration of the framework (Table 2.1), then a detailed description of each step will be given in subsequent sections: Survival tree model fitting (Section 2.3.2), Conditional survival distribution (Section 2.3.3), One-step imputation for censored observations (Section 2.3.4), Refit imputed dataset and further calculation (Section 2.3.5), and Final prediction (Section 2.3.6).

2.3.2 Survival tree model fitting

The extremely randomized tree (ERT) model is fitted to the initial training set to assess the model structure. The substantial differences between ERT and Breiman

Table 2.1: Algorithm for tree fitting

1. **Survival tree model fitting:** Generate M extremely randomized survival trees for the raw training data set under the following settings:
   a) For each split, K candidate covariates are randomly selected from p covariates, along with random split points. The best split, which provides the most distinct daughter nodes, is chosen.
   b) Any terminal node should have no less than $n_{min} > 0$ observed events.

2. **Conditional survival distribution:** A conditional survival distribution is calculated for each censored observation.

3. **One-step imputation for censored observations:** All censored data in the raw training data set will be replaced (with correctly estimated probability) by one of two types of observations: either an observed failure event with $Y < \tau$, or, a censored observation with $Y = \tau$.

4. **Refit imputed dataset and further calculation:** M independent imputed datasets are generated according to 3, and one survival tree is fitted for each of them using 1.a) and 1.b).

5. **Final prediction:** Step 2–4 are recursively repeated a specified number of times before final predictions are calculated.

(2001)'s Random Forests approach are that, first, the splitting value is chosen fully at random; second, the whole training set is used instead of only bootstrap replicates. M independent trees are fit to the entire training dataset as follows. For each tree, when reaching a node to split, K covariates along with one random split point per covariate are chosen from all non-constant covariates (splitting will stop if all covariates are constant). In our model fitting, the log-rank test statistic is used to determine the best split among the K covariates which provides the most distinct daughter nodes. Once a split has been selected, each terminal node is split again using the same procedure until no further splitting can be done without causing a terminal node to have fewer than n_{min} events (i.e. observations with $\delta = 1$). We will treat each terminal node as a group of homogeneous subjects for purposes of estimation and inference.

2.3.3 Conditional survival distribution

Calculations of conditional survival functions will be made first on the node level, then averaged over all M trees. For the l^{th} terminal node in the m^{th} tree, since there are at least n_{min} failure events, a Kaplan-Meier estimate of the survival function can be calculated within the node, which we denote by $\hat{S}^l_m(t)$, where $t \in [0, \tau]$. Noticing that for any particular subject, that subject eventually falls into only one terminal node for each fitted tree model, we can drop the index "l". Hence we denote the single-tree survival function by $\hat{S}^i_m$ for the i^{th} subject. Averaging over M trees, we have the forest level survival function $\hat{S}_i = \frac{1}{M}\sum_{m=1}^{M} \hat{S}^i_m$. Now, given a subject i that is censored at time c, i.e., $Y_i = c$ and $\delta_i = 0$, one can approximate the conditional probability of survival, $P(T_i > t | T_i > c)$, by

$$s^*_i = \begin{cases} 1 & \text{if } t \in [0, c] \\ \hat{S}_i(t)/\hat{S}_i(c) & \text{if } t \in (c, \tau] \end{cases} \tag{2.1}$$

Furthermore, we force $s^*_i(\tau+) = 0$ by imposing a point mass at time τ. This point mass will represent the probability that the conditional failure time is larger than τ.

2.3.4 One-step imputation for censored observations

When subject i is censored, the true survival time T_i is larger than C_i. However, if the subject is followed from the beginning of study (time 0), one and only one of the following two situations can happen: this subject could survive longer than the study length τ and we would not observe the failure time even if uncensored; or the subject could actually fail before the end of study. We now propose a one-step imputation procedure for these censored observations. The purpose of this one-step imputation is to unmask the above difference by utilizing the conditional survival function calculated in Section 2.3.3. To do so, we generate a new observation Y_i^* from this distribution function and treat it as the observed value if the subject were followed from time 0. Due to the construction of s_i^*, Y_i^* must be between Y_i and τ. If $Y_i^* < \tau$, then we assume that T_i is less than τ, and we replace Y_i by this new observation Y_i^* with censoring indicator $\delta_i^* = 1$. If $Y_i^* = \tau$, then we assume that the subject has T_i greater than τ, and we replace Y_i by τ with censoring indicator $\delta_i^* = 0$. This updating procedure is independently applied to all censored observations. This gives us a one-step imputed dataset. Note that the observed failure events in the dataset are not modified by this procedure.

2.3.5 Refit imputed dataset

Using the imputation procedure that we introduced in section 2.3.4, we independently generate M imputed datasets, and fit a single extremely randomized tree to each of them. We pool the M trees to assess the new model structure and survival function estimations. Subsequently, the new conditional censoring distribution can be calculated for each censored observation in the original dataset conditional on their corresponding original censoring value. The original censored observations can thus be again imputed. A new set of imputed datasets can be then generated to assess the next cycle model structure. Hence, a recursive form is established by repeating the model fitting procedure and imputation procedure. Note the term "original" here refers to the raw dataset before imputation. In other words the "conditional survival function" is always conditional on the original censoring time Y_i.

Interestingly, at this stage, all observations are either observed failure events or effectively censored at τ. The traditional Kaplan-Meier estimator will reduce to a simple empirical distribution function estimator. Details of this empirical distribution function estimator will be given in the following section.

This recursive approach can be repeated multiple times prior to the final step. Each time, the imputation is obtained by applying the current conditional survival function estimate to the original censored observations. We denote the process involving q imputations as q-fold RIST, or simply RIST q.

2.3.6 Final prediction

The final prediction can be obtained by calculating node level estimation and then averaging over all trees in the final model fitting step. For a given new subject with covariates $X^{new} = (X_1^{new}, ..., X_p^{new})$, denote $S^{new}(\cdot)$ to be the true survival function for this subject. Dropping this subject down the m^{th} tree, it eventually falls into a terminal node (which we label as node l). Note that all the observations in this node are either observed events before τ, or censored at τ, and we will treat all observations in a terminal node as i.i.d. samples from the same distribution. To estimate $S^{new}(t)$, we employ an empirical type estimator which can be expressed, in the m^{th} tree, as $\hat{S}_m^{new}(t) = \sum_{i \in \text{ node l}} \frac{I\{Y_i > t\}}{\varphi_m(l)}$ where $\varphi_m(l)$ denotes the size of node l in the m^{th} tree. Then the final prediction can be calculated as follows:

$$\text{and} \quad \hat{S}^{new}(t) \;=\; \frac{1}{M}\sum_{m=1}^{M} \hat{S}_m^{new}(t). \tag{2.2}$$

2.4 Simulation Studies

In this section, we use simulation studies to compare the prediction accuracy of RIST with three existing methods, including two popular tree-based models and the Cox model with regularization. Random Survival Forests (Ishwaran et al., 2008) and conditional inference Random Forest (Hothorn et al., 2006) are both constructed based on Breiman (2001)'s Random Forests algorithm. The Random Survival Forests (RSF) constructs an ensemble of cumulative hazard functions. The conditional inference Random Forest

(RF) approach utilizes inverse probability of censoring (IPC) weights (Van der Laan and Robins, 2003) and analyzes right censored survival data using log-transformed survival time. The above two methods are implemented through R-packages “randomSurvivalForest” and ”party”. It also interesting to compare our method to the Cox model with regularization. Although the Cox model has significant advantages over tree-based models when the proportional hazards model is the true data generator, it is still important to see the relative performance of tree-based models under such circumstances. The Cox model fittings are implemented through the R-package “glmnet” (Friedman et al., 2010; Simon et al., 2011).

2.4.1 Simulation settings

To fully demonstrate the performance of RIST, we construct the following five scenarios to cover a variety of aspects that usually arise in survival analysis. The first scenario is an example of the proportional hazards model where the Cox model is expected to perform best. The second and third scenarios represent mild and severe violations of the proportional hazards assumption. The censoring mechanism is another important feature that we want to investigate. In Scenario 4, both survival times and censoring times depend on covariate X, however, they are conditionally independent. Scenario 5 is an example of dependent censoring where censoring time not only depends on X but is also a function of survival time T. Although this is a violation of our assumption, we want to demonstrate the robustness of RIST. Now we describe each of our simulation settings in detail:

Scenario 1: A proportional hazards model adapted from Section 4 of Ishwaran et al. (2008), we let $p = 25$ and $X = (X_1, ..., X_{25})$ be drawn from a multivariate normal distribution with covariance matrix V, where $V_{ij} = \rho^{|i-j|}$ and ρ is set to 0.9. Survival times are drawn independently from an exponential distribution with mean

$\mu = b_0 \times \sum_{i=11}^{20} x_i$, where b_0 is set to 0.1. Censoring times are drawn independently from an exponential distribution with mean set to half of the average of μ. Study length τ is set to 4. Sample size is 200 and the censoring rate is approximately 30%.

Scenario 2: We draw 10 i.i.d. uniform distributed covariates and use link function $\mu = \sin(x_1 \times \pi) + 2 \times |x_2 - 0.5| + x_3^3$ to create a violation of the proportional hazards assumption. Survival times follow an exponential distribution with mean μ. Censoring times are drawn uniformly from $(0, \tau)$ where $\tau = 6$. Sample size is 200 and the censoring rate is approximately 24%.

Scenario 3: Let $p = 25$ and $X = (X_1, ..., X_{25})$ be drawn from a multivariate normal distribution with covariance matrix V, where $V_{ij} = \rho^{|i-j|}$ and ρ is set to 0.75. Survival times are drawn independently from a gamma distribution with shape parameter $\mu = 0.5 + 0.3 \times |\sum_{i=11}^{15} x_i|$ and scale parameter 2. Censoring times are drawn uniformly from $(0, 1.5 \times \tau)$ and the study length τ is set to 10. Sample size is 300 and the censoring rate is approximately 20%.

Scenario 4: We generate a conditionally independent censoring setting where $p = 25$ and $X = (X_1, ..., X_{25})$ are drawn from a multivariate normal distribution with covariance matrix V, where $V_{ij} = \rho^{|i-j|}$ and ρ is set to 0.75. Survival times are drawn independently from a log-normal distribution with mean set to $\mu = 0.1 \times |\sum_{i=1}^{5} x_i| + 0.1 \times |\sum_{i=21}^{25} x_i|$. Censoring times follow the same distribution with parameter $\mu + 0.5$. Study length τ is set to 4. Sample size is 300 and the censoring rate is approximately 32%.

Scenario 5: This is a dependent censoring example. We let $p = 10$ and $X = (X_1, ..., X_{10})$ be drawn from a multivariate normal distribution with covariance matrix V, where $V_{ij} = \rho^{|i-j|}$ and ρ is set to 0.2. Survival times T are drawn independently from an exponential distribution with mean $\mu = \frac{e^{x1+x2+x3}}{(1+e^{x_1+x_2+x_3})}$. A subject will be censored at one third of the survival time with probability $\mu/2$. The study length $\tau = 2$, sample size is 300 and the censoring rate is approximately 27%.

2.4.2 Tuning parameter settings

All three tree-based methods offer a variety of tuning parameter selections. To make our comparisons fair, we will equalize the common tuning parameters shared by all methods and set the other parameters to the default. According to Geurts et al. (2006); Ishwaran et al. (2008) the number of covariates considered at each splitting, K, is set to the integer part of $\sqrt{p}$ where p is the number of covariates. For RIST and RSF, the minimal number of observed failures in each terminal node, n_{min}, is set to 6. The counterpart of this quantity in the RF, minimal weight for terminal nodes is set to the default. For RSF and RF, 1000 trees were grown. Two different splitting rules are considered for RSF: the log-rank splitting rule and the random log-rank splitting rule (see Section 6 in Ishwaran et al. (2008)). In the RF, a Kaplan-Meier estimate of the censoring distribution is used to assign weights to the observed events. The imputation process in RIST can be done multiple times before reaching a final model. Here we consider 1, 3, and 5 imputation cycles with $M = 50$ trees in each cycle (namely 1-fold, 3-fold, and 5-fold RIST).

The Cox models are fit with penalty term $\lambda P_\alpha(\beta) = \lambda[(1-\alpha)/2||\beta||_2^2 + \alpha||\beta||]$. We use the lasso penalty by setting $\alpha = 1$. The best choice for λ is selected using the default 10-fold cross-validation.

2.4.3 Prediction Error

The survival function is the major estimation target in all tree-based methods and can be easily calculated for the Cox model. We first define 3 prediction errors for survival function estimations as follows: Integrated absolute error and supremum absolute error can be viewed as L_1 and L_∞ measures of the survival function estimation error.

To be more specific, let $S(t)$ denote the true survival function and let $\hat{S}(t)$ denote its estimate. Integrated absolute error is defined as $\int_0^\tau |S(t) - \hat{S}(t)|dt$ and Supremum absolute error is defined as $\sup_{0 \leq t \leq \tau} |S(t) - \hat{S}(t)|$. Noticing that both measurements require knowledge of the true data generator, which is typically not known in practice, we also utilize the widely adopted integrated Brier score (Graf et al., 1999; Hothorn et al., 2006) as a measure of performance since it can be calculated from observed data only. The Brier score for censored data at a given time $t > 0$ is defined as

$$\begin{aligned} BS(t) = \frac{1}{N}\sum_{i=1}^{N} \quad & \{ \quad (\hat{S}(t|X_i))^2 I(Y_i \leq t \wedge \delta_i = 1)\hat{G}(Y_i)^{-1} \\ & +(1 - \hat{S}(t|X_i))^2 I(Y_i > t)\hat{G}(Y_i)^{-1} \quad \}, \end{aligned} \tag{2.3}$$

where $\hat{G}(\cdot)$ denotes the Kaplan-Meier estimate of the censoring distribution. The integrated Brier score is further given by

$$IBS = max(Y_i)^{-1} \int_0^{max(Y_i)} BS(t)dt. \tag{2.4}$$

In the simulation study validation set, where the failure times are fully observed, $\hat{G}(t)$ reduces to 1 and $\delta = 1$. The integrated Brier score can then be viewed as a degenerate version of an L_2 measure of the survival function estimation error. In our simulations, the Brier score is only calculated up to the maximum study length τ since there is no information available beyond τ in the training dataset. Hence the integrated Brier score in our simulation study is defined by $IBS = \tau^{-1} \int_0^\tau BS(t)dt$. Note that this definition will also prevent errors at large t from dominating the results.

The fourth prediction error that we utilize is Harrell's concordance index (C-index) (Harrell Jr et al., 1982; Ishwaran et al., 2008) which can also be used with observed data only. The C-index provides a nonparametric estimate of the correlation between the estimated and true observed values based on the survival risks of a pair of randomly selected subjects. To compare the risks of two subjects, RIST uses area under the predicted survival curve; RSF uses cumulative survival function; the RF uses predicted survival time; and the Cox model uses the link function. A detailed calculation of the C-index algorithm is given in Ishwaran et al. (2008), and the prediction error is defined as 1 minus the C-index.

2.4.4 Simulation results

Each simulation setting is replicated 500 times and results are presented in the following tables. For convenience, within each scenario, we use the best method in terms of performance as the reference group which we rescale to 1. Prediction errors for all other methods are scaled and presented as a ratio to the reference group, i.e. prediction errors larger than 1 will indicate a worse performance. The last column is the original scale multiplier. Major findings are summarized below:

1. In all simulation settings with survival function prediction error, RIST performs better than the other two tree-based methods and the improvements are significant. For example, under the proportional hazards model (Scenario 1) with integrated absolute error of the survival function (Table 2), RSF0 and RF perform 37.7% and 68.9% worse than RIST respectively. In all other scenarios, RIST performs at least 19% better than RSF and the improvements can be up to 31.4% better in terms of this error measurement. For supremum error, RIST performs $21.6 \sim 55.5\%$ better than RF and improvements over RSF generally lie between $10 \sim 20\%$. Improvements in terms of integrated Brier score are less impressive due to the large variability when generating the survival times, however, performances of RIST are uniformly better than RSF and RF.

2. Results for comparing RIST with the Cox model can vary from situation to situation. In Scenarios 3 and 4 where the proportional hazards assumption is severely

violated, performance of the Cox model can be over 40% worse than RIST in terms of both integrated and supremum survival error. On the other hand, under the proportional hazards model, the Cox model performs 26.4% better than RIST. However, when compared to RSF0 and RF (which 74.1% and 113.5% worse than the Cox model), RIST still shows a much stronger performance relative to the other tree-based methods.

3. If we focus on the worst performing scenario for each method, we can see that the robustness of RIST is superior to any competing methods. In fact, RIST is the most robust in terms of all three survival function estimation errors. And RIST never falls into the “worst two” category in any situation using any error measurement, whereas all other methods always, at some point, fail to compete with the others (i.e., has largest prediction error).

4. 3-fold and 5-fold RIST generally perform better than 1-fold RIST, however, higher-fold imputation does not always further improve the performance. The reason is that after several cycles of imputation, the model structure tends to have stabilized. This might also possibly be due to overfitting in certain settings. Scenario 5 represents a dependent censoring case which violates our model assumptions, and slight overfitting can be seen. This phenomenon indicates that our imputation procedure is somewhat sensitive to the information carried by censored observations, but not excessively so. Nevertheless, severe violation of the independent censoring assumption could further downgrade the performance of RIST.

5. For many simulation settings, the C-index errors are very close for all the methods. Simulations show that the C-index is not as sensitive as other measurements. For example, in Scenario 1 (the proportional hazards model) where the Cox model is

clearly superior to any tree-based models, RSF1 still shows an even lower C-index error than the Cox model. Hence interpretability of the C-index is sometimes unclear.

6. Performance of the RF method is generally not as strong as the other approaches. The likely reason is that this method utilizes inverse probability of censoring (IPC) which relies heavily on the assumption that $G(T|X) = P(C > T|X)$ is strictly greater than zero almost everywhere. However, in real life study designs, such as in clinical trials running for a predefined period, this assumption is violated (Hothorn et al., 2006). Under such circumstances, the estimation of mean survival time would be expected to be biased.

Table 2.2: Integrated absolute error for survival function [†]

				Prediction error based on 500 simulations				
Settings	Cox	RSF0	RSF1	RF	RIST1	RIST3	RIST5	Original Scale
1	1	1.741	1.753	2.135	1.281	1.268	1.264	0.172
2	1.047	1.253	1.217	1.153	1.022	1.009	1	0.378
3	1.464	1.190	1.314	1.358	1.006	1.000	1	0.791
4	1.201	1.195	1.281	1.270	1.016	1.005	1	0.320
5	1.081	1.316	1.243	1.213	1	1.006	1.008	0.118

Table 2.3: Supremum absolute error for survival function [‡]

				Prediction error based on 500 simulations				
Settings	Cox	RSF0	RSF1	RF	RIST1	RIST3	RIST5	Original Scale
1	1	1.364	1.361	1.788	1.151	1.150	1.151	0.073
2	1.075	1.120	1.014	1.216	1.002	1.003	1	0.112
3	1.438	1.113	1.157	1.375	1.001	1	1.001	0.139
4	1.250	1.134	1.103	1.340	1.002	1.000	1	0.142
5	1	1.323	1.238	1.399	1.198	1.204	1.206	0.082

[†]: Integrated absolute error for survival function is defined as $\int_0^\tau |S(t) - \hat{S}(t)|dt$.

[‡]: Supremum absolute error for survival function is defined as $\sup\limits_{0\leq t\leq\tau} |S(t) - \hat{S}(t)|$.

RSF0 and RSF1 are Random Survival Forests using logrank and random logrank splitting rules respectively. RIST1, RIST3, and RIST5 are 1-fold, 3-fold, and 5-fold RIST respectively.

As suggested by one of the reviewers, in addition to presenting mean prediction errors, we also want to further analyze where the differences occur in time over the

Table 2.4: C-index error

Prediction error based on 500 simulations

Settings	Cox	RSF0	RSF1	RF	RIST1	RIST3	RIST5	Original Scale
1	1.002	1.015	1	1.030	1.008	1.007	1.007	0.305
2	1.079	1	1.007	1.056	1.017	1.017	1.018	0.439
3	1.405	1.010	1.006	1.124	1.001	1.000	1	0.356
4	1.273	1	1.000	1.041	1.006	1.004	1.005	0.393
5	1	1.059	1.027	1.074	1.037	1.036	1.037	0.291

Table 2.5: Integrated Brier score

Prediction error based on 500 simulations

Settings	Cox	RSF0	RSF1	RF	RIST1	RIST3	RIST5	Original Scale
1	1	1.038	1.037	1.135	1.018	1.017	1.017	0.125
2	1.009	1.008	1.007	1.020	1.000	1.000	1	0.130
3	1.101	1.022	1.041	1.094	1.000	1.000	1	0.128
4	1.044	1.018	1.032	1.063	1.001	1.000	1	0.124
5	1.059	1.021	1.014	1.028	1.000	1	1.001	0.116

RSF0 and RSF1 are Random Survival Forests using logrank and random logrank splitting rules respectively. RIST1, RIST3, and RIST5 are 1-fold, 3-fold, and 5-fold RIST respectively.

study duration. Hence we plot the mean survival errors over time for two somewhat typical settings: Scenario 1, the proportional hazards model; and Scenario 3, in which the proportional hazards assumption is violated. The mean survival error over time is calculated by averaging $|S(t) - \hat{S}(t)|$ over all subjects in the validation set, and the plot is the average over 500 simulation runs. As presented in Figure 2 (Scenario 1), the Cox model performs uniformly best. Comparison among tree-based methods show that RIST5 remains relative strong in performance under the proportional hazards model. In Figure 3 (Scenario 3), RIST has a significant improvement over all other competing methods, and the improvements occur over the entire range of t. Due to violation of the proportional hazards assumption, the Cox model has the worst performance in this setting. One interesting fact that we observed is that, in many circumstances, RSF estimations of the survival functions seem be unstable towards the end of study duration and the prediction error is increased while all other methods tend to have their prediction errors decreasing towards the end.

Figure 2.2: Proportional Hazards Model

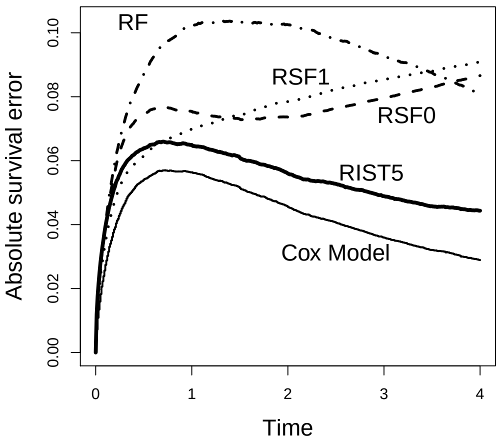
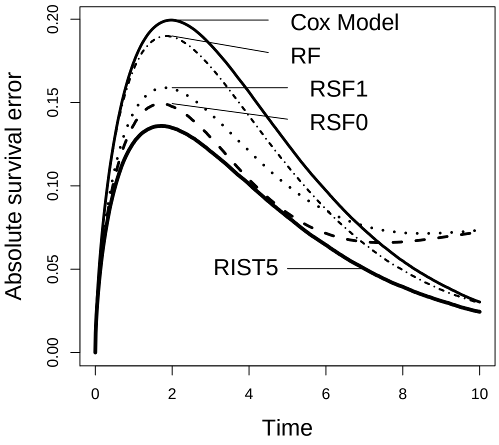


Figure 2.3: Non-Proportional Hazards



2.5 Data Analysis

In this section we compare RIST with RSF, RF, and the Cox model on two datasets: the German Breast Cancer Study Group (GBSG) data and the Primary Biliary Cirrhosis (PBC) data. We use Brier score and integrated Brier score as the criteria for comparison. The integrated Brier score, as we observed in the simulation studies, provides a slightly more sensitive measurement than the C-index. A random assignment algorithm (a slight modification from Ishwaran et al. (2008)) is also being introduced to handle missing covariate data in the PBC data section.

2.5.1 Breast Cancer Data

In 1984, the German Breast Cancer Study Group (GBSG) started a multi-center randomized clinical trial to compare recurrence-free and overall survival between different treatment modalities (Schumacher et al., 1994). In this section we utilize this dataset to compare RIST with other methods.

2.5.1.1 Data description

By March 31, 1992, median follow-up time was 56 months with 197 events for disease-free survival and 116 deaths observed. The recurrence-free survival times of the 686 patients (with 299 events) who had complete data were analyzed in Sauerbrei and Royston (1999). The $p = 8$ observed factors are age, tumor size, tumor grade, number of positive lymph nodes, menopausal status, progesterone receptor, estrogen receptor, and whether or not hormonal therapy was administered. There is no missing data. This data-set has been studied by both Ishwaran et al. (2008) and Hothorn et al. (2006) for tree types of model fitting, hence we also utilize this dataset in our paper.

We randomly divide the dataset into two equal sized subsets, and then use one as a training set and the other as a validation set. 500 independent training datasets were thus generated and prediction error calculated according to the corresponding validation sets. All parameter settings are identical to those given in Section2.4.2.

2.5.1.2 Results

We present the relative over-time Brier scores in Figure 4 (using 5-fold RIST as the reference group, and subtracted from each method accordingly). The plot is constructed so that worse performance compared to 5-fold RIST is above 0. The Brier score for RF is significantly distinct from other methods and its relative Brier score is over 0.15 more than RIST towards the end of study. Among all other methods, RIST and RSF0 performs similar, while RIST has lower Brier score at a majority of time points across the entire range. The Cox model and RSF1 perform worse than the above two; however, they both perform significantly better than RF.

The boxplot for integrated Brier scores are shown in Figure 5. The boxplot for RF is above the upper bound (with mean 0.2535 and 1^{st}, 2^{nd}, and 3^{rd} quartiles 0.2438, 0.2529, and 0.2623 respectively) and will not be presented in this plot. RIST performs best in terms of both mean and median integrated Brier score. The improvement compared to the Cox model, RSF1 and RF, is significant. RSF0 performs close to RIST, however RIST5 has lower integrated Brier score than RF0 in 62.2% of the simulations, and outperforms the Cox model and RSF1 in 78.8% and 93.8% of the simulations respectively.

A variable importance (Breiman, 2001; Ishwaran, 2007) analysis is done by using the validation set to assess the variable importance measure. However, similar results were found among all tree-based methods.

Figure 2.4: Relative Brier score

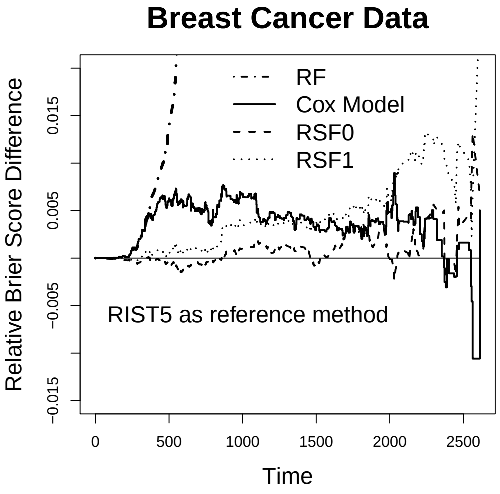
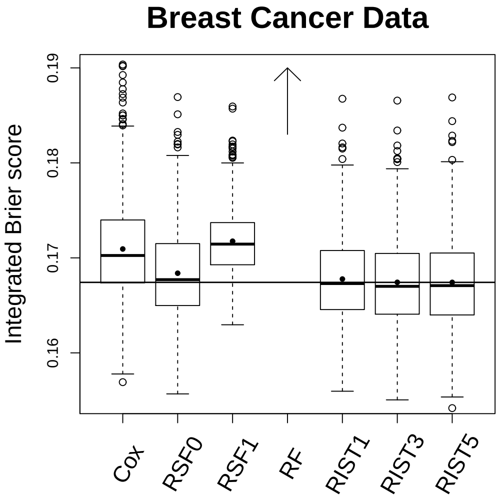


Figure 2.5: Integrated Brier score



2.5.2 PBC Data

The Mayo clinical trial of primary biliary cirrhosis (PBC) of the liver (Fleming and Harrington, 2011) has long been famous and considered a benchmark dataset in survival analysis. We compare the performance of RIST with other methods on this dataset. A method for handling missing covariates is also introduced.

2.5.2.1 Data description

This Mayo clinical trial study was conducted between 1974 and 1984 and the study analysis time was in July, 1986. A total of 424 PBC patients, referred to the Mayo clinic during that ten-year interval, met the eligibility criteria for the randomized trial. 312 cases in the dataset participated in the randomized trial and contain largely complete data and hence will be used in our analysis. The additional 112 cases did not participate in the clinical trial and these data will not be used. The data contains 17 covariates including treatment, age, sex, ascites, hepatomegaly, spiders, edema, bilirubin, cholesterol, albumin, urine copper, alkaline phosphatase, SGOT, triglicerides, platelets, prothrombin time, and histologic stage of disease.

As with the breast cancer example, we randomly divide the PBC data set into a training dataset and a validation set with equal sample size and independently repeat this 500 times. Model parameter settings here are also the same as in the breast cancer example.

2.5.2.2 Missing covariate method

Missing data is an issue in the PBC dataset. Among the 312 subjects, there are 28 subjects with missing cholesterol measurements, 30 with missing triglicerides measurements, 2 with missing urine copper measurements and 4 with missing platelet measurements. There are 276 subjects with complete measurements for all covariates. Our algorithm for handling missing data is very similar to Ishwaran et al. (2008), where the missing X values are randomly generated from the empirical distribution of the in-bag observations in a node. Ishwaran et al. (2008)'s method will be implemented in both RSF0 and RSF1.

Now We describe our missing data algorithm as follows: To find the best splitting variable from the K randomly chosen covariates, the test statistic for any variable X_p

is calculated by omitting the subjects that have missing X_p value. When the splitting variable is chosen and daughter nodes are built, those subjects with missing splitting variable are randomly assigned to either daughter node with probabilities proportional to the sizes of the daughter nodes. This random assignment algorithm is also applied during the prediction process. Suppose we drop a subject with missing covariate X_p down a single tree. Whenever X_p is required to determine which further node it falls into, we randomly throw this subject into either node with probability proportional to node size as described above.

2.5.2.3 Results

Similar to the Breast Cancer data analysis, we present the relative over-time Brier scores in Figure 6 using 5-fold RIST as the reference group. The Brier score of RF increases dramatically as time increases. We restrict our plotting frame so that we can focus more on the differences between the other methods. The Brier score of the Cox model and RSF1 is higher than RIST5 at almost every time over the entire study duration. RSF0 has higher prediction error than RIST5 at most time points, however, it out-performs RIST5 towards the end of study.

The boxplot for integrated Brier scores are shown in Figure 7. We again restrict the plotting frame so that for the majority of time RF will be above the upper bound and differences between other methods can be easily seen. RIST5 performs best, followed by RIST3, RIST1, RSF0, the Cox model and RSF1. A t-test comparing RSF0 and RIST5 shows that RIST5 is significantly better with P-value < 0.001. In fact, in 65.2% simulations, RIST5 has lower integrated Brier score than RSF0. Moreover, RIST5 out-performs the Cox model and RSF1 in 87.8% and 99.6% simulation runs respectively.

Figure 2.6: Relative Brier score

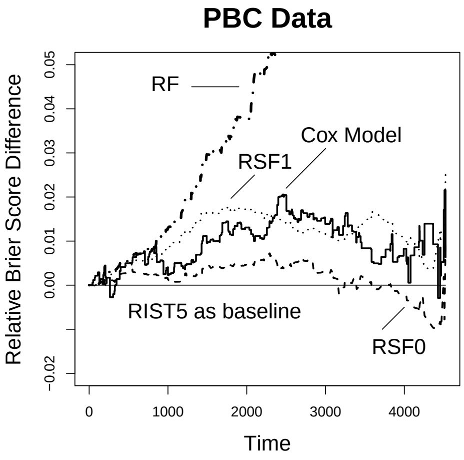
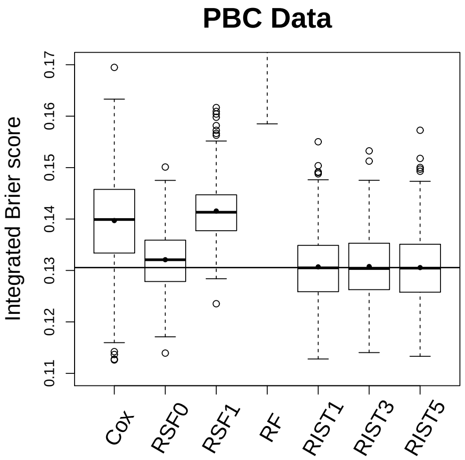


Figure 2.7: Integrated Brier score



2.6 Discussion

In this paper, we introduced recursively imputed survival trees (RIST), a novel censoring imputation approach integrated with a tree-based regression method for right-censored survival data. While preserving information carried by the censored observations (by calculating conditional survival distribution), the imputation method extends the utility of censored observations and uses the updated conditional failure information to improve model prediction. The regression procedure is built on the newly developed tree method, extremely randomized trees (Geurts et al., 2006), which is an alternative to Breiman's popular random forests method. Through a recursive algorithm, both the model fitting processes and the imputation processes affect each other, and the performances of both improve simultaneously.

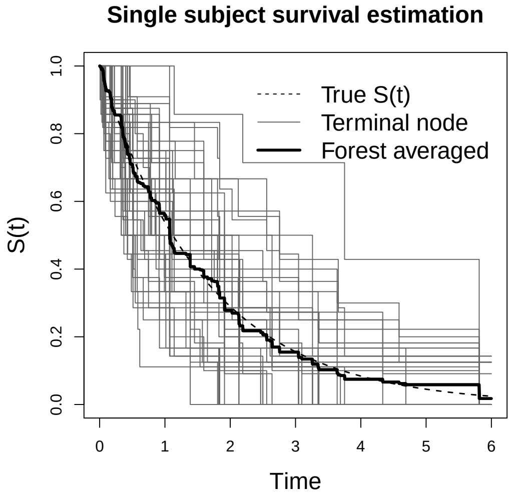
2.6.1 Why RIST works

Up to this point, we have only used simulations to demonstrate the performance of RIST. It is important and interesting to discuss the motivation and driving force behind our proposed method. Here we provide several explanations that will help further understand this new approach.

One potential advantages of RIST comes from the tree-based modeling point of view. Since the entire training set is used to build each single tree, extremely randomized trees can build larger models (i.e with more terminal nodes) compared to Random Forests which use bootstrap samples. Furthermore, after the first imputation cycle, additional observed events are created which allow each tree to grow even deeper. One may wonder whether this could cause over-fitting; however, the random generation of the imputed values provides sufficient diversity which will help eliminate over-fitting.

Moreover, we found that the Monte Carlo EM (MCEM) algorithm (Wei and Tanner, 1990; McLachlan and Krishnan, 2007) is the best way to explain our proposed procedure theoretically. The random generation of imputed values can be viewed as the Monte Carlo E-step without taking the average of all randomly generated sample points, while the survival tree fitting procedure is explicitly an M-step to maximize the nonparametric model structure. The "random E-step" imputation procedure does not only preserve the information carried by censored observations, but it also introduces an extra level of diversity into the next-level of model fitting. As is well known, diversity is one of the driving forces behind the success of ensemble methods as has been addressed by many researchers, including Breiman (2001); Dietterich (2000). An interesting phenomenon of diversity can be seen when averaging the terminal node survival function estimation over the forest. Figure 8 (of a subject from Scenario 2) shows that even though an individual terminal node estimation (using n_{min} observed events) could have a high variance or be largely biased, the overall forest estimation will still be very accurate. In the most common ensemble tree methods, diversity can be created through taking bootstrap samples and random selections of variables and their splitting values. With independently imputed datasets, the patterns being recognized by each tree in a forest will present an even greater level of diversity. The accuracy of survival function and conditional survival function estimations can therefore be even further improved.

Figure 2.8: Diversity and forest averaging



The effect that we have seen over the imputation cycle can also be visually explained as a "blurring effect" in optics: While each model fitting step sums up all information from adjacent observations of the target point in the feature space, similar effects also happen to other adjacent observations simultaneously. The next imputation step allows information from remote observations to be carried into adjacent censored observations which can be used in calculating the target survival function estimation. Hence, over several imputation cycles, the overall information that defines the target prediction does not come solely from the partitioned neighborhood of the target point, it comes instead from a "blurred" neighborhood that reaches out to a much wilder range.

Another reason that we realized later on is that by imputing the censored observations, the chance of using a signaled variable as splitting rule in the refitted model is slightly increased, since the imputation is based on the signals that are found in the initial model fitting. When the initial model is reliable, the imputation enhances the influence of the signaled variables. However, this might run into trouble when the initial model is not detecting the true signal correctly, such as in a high-dimensional setting or when the model structure is too complicated. My third topic, reinforcement learning trees for survival data, is proposed to solve this problem.

2.6.2 Other Issues

In multi-fold RIST, most of the improvement is gained during the first several imputation cycles. Additional recursive steps of RIST can help adjust the imputed value and the fitted model structure; however, the increments of improvement tend to be small since the model structure stabilizes fairly quickly. Unfortunately, we do not yet have explicit convergence criteria for RIST. However, based on our simulation experience, it appears that 3-fold to 5-fold RIST generally performs best. Although higher fold imputations perform reasonably well and may even be optimal in some settings, over-fitting also appears to be a possibility. In addition, as fold level increasing, the computational intensity also increases. Hence, we do not recommend going beyond 5-fold RIST.

Another issue that has been addressed frequently in tree-based model fitting is the choice of splitting statistics. During our research, we examined the performance of several alternatives to the log-rank statistic, including the supremum log-rank (Kosorok and Lin, 1999) statistics. However, no significant differences in performance of RIST were detected under the simulation settings that we presented.

Although it is not the focus of our paper, the missing data issue often occurs. Our missing data algorithm is very similar to the approach given in Ishwaran et al. (2008). However, the way we handle missing subjects can ensure that there are a sufficient number of non-missing subjects in each node. This is because we randomly categorize the missing subjects into daughter nodes after the splitting has been done. For our current method, we suggest removing any subject with missing Y value or missing censoring indicator. Although these data can be easily handled with the same logic based on missing covariate classification, we feel that our censoring imputation method relies somewhat on the accuracy of outcome variables, so that imputing subjects with incomplete outcomes may eventually increase prediction error.

Chapter 3

Reinforcement learning trees

3.1 Introduction

My second dissertation topic is a new type of tree-based regression method, reinforcement learning trees (RLT), which exhibits significantly improved performance over traditional methods such as random forests (Breiman, 2001). The innovations are three-fold. First, the new method implements reinforcement learning at each selection of a splitting variable during the tree construction processes. By splitting on the variable that brings the greatest future improvement in later splits, rather than choosing the one with largest marginal effect from the immediate split, the constructed tree utilizes the available samples in a more efficient way. Moreover, such an approach can be adapted to make high-dimensional cuts available at a relatively small computational cost. Second, we propose a variable screening method that progressively mutes noise variables during the construction of each individual tree. The muting procedure also takes advantage of reinforcement learning and prevents noise variables from being considered in the search for splitting rules, so that towards a terminal node when the sample size is small, the splitting rules are still constructed from only strong variables. Last, we investigate asymptotic properties of the proposed method. We can show that under the proposed splitting variable selection procedure, the constructed trees are consistent. The error bounds for the proposed RLT are shown to depend on a pre-selected number p_0, where p_0 is an educated guess of the number of strong variables which is usually much smaller than the total number of variables p but at least as large as the true number of strong variables p_1. Hence when p_0 is properly chosen, the error bounds can be significantly improved.

We introduce a new philosophy—reinforcement learning—into the tree-based model framework. For a comprehensive review of reinforcement learning within the artificial intelligence field in computer science and statistical learning, we refer to Sutton and Barto (1998). An important characteristic of reinforcement learning is the "peek-at-the-future" notion which benefits the long-term performance rather than short-term performance. The main features we will employ in the proposed method are: first, to choose variable(s) for each split which will bring the largest return from future branching splits rather than only focusing on the immediate consequences of the split. Such a splitting mechanism can break any hidden structure and avoid inconsistency by forcing splits on strong variables even if they do not show any marginal effect; second, progressively muting noise variables as we go deeper down a tree so that even as the sample size decreases rapidly towards a terminal node, the strong variable(s) can still be properly identified from the reduced space. One consequence of the new approach, which we call reinforcement learning trees (RLT), as we will show later, is that the convergence rate should not depend on p, but instead, it depends on a pre-specified value p_0 which is much smaller than p and larger than p_1. Hence, when p_0 is properly chosen, the convergence rate can be greatly improved.

Another extension we bring with the proposed RLT is a high-dimensional cut which uses a linear combination of variables to create a splitting rule. In traditional tree-based methods, searching for a high-dimensional cut will dramatically increase the computational intensity. However, with the pre-identification of important variables, the cutting surface can be reasonably formed without exhaustive searching. In the simulation studies and data analyses presented later, we will examine the performance of the newly proposed RLT with both one-dimensional and high-dimensional cuts and show that the benefit can be profound in some situations.

The part of the dissertation is organized as follows. In Section 3.2, we introduce the underlying model and notation to facilitate the formulation of our method. In Section 3.3, we give details of the methodology for the proposed approach. Theoretical results and their interpretation are given in Section 3.4. Most of the details of the proofs will be deferred to the last section. In Sections 3.5 we compare RLT with popular statistical learning tools, such as random forests (Breiman, 2001), BART (Chipman et al., 2010), gradient boosting (Friedman, 2001) and GLM with LASSO (Friedman et al., 2010), using simulation studies and real datasets. Section 3.6 contains some discussion and gives rationale for both the method and asymptotic behaviors. Future research directions are also discussed. The paper concludes with the proofs.

3.2 Statistical model

We consider a regression or classification problem from which we observe a sample of i.i.d. training observations $\mathcal{D}_n = \{(\mathbf{X}_1, Y_1), (\mathbf{X}_2, Y_2), ..., (\mathbf{X}_n, Y_n)\}$, where each $\mathbf{X}_i = (X_i^{(1)}, ..., X_i^{(p)})^T$ denotes a set of p variables from a feature space $\mathcal{X}$. For the regression problem, Y is a real valued outcome with $E(Y^2) < \infty$; and for the classification problem, Y is binary outcome that takes values of 0 or 1. We also assume that the expected value $E(Y|\mathbf{X})$ is completely determined by a set of $p_1 < p$ variables. We refer to these p_1 variable as "strong variables", and refer to the remaining $p_2 = p - p_1$ variables as "noise variables". Without loss of generality, we assume that the strong variables are the first p_1 variables, which means $E(Y|\mathbf{X}) = E(Y|X^{(1)}, X^{(2)}, ..., X^{(p_1)})$. The goal is to consistently estimate the function $f(x) = E(Y|\mathbf{X} = x)$ and derive asymptotic properties for the estimator. To facilitate later arguments, we use $\mathcal{P}$ to denote the set $\{1, 2, ..., p\}$.

3.3 Reinforcement learning trees

In short, the proposed reinforcement learning trees are traditional random forests with a special type of splitting variable selection and noise variable muting at each internal node. These features are made available by implementing a reinforcement learning mechanism. Let us first consider an example which demonstrates the impact of reinforcement learning: Assume that $E(Y|\mathbf{X}) = I(X^{(1)} > 0.5)I(X^{(2)} > 0.5)$, so that $p_1 = 2$ and $p_2 = p - 2$. The difficulty in estimating this structure with conventional random forests is that neither of the two strong variables show marginal effects. The immediate reward, i.e. reduction in prediction errors, from splitting on these two variables is identical to the reward obtained by splitting on one of the noise variables. Hence, it unlikely that, when p is relatively large, either $X^{(1)}$ or $X^{(2)}$ would be chosen as the splitting variable. However, if we know in advance that splitting on either $X^{(1)}$ or $X^{(2)}$ would yield significant rewards down the road for later splits, we could confidently force a split on either variable regardless of the immediate rewards.

How we identify the most important variable at any internal node is to first fit at that node an embedded random forest and acquire the associated variable importance measures for all the covariates. Then we proceed to split the node using the most important variable(s). When doing this recursively for each daughter node, we can focus the splits on the variables which will be very likely to lead to a tree yielding the smallest prediction error in the long run.

Unfortunately, since the sample size shrinks as we move towards a terminal node, it becomes increasingly difficult to identify the important variables regardless of what embedded model we are using. On the other hand, since we have variable importance information in all the splits from the root node down to a terminal node, we should have a good idea about which variables are strong and which are not. Therefore, we will utilize this information to progressively mute noise variables during the tree construction process and to gradually restrict the search for splitting variables within a subspace of the entire feature space as the internal node sample sizes get smaller.

The remainder of this section is structured as follows: We first give a higher level algorithm outlining the main features of the RLT method in Section 3.3.1 without specifying the definitions of the subcomponents: embedded model, variable importance, variable deletion, and high-dimensional split. Detailed definitions of these components are given in subsequent subsections. In Sections 3.3.2 and 3.3.3 we give details of how to fit the embedded model and calculate variable importance at each internal node. In Section 3.3.4, we introduce a variable screening method that progressively mutes noise variables at each internal node. In Section 3.3.5, we extend one dimensional splits to high-dimensional splits by utilizing the available variable importance information at each internal node.

3.3.1 Reinforcement learning trees

RLT construction still follows the general pattern for an ensemble of binary trees: we first draw bootstrap samples to fit trees and then average. To construct a binary tree, a splitting variable and a splitting value is determined at each internal node, starting from the root node. This internal node is then split into two daughter nodes by grouping the observations using the selected variable and splitting value. The algorithm stops when the node sample size is sufficiently small. The key ingredient of RLT is the selection of splitting variable and also the method of constructing daughter nodes. These special features are carried out using the embedded model and variable importance measures. Table 3.1 summarizes the RLT algorithm.

Table 3.1: Algorithm for reinforcement learning trees

```
1.  Draw $M$ bootstrap samples from $D$.
2.  For the $m$-th bootstrap sample, where $m \in \{1, ..., M\}$, fit one RLT model $\widehat{f}_m$, using the following rules:
    a)  At an internal node $A$, fit an embedded model $\widehat{f}_A^*$ to the data in $A$, restricted to the set of variables $\{1, 2, ..., p\} \backslash \mathcal{P}_A^d$, i.e. $\mathcal{P} \backslash \mathcal{P}_A^d$, where where $\mathcal{P}_A^d$ is the set of muted variables at the current node $A$. Details are given in Section 3.3.2.
    b)  Using $\widehat{f}_A^*$, calculate the variable importance measure $\widehat{VI}_A(j)$ for each variable $X^{(j)}$, where $j \in \mathcal{P}$. Details are given in Section 3.3.3.
    c)  Split node $A$ into two daughter nodes using either i) or ii).
        i)  For a one-dimensional split, use the variable with the largest variable importance measure, namely $\arg\max_j \widehat{VI}_A(j)$, as the splitting variable. The cut point $c$ is chosen randomly and uniformly. We call this method RLT1.
        ii) For a high-dimensional split, a linear combination of variables is used. Details are given in Section 3.3.5. We call this method RLT$k$, where $k$ is the number of variables used in the linear combination.
    d)  Update the set of muted variable set $\mathcal{P}^d$ for the two daughter nodes by adding the variables with the lowest variable importance measures at the current node. Details are given in Section 3.3.4.
    e)  Apply a)–d) on each daughter nodes until node sample size is smaller than a pre-specified value $n_{min}$.
3.  Average $M$ trees to get a final model $\widehat{f} = M^{-1} \sum_{m=1}^{M} \widehat{f}_m$. For classification, $\widehat{f} = I\left(0.5 < M^{-1} \sum_{m=1}^{M} \widehat{f}_m\right)$.
```

3.3.2 Embedded model

To assess the variable importance $\widehat{VI}_A(j)$ for each variable j at any internal node A, we must first fit an embedded model to the internal node data. Note that at the root node, where the set of muted variables $\mathcal{P}^m = \emptyset$, all variables in the set $\mathcal{P} = \{1, 2, ..., p\}$ are considered in the embedded model and their variable importance measures will be assessed. However, as we move further down the tree, some variables will be muted and $\mathcal{P}^m \neq \emptyset$, then the embedded model will be fit using only the non-muted variable set $\mathcal{P} \backslash \mathcal{P}_A^d$. For the choice of the embedded model, we use random forests (Breiman, 2001). It is not necessary that random forests be used here. Alternatively, any learning method which is verified to be consistent with a certain convergence rate, for example, purely random forests, can be used to estimate the embedded model.

Suppose we are at an internal node A in the tree building process. To be specific, when a one-dimensional split is used, any internal node A can be expressed as a hypercube in the feature space, i.e. $A = \{(X^{(1)}, ..., X^{(p)}) : X^{(j)} \in (a_j, b_j] \subseteq [0, 1], \text{for}\, j \in \mathcal{P}\}$. Denote the samples at this internal node as $D_A = \{(\mathbf{X}_i, Y_i) : \mathbf{X}_i \in A\}$. We fit a random forests model, denoted by $\widehat{f}_A^*$, to the internal node data D_A with only variables that are in the set $\mathcal{P} \setminus \mathcal{P}_A^d$. For convenience, we use all the default settings in Breiman (2001) for the embedded random forests. To facilitate our later arguments, we denote the number of trees in the embedded model as M^* and denote each tree as $\widehat{f}_{A,m}^*$, for $m \in (1, 2, ..., M^*)$.

3.3.3 Variable importance

Since the purpose of fitting the embedded random forests is to determine the most important variable, we need to properly define a variable importance measure $VI_A(j)$ for each variable $j \in \mathcal{P}$ at an internal node A and use the embedded model to calculate the estimate $\widehat{VI}_A(j)$. The variable importance calculation in Breiman (2001) seems to be a natural choice here since we use random forests as the embedded model. We give the formal definition of the variable importance measure in the following. In Section 3.4 and Appendix section, we will carefully investigate the properties of VI_A and the asymptotic properties of its estimate $\widehat{VI}_A$.

Definition 3.3.1. *At any internal node A, denote $\tilde{X}^{(j)}$ as an independent copy generated from the marginal distribution of $X^{(j)}$ within A, the variable importance of the j-th variable within A, namely $VI_A(j)$, is defined by:*

$$\frac{E\big[\big(f(X^{(1)},...,\tilde{X}^{(j)},...,X^{(p)}) - f(X^{(1)},...,X^{(j)},...,X^{(p)})\big)^2|A\big]}{E\big[\big(Y - f(X^{(1)},...,X^{(j)},...,X^{(p)})\big)^2|A\big]},$$

where the $E[\cdot|A]$ is a conditional expectation defined by $E[g(Y,\mathbf{X})|A] = E[g(Y,\mathbf{X})|I(\mathbf{X} \in A)]$, for any function g.

In practice, following Breiman (2001)'s procedure, to calculate $\widehat{VI}_A(j)$ for each fitted embedded tree, we randomly permute the values of variable j in the out-of-bag (OOB) data (to mimic the independent and identical copy $\tilde{X}^{(j)}$), drop these permuted observations down the fitted tree and then calculate the resulting mean squared error (MSE) increase. Intuitively, when j is a strong variable, randomly permuting the values of $X^{(j)}$ will result in a large $\widehat{VI}_A(j)$, while randomly permuting the values of a noise variable should result in little or no increase in MSE, so $\widehat{VI}_A(j)$ should be small. Hence $\widehat{VI}_A(j)$ calculated from the embedded model can identify the variable with greatest need-to-be-split in the sense that it explains the most variation in the outcome variable Y in the current node (see Section 3.4). Another important property that we observe is that for all the variables in the muted set $\mathcal{P}_A^d$, since they are not involved in the embedded model $\widehat{f}_A^*$, randomly permuting their values will not increase MSE. Hence,

Table 3.2: Variable Importance

1. For the m-th tree $\widehat{f}^*_{A,m}$, $m \in (1, 2, ..., M^*)$, in the embedded model, do steps a)–c).

a. Select the corresponding m-th OOB (out-of-bag) data which consists of the data not selected in the m-th bootstrap sample.

b. Drop OOB data down the fitted tree $\widehat{f}^*_{A,m}$ and calculate mean squared error, $MSE_{A,m}$.

c. For each variable $j \in \mathcal{P} \setminus \mathcal{P}^d_A$, do the following:

i) Randomly permute the values of the j^th variable $X^{(j)}$ in the OOB data.

ii) Drop permuted OOB data down the fitted tree $\widehat{f}^*_{A,m}$, and calculate the permuted mean squared error, $PMSE^j_{A,m}$.

2. Average over M^* measurements to get the variable importance measure for variable j:

$$VI_A(j) = \frac{\sum_{m=1}^{M^*} PMSE^j_{A,m}}{\sum_{m=1}^{M^*} MSE_{A,m}} - 1.$$

for $j \in \mathcal{P}^d_A$, we must have $\widehat{VI}_A(j) = 0$. Table 3.2 gives details on how to assess the variable importance measure based on the embedded random forest estimator $\widehat{f}^*_A$.

3.3.4 Variable muting

As we discussed previously, with sample size reducing rapidly towards a terminal node during the tree construction, searching for a strong variable becomes increasingly difficult. The lack of signal from strong variables can eventually cause the splitting variable selection to behave completely randomly, and then the constructed model is similar to purely random forests. Hence, the muting procedure we introduce here is to prevent some noise variables from being considered as the splitting variable. We call this set of variables the muted set. At each internal node, we force p_d variables into the muted set, and we remove them from consideration as splitting variable at any branch of this internal node. On the other hand, to prevent strong variables from being removed from the model, we set a minimal number of p_0 variables that we always keep. This set of variables are called the protected set. We give the details of their definitions in the following. Note that both the muted set and protected set will be updated for each daughter nodes after a split is done. We first take a loot at the root node, then generalize the procedure to any internal node.

At the root node: At the root node we have $A = [0,1]^p$. After selecting the splitting variable, assume that the two resulted daughter nodes are A_L and A_R. Then we sort the variable importance measures $\widehat{VI}_A(j)$ calculated from the embedded model $\widehat{f}_A^*$ and find the p_d-th smallest value within the variable set $\mathcal{P}$ denoted by $\widehat{VI}_A^{p_d}$ and the p_0-th largest value denoted by $\widehat{VI}_A^{p_0}$. Then we define:

- The muted set for the two daughter nodes: $\mathcal{P}_{A_L}^d = \mathcal{P}_{A_R}^d = \{j : \widehat{VI}_A(j) \leq \widehat{VI}_A^{p_d}\}$, i.e. the set of variables with the smallest p_d variable importance measures.
- The protected set $\mathcal{P}_A^0 = \mathcal{P}_{A_L}^0 = \mathcal{P}_{A_R}^0 = \{j : \widehat{VI}_A(j) \geq \widehat{VI}_A^{p_0}\}$, i.e. the set of variables with largest p_0 variable importance measures. Note that the variables in the protected set will not be muted in any of the subsequent internal nodes.

At internal nodes: After the muted set and protected set have been initialized at the root split, we update the two sets in subsequent splits. Suppose at an internal node A, the muted set is $\mathcal{P}_A^d$, the protected set is $\mathcal{P}_A^0$ and the two daughter nodes are A_L and A_R. We first update the protected set for the two daughter nodes by adding the splitting variable(s) into the set:

$$\mathcal{P}_{A_L}^0 = \mathcal{P}_{A_R}^0 = \mathcal{P}_A^0 \cup \{\text{splitting variable(s) at node} A\}.$$

Note that when a one-dimensional split is used, the splitting variable is simply $\arg\max_j \widehat{VI}_A(j)$, and when a high-dimensional split is used, multiple variables could be involved.

To update the muted set, after sorting the variable importance measures $\widehat{VI}_A(j)$, we find the p_d-th smallest value within the restricted variable set $\mathcal{P} \setminus \mathcal{P}_A^d \setminus \mathcal{P}_A^0$, which value is denoted $\widehat{VI}_A^{p_d}$. Then we define the muted set for the two daughter nodes as

$$\begin{aligned}\mathcal{P}_{A_L}^d &= \mathcal{P}_{A_R}^d \\ &= \mathcal{P}_A^d \cup \{j : \widehat{VI}_A(j) \leq \widehat{VI}_A^{p_d}\} \setminus \mathcal{P}_A^0.\end{aligned}$$

Remark 3.3.2. *There are two tuning parameters in the muting procedure, the number of protected variables p_0 and the number of extra muted variables at each split p_d. Ideally, we want to choose $p_0 = p_1$, which is the number of strong variable, hence the strong variables can always be protected. p_d can be any positive value less than p_2, and the noise variables will all be muted after finitely many splits. In practice, since we have little information about how large p_1 is, we want to set p_0 to be a reasonable large number, say $\sqrt{p}$ for a high-dimensional situation. Our updating procedure will add a strong variable into the protected set when it is used as a splitting variable. p_d dose not need to be a fixed number. It can vary depending on $|\mathcal{P} \setminus \mathcal{P}_A^d|$, which is the number of nonmuted variables at each internal node. In Section 3.5 we will evaluate different choices for p_d such as 0 (no muting), $20\% \cdot |\mathcal{P} \setminus \mathcal{P}_A^d|$ (moderate muting, which is suitable for most situations), and $50\% \cdot |\mathcal{P} \setminus \mathcal{P}_A^d|$ (very aggressive muting).*

3.3.5 High-dimensional cuts

Using a linear combination of several variables to construct a splitting rule was considered in Breiman (2001). However, the idea never achieved much popularity. The major difficulty is computational intensity. Exhaustively searching for a linear combination of $k < p$ variables means computing and comparing approximately n^k

different splitting possibilities (any k dimensional cut can be defined by k points in the feature space, and there can be as many as $n(n-1)\cdots(n-k)$ possible ways to select these k points: when n is large, this is approximately n^k). By further considering the possibility of drawing k from p total variables, it seems that the computational burden overshadows the benefit.

However, the proposed reinforcement learning splitting variable selection approach reopens the possibility of a high-dimensional split. We develop our proposed high-dimensional cut based on the following two facts. First, the splitting rule should only involve important variables. Second, the magnitude of coefficients in the linear combination should be positively related to the variable importance measure. This means that if we view the linear combination as an axis in a high-dimensional space, the axis should lean more towards the strong variables (with large variable importance) and be almost orthogonal to the noise variables (with zero variable importance).

Before presenting the algorithm for the high-dimensional cut, we define two parameters that we use to control the complexity of a high-dimensional cut:

- k: The maximum number of variables considered in the linear combination. Note that when $k = 1$, this simplifies to the usual one dimensional cut.
- α: The minimal variable importance, taking values in $(0, 1)$, of each variable in this linear combination in terms of the percentage of maximum $\widehat{VI}$ at the current node. For example, if $\alpha = 0.5$ and $max_j(\widehat{VI}(j)) = 10$ at the current node, then any variable with $\widehat{VI}$ less than 5 will not be considered for the linear combination. The purpose of this parameter is to ensure that the high-dimensional cut does not involve noise variables.

The high-dimensional split focuses on creating a linear combination of the form $\mathbf{X}^T\boldsymbol{\beta}$, which can be viewed as a high-dimensional axis, where $\boldsymbol{\beta}$ is a coefficient vector with dimension $p \times 1$. We can then project each observation onto this axis to provide a scalar ranking for splitting. We first give the definition of $\widehat{\boldsymbol{\beta}}_j(A)$ for each $j \in \{1, ...p\}$ at node A:

$$\begin{aligned}\widehat{\boldsymbol{\beta}}_j(A) \;\; &= \;\; \widehat{VI}_A(j) \cdot I[\widehat{VI}_A(j) > 0] \cdot I[\widehat{VI}_A(j) \geq \widehat{VI}_A^{(k)}] \\ &\cdot I[\widehat{VI}_A(j) \geq \alpha \cdot max_j \widehat{VI}_A(j)] \cdot sign(\rho_{X^{(j)},Y}(A)),\end{aligned}$$

where $\rho_{X^{(j)},Y}(A)$ is the Pearson's correlation coefficient between $X^{(j)}$ and Y within node A.

Now we give the details of each component in $\widehat{\boldsymbol{\beta}}_j(A)$. The first component is simply the variable importance measure of $X^{(j)}$. The second to the fourth component set restrictions based on the value of $\widehat{VI}_A(j)$, so that $\widehat{\boldsymbol{\beta}}_j(A)$ is non-zero only if: $\widehat{VI}_A(j)$ is positive, larger or equal to the k-th largest $\widehat{VI}$ in the current node, and larger than $\alpha \cdot 100\%$ of the largest $\widehat{VI}$ in the current node. These restrictions will eliminate all muted variables and the variables with small $\widehat{VI}$. The last component sets the sign of $\widehat{\boldsymbol{\beta}}_j(A)$ so that variables with the same trend have the same sign.

After having each $\widehat{\boldsymbol{\beta}}_j(A)$, we can calculate $\mathbf{X}_i^T \widehat{\boldsymbol{\beta}}(A)$ for each observation $\mathbf{X}_i$ in the current node. This is precisely the scalar projection of each observation for ranking mentioned above. We then select a random uniform splitting value c for this projection to separate the current node into two daughter nodes: $\{i : \mathbf{X}_i^T \widehat{\boldsymbol{\beta}}(A) \leq c, \mathbf{X}_i \in A\}$ and $\{i : \mathbf{X}_i^T \widehat{\boldsymbol{\beta}}(A) > c, \mathbf{X}_i \in A\}$.

3.4 Theoretical results

In this section, we develop large sample theory for the proposed RLT method. We only focus on the proof for one-dimensional splits (RLT1) in a regression problem with fixed muting parameters p_d, and we assume that the number of protected variable p_0

is larger than p_1, the number of strong variables. The main results are Theorem 3.4.7 which bounds below the probability of using strong variables as the splitting rule, and Theorem 3.4.8 which established consistency and derives an error bound for RLT1. We assume, for convenience in the proofs, that the covariates $\mathbf{X}$ are generated uniformly from the feature space $\mathcal{X} = [0, 1]^p$. First, we need several other key assumptions.

Assumption 3.4.1. *There exist a set of strong variables $\mathcal{S} = (1, ..., p_1)$ such that $f(X) = E[Y|\mathbf{X}] = E[Y|X^{(j)}, j \in \mathcal{S}]$ and $P\left(\frac{\partial f}{\partial X^{(j)}} = 0\right) = 0$ for $j \in \mathcal{S}$. The set of noise variables is then $\mathcal{S}^c = (p_1 + 1, ..., p)$. The true function f is Lipschitz continuous with Lipschitz constant c_f.*

Remark 3.4.2. *The requirements for the distribution of* $\mathbf{X}$ *and the feature space seem restrictive, however, for any distribution with independent marginals, we can transform the distribution into the required multivariate uniform distribution. A direct consequence of this assumption is that, due to the construction of the splitting rules, any internal node can be now viewed as a hypercube in the feature space $\mathcal{X}$, i.e. any internal node $A \subseteq [0, 1]^p$ has the form*

$$\{(X^{(1)}, ..., X^{(p)}) : X^{(j)} \in (a_j, b_j] \subset [0, 1],\ for\, j \in 1, ..., p\}. \tag{3.1}$$

Through out the rest of this paper, we will use the terms "internal node" and "hypercube" interchangeably provided that the context is clear.

We need to precisely define how "strong" a strong variable is, not only globally, as we did in Definition 3.4.1, but also locally at any internal node A. Thus we have the following assumption for the lower bound of variable importance:

Assumption 3.4.3. *For any hypercube A defined in the form of Equation 3.1 with the property that, for any strong variable j, $\min_{i \in \{\mathcal{S} \backslash j\}} (b_i - a_i) \geq \delta > 0$, there exist positive valued monotone functions $\psi_1(\delta)$ and $\psi_2(b_j - a_j)$, such that the variable importance of any strong variable j is bounded below by*

$$\frac{VI_A(j)}{\psi_2(b_j - a_j)} \geq \psi_1(\delta), \tag{3.2}$$

where $VI_A(j)$ is as defined in Definition 3.3.1.

Remark 3.4.4. *This assumption basically requires that the surface of f can not be extremely flat, however, this does not require a lower bound on $\left|\partial f / \partial X^{(j)}\right|$. It is easy to verify Assumption 3.4.3 for a linear model, since the variable importance of a strong variable j does not depend on the interval length of other variables. In this case, we have $\psi_1(\delta) \equiv 1$ and $\psi_2(b_j - a_j) = (b_j - a_j)^2$. If f is a polynomial function with any kind of interaction, for small values of δ and $b_j - a_j$, $\psi_1(\delta)$ and $\psi_2(b_j - a_j)$ can be approximated by polynomial functions δ^{ζ_1} and $(b_j - a_j)^{\zeta_2}$, where ζ_2 is the lowest order of $X^{(j)}$ in f, and ζ_1 is the lowest order of all other variables in the interaction.*

Assumption 3.4.5. *With $f(\mathbf{X})$ being the true underlying function, the observed value are $Y_i = f(\mathbf{X}_i) + \epsilon_i$, where the ϵ_is are i.i.d. with mean 0 and variance σ^2. Moreover, the following Bernstein condition on the moments of ϵ is satisfied:*

$$E(|\epsilon|^m) \leq \frac{m!}{2} K^{m-2},\ m = 2, 3, ..., \tag{3.3}$$

for some constant $1 \leq K < \infty$.

Another assumption is on the embedded model. Although we use random forests as the embedded model in practice, we do not want to rule out the possibility of using any other kinds of embedded models. Hence we make the following assumption for the embedded model, which is at least satisfied for purely random forests:

Assumption 3.4.6. *The embedded model* $\widehat{f}^*$ *fitted at any internal node* A *with internal sample size* n_A *is uniformly consistent with an error bound: there exist some fixed constant* $0 < K < \infty$ *so that for any* $\delta > 0$, $P\left(|\widehat{f}^* - f| > \delta \middle| A\right) \leq C \cdot e^{-\delta \cdot n_A^{\eta(p)} \cdot K}$, *where* $0 < \eta(p) \leq 1$ *is a function of the dimension* p, *and the conditional probability on* A *means that the expectation is taken within the internal node* A. *Note that it is reasonable to assume that* $\eta(p)$ *is a non-increasing function of* p *since larger dimensions should result in poorer fitting. Furthermore, we assume that the embedded model* $\widehat{f}^*$ *lies in a class of functions* $\mathcal{F}$ *with finite entropy integral under the* $L^2(P)$ *norm (Van Der Vaart and Wellner, 1996).*

Now we present two key results, Theorem 3.4.7 and Theorem 3.4.8. Theorem 3.4.7 analyzes the asymptotic behavior of the variable importance measure and estalisheds the probability for selecting the true strong variables and muting the noise variables. For simplicity, we only consider the case that one RLT1 tree is fitted to the entire dataset, i.e $M = 1$ and the bootstrap ratio is 100%. For the embedded model, we fit only one tree using half of the data and calculate the variable importance using the other half. We set the minimum sample size for each terminal node in RLT1 to be n^γ where $0 < \gamma < 1$. At each internal node, the splitting point c is chosen uniformly between the q-th and $(1-q)$-th quintile of each variable, where $q \in (0, 0.5]$. The smaller

q is, the more diversity it induces. When $q = 0.5$, this degenerates into a model where each internal node is always split into two equally sized daughter nodes.

Theorem 3.4.7. *For any internal node $A \in \mathcal{A}_n$ with sample size n_A, where $\mathcal{A}_n$ is the set of all internal nodes in the constructed RLT, define $\widehat{j}_A$ to be the selected splitting variable at A and let p_A denote the number of non-muted variables at A. Then, under Assumptions 3.4.1, 3.4.3, 3.4.5, 3.4.6, we have,*

a. $P\left(\widehat{j}_A \in \mathcal{S}\right) \geq 1 - C_1 e^{-\psi_1(\frac{n_A}{n}) \cdot \psi_2(\frac{n_A}{n}) \cdot n_A^{\eta(p_A)}/K_1}$*, i.e. with probability close to 1, we always select a strong variable as the splitting variable.*

b. $P\left(VI_A(\widehat{j}_A) > 2 \cdot \max_j VI_A(j)\right) \geq 1 - C_2 e^{-\psi_1(\frac{n_A}{n}) \cdot \psi_2(\frac{n_A}{n}) \cdot n_A^{\eta(p_A)}/K_2}$*, i.e. for any internal node in the constructed RLT model, the true variable importance measure for the selected splitting variable is at least half of the true maximum variable importance with probability close to 1.*

c. The protected set $\mathcal{P}_A^0$ contains all strong variables, i.e. $P\left(\mathcal{S} \in \mathcal{P}_A^0\right) > 1 - C_3 e^{n^{\eta(p)}/K_3}$.

Note that in the above three results, ψ_1, ψ_2, and the constants C_k and K_k, $k = 1, \ldots, 3$, do not depend on p_A or the particular choice of A.

As we discussed in Remark 3.4.4, for any polynomial function, $\psi_1(\delta)$ and $\psi_2(b_j - a_j)$ can be approximately represented by δ^{ζ_1} and $(b_j - a_j)^{\zeta_2}$. Since $n_A > n^{\gamma}$, we have $n_A/n > n^{\gamma-1}$. Hence, to have the probability in Theorem 3.4.7 converging to 1, since our model eventually only involves p_0 variables, we need to tune the terminal node size parameter γ such that $n^{(\gamma-1)\zeta_1} \cdot n^{(\gamma-1)\zeta_2} \cdot n^{\gamma\eta(p_0)} \to \infty$, which requires that $\gamma > \frac{\zeta_1+\zeta_2}{\zeta_1+\zeta_2+\eta(p_0)/2}$. For a linear model, we only need $\gamma > 2/(2 + \eta(p_0))$. However, in some worst case scenarios where f is relatively flat, γ has to be close to 1. This is in fact a very intuitive result because if, for example, $f = (X^{(j)})^{100}$, then we need a much longer interval over $X^{(j)}$ to detect a positive variable importance measure.

To show consistency and an error bound for RLT, we verify that the entire RLT is constructed using only strong variables provided γ is properly chosen, and that the total variation can be bounded by the variable importance measures at each terminal node, which converges to zero eventually. The most important result at this juncture is to show that the splitting variable selection process shrinks the strong variable interval length to zero at all terminal nodes. On the other hand, the variable muting mechanism relaxes the choice of γ so that it only depends on p_0 rather than p, hence the error bound for RLT only depends on p_0. To show this property, we separate the constructed RLT into two parts: the first (upper) part of the tree consists of all internal nodes with sample size larger than n^{γ^*}, where γ^* is a value between 0 and 1 such that $\psi_1(n^{\gamma^*-1}) \cdot \psi_2(n^{\gamma^*-1}) \cdot n^{\gamma^*\eta(p)} \to \infty$. Within this part of the tree, all noise variables are gradually muted so that only p_0 protected variables, which include all strong variables, remain active in each node. Note that γ^*, unlike the terminal node size parameter γ, is not a tuning parameter, but is an endogenous value determined by the true function f, the embedded model convergence rate η, and p. By the properties of ψ_1, ψ_2 and η, γ^* must be larger than γ. The second (lower) part part of the tree consists of all subsequent nodes with sample size smaller than n^{γ^*}. Since in these nodes, the embedded model only involves the p_0 protected variables, we only need to tune γ such that $\psi_1(n^{\gamma-1}) \cdot \psi_2(n^{\gamma-1}) \cdot n^{\gamma\eta(p_0)} \to \infty$, implying that γ depends only on $\eta(p_0)$, and thus the convergence rate for RLT only depends on p_0 and not p.

Theorem 3.4.8. *Under Assumptions 3.4.1, 3.4.3, 3.4.5, and 3.4.6,* $E\left[(\widehat{f} - f)^2\right] = O_p(n^{-C})$*, where C is a constant that depends only on γ, q, and p_1. Moreover, C is a strictly monotone decreasing function in p_1.*

Table 3.3: Parameter settings

Lasso	10-fold cross-validation is used with $\alpha = 1$ for lasso and λ is set to minimize cross-validation error.
Boosting	10-fold cross-validation is used. Number of trees = 3000. Optimal number of boosting iterations is determined by cross-validation.
BART	All settings are default except, when $p \geq n$, the naive estimator $\widehat{\sigma}$ is used (as implemented in Chipman et al. 2008).
RF	All settings are default.
RF2	Select the top $\sqrt{p}$ important variables from a single random forests model and refit.
RLTk	$M = 50$ trees are fit to each RLT model. We consider $k = 1, 2, 5$, namely RLT1, RLT2 and RLT5. For each of these models, as mentioned in Remark 3.3.2, we consider no muting ($p_d = 0$), moderate muting ($p_d = 20\% \cdot \lvert\mathcal{P} \setminus \mathcal{P}_A^d\rvert$ at any node A), and aggressive muting ($p_d = 50\% \cdot \lvert\mathcal{P} \setminus \mathcal{P}_A^d\rvert$ at any node A). To be on par with RF2, we set the number of protected variables p_0 to be $\sqrt{p}$. We also set terminal node size $n_{min} = n^{\frac{1}{3}}$.

3.5 Numerical studies

3.5.1 Competing methods and parameter settings

We compare our method with several major competitors, including the linear model with lasso, as implemented in the R package “glmnet” (Friedman et al. 2008); random forests (Breiman 2001), as implemented in the R package “randomforest”; gradient Boosting (Friedman 2001), as implemented in the R package “gbm”; and Bayesian Additive Regression Trees (Chipman et al. 2008), as implemented in the R package “BayesTree”. We also include another interesting version of random forests (RF2), which fits the model, selects a set of most important variables, and refits using only these variables. For our proposed reinforcement learning trees (RLT), we include nine different versions, consisting of combinations of different tuning parameter values. The details for all simulation settings are given in the following Table 3.3:

3.5.2 Simulation scenarios

We create four simulation scenarios that represent different aspects which usually arise in machine learning. Such aspects include size of dimension, correlation between variables, and non-linear structure. For each scenario, we generate 200 training samples to fit the model and 1000 test samples to calculate the prediction mean squared error (MSE). Each simulation is repeated 200 times, and the averaged MSE is presented. We now describe each of our simulation settings in the following:

Scenario 1: Classification with small p. Set $p = 10$, and draw X_i independent uniforms from $[0, 1]^p$. Set $\mu_i = \Phi(10 \times (X_{i,1} - 1) + 20 \times |X_{i,2} - 0.5|)$, where Φ denotes the standard normal $c.d.f.$ Draw Y_i independently from $binomial(\mu_i)$.

Scenario 2: Non-linear model with correlated covariance. Set $p = 100$. To impose correlation, draw Z_i and R_i as independent uniforms from $[0, 0.8]^p$ and $[0, 0.2]$, respectively. Set the covariate vector $X_i = (Z_{i,1} + R_i, Z_{i,2} + R_i, ..., Z_{i,p} + R_i)$ and $Y_i = 10sin(\pi X_{i,1}X_{i,2}) + 20(X_{i,3} - 0.5)^2 + \epsilon_i$, where the ϵ_i are i.i.d $N(0, 1)$.

Scenario 3: Strong correlation and no marginal effect. Set $p = 100$, and draw X_i independently from $N(\mathbf{0}_{p\times1}, \Sigma_{p\times p})$, where $\Sigma_{i,j} = \rho^{|i-j|}$ and $\rho = 0.5$, and $Y_i = 5(X_{i,10}X_{i,30}) + \epsilon_i$, where the ϵ_i are i.i.d $N(0, 1)$.

Scenario 4: linear structure with strong correlation and large p. Set $p = 300$, and draw X_i independently from $N(\mathbf{0}_{p\times1}, \Sigma_{p\times p})$. To increase correlation, we set $\Sigma_{i,j} = \rho^{|i-j|} + 0.2 \cdot I_{(i\neq j)}$ and $\rho = 0.5$, and $Y_i = 5(X_{i,10} + X_{i,20} + X_{i,30}) + \epsilon_i$, where the ϵ_i are i.i.d $N(0, 1)$.

The first three scenarios all contain some non-linear effects which would not be captured by the Lasso. Hence we expect the Lasso to perform worse compared to other tree-based methods. However in Scenario 4, we expect the lasso to perform best due to the underlying linear model. Also, under such a linear structure, RLT2 and RLT5 should perform better than RLT1 since the linear combination split can utilize the samples in a much more efficient way. In all scenarios, we expect RF2 to perform better than RF since the number of strong variables is always less than $\sqrt{p}$, and thus the variable selection done in RF2 should be beneficial.

3.5.3 Simulation results

Table 3.4 summarizes testing sample MSE for each simulation setting. In Figure 3.1, we choose three RLT methods, RLT1 with no muting, RLT2 with moderate muting and RLT5 with aggressive muting, to plot against competing methods. There is clear evidence that under almost all settings, the proposed splitting variable selection, high-dimensional cut, and variable muting procedures all work individually and also work in combination. In general, the results show preference towards RLTk methods in general, although the method falls behind the Lasso for the linear model, which is expected. RLTk methods show advantages over all competing methods on capturing the non-linear effects in scenarios 1, 2 and 3. Scenario 3 provides an interesting illustration of how the splitting variable selection works, as is shown by RLT1 under no muting, where the MSE is reduced by up to 60.0%. When there are no marginal effects, and when the dimension is reasonably high, none of the competing methods seem to be able to capture a clear pattern. Even by reducing the dimension from 100 to $\sqrt{100} = 10$, as is done in RF2, random forests produce large MSEs. However, a slight signal in the variable importance measure from the embedded random forests can push the splits onto strong variables and improve the performance.

The improvement obtained from high-dimensional splits is also profound. In linear models, utilizing high-dimensional splits can yield huge improvements over RLT1 especially when no muting is implemented. The MSE reduction obtained by going from RLT1 to RLT5 is 39.0% (under no muting) in scenario 4. The reason is that under such a structure, linear combination splits cut the feature space more efficiently. When there is no linear combination structure, a high-dimensional split may not always be beneficial. As can be seen in scenario 3, although RLT's are significantly better than competing methods, both RLT2 and RLT5 perform slightly worse than RLT1. However, the decrease in performance is slight because of the "α" parameter enforced in the splitting process. The resulting threshold on variable importance prevents too many noise variables from being employed in the linear combination split.

When comparing different muting procedures, we also see interesting results. In scenarios 1, 2 and 4, more aggressive muting procedures improve the performance of RLT regardless of whether high-dimensional splits are implemented. In scenario 4, the MSE is reduced by 38.9%, when going from no muting to aggressive muting for RLT1, and by 29.9%, when going from no muting to moderate muting for RLT1. An interesting case is scenarios 3, where the muting procedure harms the performance, although the performance is still better than competing methods. Note that in scenario 3, a setting with no marginal effect and only two strong variables, a very aggressive muting procedure appears to mute the strong variables early on so that they are ruled out from the model. Considering that the embedded model (RF) is not especially accurate in this situation, aggressive muting may not be a good choice for scenarios 3.

3.5.4 Data analysis example

The diagnostic Wisconsin breast cancer database (Mangasarian et al. 1995) has been a popular dataset for evaluating machine learning. We obtained the data from the UC Irvine Machine Learning Repository (http://archive.ics.uci.edu/ml/). The dataset contains diagnostic results from 569 subjects, classed as either "benign" or "malignant". A total of 30 features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. The features describe characteristics of the cell nuclei present in the image, such as radius, texture, perimeter, area, etc. In our analysis of this data,

Figure 3.1: Box plot of prediction Mean Squared Error

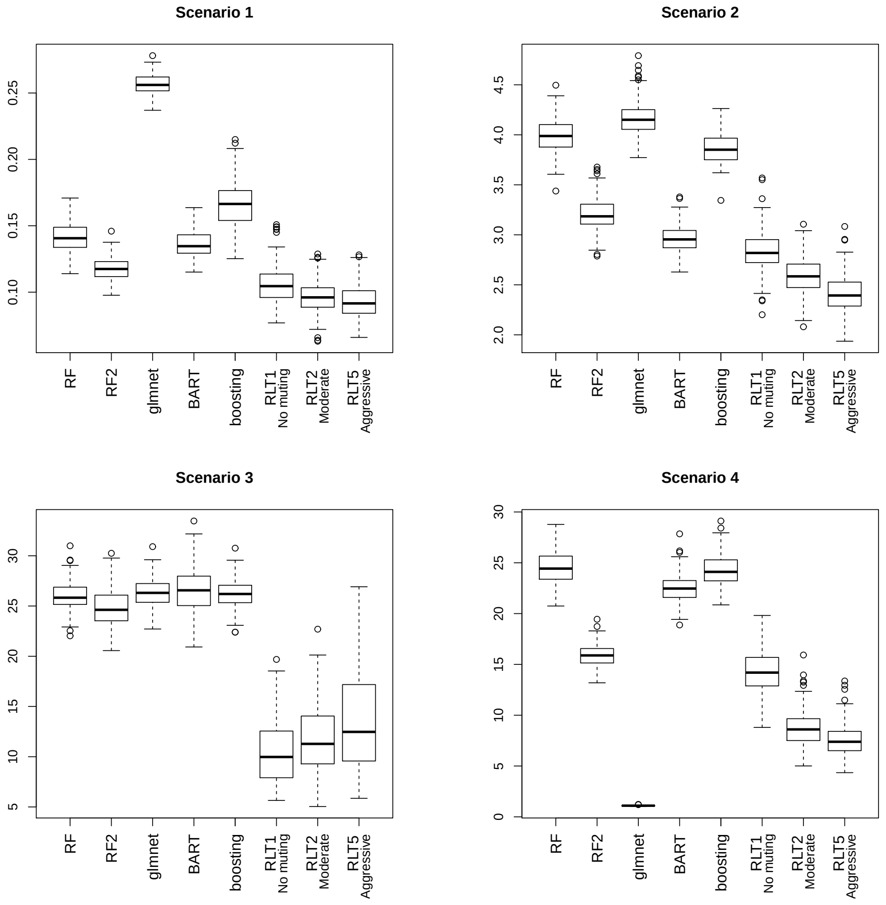


Table 3.4: Prediction Mean Squared Error

		Scenario 1	Scenario 2	Scenario 3	Scenario 4
RF		0.142	4.005	25.811	24.658
RF2		0.118	3.217	24.449	15.962
glmnet		0.257	4.191	26.100	1.099
BART		0.137	2.963	26.358	22.611
boosting		0.167	3.876	25.927	24.306
Muting	RLTk				
No	RLT1	0.106	2.831	9.774	14.271
	RLT2	0.100	2.698	10.209	9.103
	RLT5	0.101	2.706	10.421	8.709
Moderate	RLT1	0.098	2.658	11.644	10.009
	RLT2	0.096	2.593	11.938	8.682
	RLT5	0.096	2.597	11.917	8.525
Aggressive	RLT1	0.093	2.468	13.568	8.726
	RLT2	0.093	2.415	14.020	7.618
	RLT5	0.093	2.408	14.045	7.556

we want to compare the performance of different methods and also demonstrate the impact of increased dimension on prediction error.

The original data is standardized to let each covariate have mean zero and variance one. We keep the exact same parameter settings given in section 4.1 and create an independent set of new covariates to increase the total number of covariates p by 100, 200, 300, 400 and 500. These extra covariates are independent standard normal random deviates. We then randomly sample 300 observations without replacement from the total of 569 as the training dataset, and use the remaining observations as a testing sample to compute the misclassification rate. Due to the high dimension, this procedure is repeated 500 times and averaged to stabilize the results.

The misclassification rates are summarized in Table 3.5. We picked three RLT method: RLT1 with no muting (the overall worst RLT method), RLT2 with moderate muting and RLT5 with aggressive muting, and plotted them against competing methods in Figure **??**. When only the original 30 covariates are used, glmnet performs best with a misclassification rate of 3.1%, followed by RLT5 with no muting (3.3%), all moderate muting RLT's ($3.3 \sim 3.4\%$), BART (3.8%) and RLT2 with no muting (3.8%). As the dimension reaches 530, RLT become the dominant methods with misclassification rates in the range of $3.5 \sim 3.8\%$, except RLT1 with no muting and RLT2 with no muting. glmnet (4.1%) and RF2 (4.4%) are the best two among the competing methods.

It is interesting to observe two sets of comparisons here: RLT1 with no muting vs. RF; and aggressive RLT's vs. RF2. RLT1 with no muting and RF start off with similar performance when $p = 30$. However, as the dimension increases, the reinforcement learning variable selection starts to show its benefit and eventually reduces the misclassification rate by 10.79% from RF. On the other hand, the misclassification rates for both RF2 and aggressive RLT methods decrease in this simulation. Keeping in mind that both methods will exclude a large proportion of variables, it is not surprising to see this pattern. With only 30 covariates in the initial model, RF2 will only consider the best 5 variables, and aggressive RLT's will mute, on average, 22.5 (75%) variables in the first two splits and only 5 variables are protected against muting. This causes both of them to very likely miss some true strong variables. As p increases, the methods will eventually be able to fit the model with the most strong variables included. However, aggressive RLT's are uniformly better in this comparison regardless of the implementation of high dimensional splits.

The plot also shows an important advantage of RLT: it performs consistently across changing dimension, which means that it has good immunity to dimension. While being the second best method at $p = 30$, RLT5 with moderate muting has its misclassification rate increase by only 10.35% when p is increased to 530. This is quite impressive compared to glmnet's increase of 33.64%, RF's of 24.33% and BART's of 50.58%.

Figure 3.2: Misclassification rate by increasing dimension

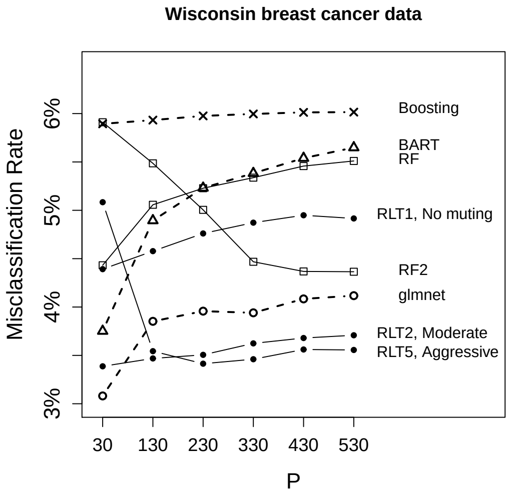


Table 3.5: Diagnostic Wisconsin Breast Cancer Dataset misclassification rate

		p=30	p=130	p=230	p=330	p=430	p=530
RF		0.044	0.051	0.052	0.053	0.055	0.055
RF2		0.059	0.055	0.050	0.045	0.044	0.044
glmnet		0.031	0.039	0.040	0.039	0.041	0.041
BART		0.038	0.049	0.052	0.054	0.055	0.056
Boosting		0.059	0.059	0.060	0.060	0.060	0.060
Muting	RLTk						
No	RLT1	0.044	0.046	0.048	0.049	0.049	0.049
	RLT2	0.038	0.040	0.041	0.042	0.043	0.043
	RLT5	0.033	0.036	0.037	0.038	0.039	0.038
Moderate	RLT1	0.034	0.035	0.036	0.037	0.038	0.038
	RLT2	0.034	0.035	0.035	0.036	0.037	0.037
	RLT5	0.033	0.034	0.035	0.036	0.037	0.037
Aggressive	RLT1	0.051	0.036	0.035	0.035	0.036	0.036
	RLT2	0.051	0.035	0.034	0.035	0.036	0.036
	RLT5	0.050	0.035	0.034	0.035	0.035	0.035

3.5.5 Numerical study conclusion

In this numerical study section, we compared the performance of the proposed RLT method with several popular learning tools. Under both simulated scenarios and the Wisconsin Breast Cancer Dataset, the results favor RLT methods. There is a significant improvement over competing methods in most situations, however, the results vary some depending on the choice of tuning parameters. RLT methods with moderate muting generally perform the best and most stably across different settings, and incorporating high dimensional splits seems almost always beneficial. On the other hand, when the dimension is relatively low, aggressive muting can sometimes cause strong variables to be muted and harm the performance; when the dimension is high, aggressive muting starts to show a noticeable benefit. The behavior of different muting procedures needs further analysis, and we do not suggest using aggressive muting, unless the dimension is very high, due to its apparent instability in low-dimensional situation.

3.6 Discussion

We proposed reinforcement learning trees in this paper. By fitting an embedded random forest model at each internal node, and calculating the variable importance measures, we can increase the chance of selecting the most important variables to cut and thus utilize the available training samples in an efficient way. The proposed high-dimensional splitting strategy extends the use of variable importance measures and creates splitting rules based on a linear combination of variables. The variable muting procedures further concentrates the splits on the strong variables at deep nodes in the tree where the node sample size is small. All of these procedures take advantage of Reinforcement Learning and yield significant improvement over existing methods especially when the dimensional is high and the true model structure is sparse. There are several remaining issues we want to discuss in this section including the choice of tuning parameters, computational issue, and future research directions.

3.6.1 Choosing the tuning parameters

The number of trees M in RLT does not need to be very large to achieve good performance. In all simulations, we used $M = 50$. The use of high-dimensional splits (RLT2 and RLT5) seems beneficial in most situations, and the drawbacks are negligible even when there is no linear effect. Hence we recommend choosing $k = 2$ to 5 and using $\alpha = 0.5$. In all simulations, we use terminal node size equal to $n^{1/3}$ which seems to perform reasonably well. However, the optimal choice of γ needs further theoretical analysis. The choice for muting parameters seems tricky. Ideally, the choice of p_0 and p_d should depend on sample size n, dimension p, and even the performance of the embedded model, which can be hard to evaluate. In general, we recommend using a moderate muting procedure, i.e., $p_d = 20\% \cdot |\mathcal{P} \setminus \mathcal{P}_A^d|$ at each internal node, and using $p_0 = \sqrt{p}$. However, the choice of these parameters is flexible and should depend on the setting. For example, when p is extremely large, a more aggressive muting procedure should probably be used to force a sparse structure. These adjustments require testing on a massive number of datasets and will be a focus area for our future research.

3.6.2 Computational intensity and R package "RLT"

The computational cost of RLT is higher than the original random forests, which is expected since more computations need to be done at each internal node to search for the optimal splitting variable. In a worst case scenario, RLT will fit as many as $n^{1-\gamma}$, $0 < \gamma < 1$ embedded models if we require the terminal sample size to be at least n^{γ}. However, this is not entirely necessary because as splitting moves towards a terminal

Table 3.6: Computational time of RLT (in seconds)

	p=100	p=200	p=300	p=400	p=500
n=100	2.8	5.0	7.2	8.9	11.0
n=200	8.8	15.8	21.3	27.8	35.4
n=300	15.7	28.9	39.7	51.9	68.9
n=400	23.4	42.2	60.4	77.6	105.2
n=500	30.5	56.7	80.2	105.9	140.1

node, the sample size shrinks rapidly and will not require as much computation as needed at root nodes. Hence, the number of trees in the embedded model can decrease as the internal node sample size decreases. Moreover, the muting procedure eliminates a large proportion of variables so that the embedded model takes less time to fit. On the other hand, RLT carries out high-dimensional splitting at little extra computational cost, which compared to exhaustive searching, is much less computationally intensive. The proposed methods is implemented in the R package "RLT" based on R3.0.1. The current version of "RLT" is available at author's personal website `http://www.bios.unc.edu/~rzhu`. Parallel computing and extremely randomized trees are implemented to reduce the computational burden. The following table summarizes the computation time in seconds for Scenario 3 with aggressive muting and no linear combination split on a 4 core CPU.

Chapter 4

Reinforcement learning trees for survival data

4.1 Introduction

In this third part of my dissertation, we extend reinforcement learning trees (Zhu et al., 2012) to right censored survival data. Reinforcement learning trees demonstrates significantly improved accuracy especially under high-dimensional sparse settings. The new proposed method translates the advantages of reinforcement learning trees into the high-dimensional survival data setting and forces the constructed trees to focus their splits on strong variables via two key components. First, at each internal node, an embedded survival model is fit to evaluate the overall variable importance using integrated Brier score. The most important variable is used as the split, this contrasts with traditional tree models where the variable with the largest marginal effect is used. The proposed implementation significantly increases the chance of a strong variable being used as the splitting rule and hence results in improved accuracy. The second key component is a muting process during the construction of each tree. The variables with the smallest variable importance measures are gradually muted (not being considered in daughter nodes) to prevent too many noise variables from entering the model and resulting in improved model fitting.

The framework of the survival model proposed in this paper is based on reinforcement learning trees (RLT). In regression modeling, under mild conditions, RLT is shown to have a convergence rate that does not depend on the original number of variables p but instead on a pre-selected number of protected variables p_0 (Zhu et al., 2012) that can be significantly smaller than p. Under high-dimensional sparse settings, this is a very desirable property. We would like to believe that the proposed model should also enjoy similar properties, however, the asymptotic behavior of the variable importance measure requires further investigation and is beyond the scope of this paper. The advantage of using reinforcement learning is that it can break down potential interactions in the true model structure (The checker-board model in Biau et al. (2008) and Zhu et al. (2012) is a notorious such example), which is not always detectable using traditional tree-based models. We shall see from the simulation section that through the embedded model and proper variable importance measure, the proposed method can achieve a similar goal for right censored survival data.

4.2 Notation and the survival model

Let $\mathbf{X} = (X^{(1)}, ..., X^{(p)})$ denote a set of p covariates from a feature space $\mathcal{X}$, and we use $\mathcal{P}$ to denote the set $\{1, 2, ..., p\}$. The failure time T given $\mathbf{X} = x$ is generated from the distribution function $F_x(\cdot)$, and we let the survival function $S_x(\cdot) = 1 - F_x(\cdot)$. The censoring time C given $X = x$ follows a conditional distribution function $G_x(\cdot)$. Each observation is a triplet (Y, δ, X), where $Y = (T \wedge C)$, and $\delta = I(T \leq C)$. We observe a sample of n i.i.d. training observations $\mathcal{D}_n = \{(Y_1, \delta_1, \mathbf{X}_1), ..., (Y_n, \delta_n, \mathbf{X}_n)\}$. We further assume a conditionally independent censoring mechanism which posits that T and C are independent given covariates X. This assumption guarantees that the logrank test statistics used in splitting criteria are asymptotically normally distributed, hence the comparison between different splitting rules is appropriate. Another typical assumption in high-dimensional data is sparsity, which assumes that the survival function $S_x(\cdot)$, or equivalently the distribution function $F_x(\cdot)$, is completely determined by a set of $\mathcal{P}_1 \in \mathcal{P}$ variables, i.e. $S_x(\cdot) = S(\cdot|\mathbf{X}^{(j)} = x^{(j)}, j \in \mathcal{P}_1)$. We denote the set $\mathcal{P}_1$ as the set of strong variables and denote the set $\mathcal{P}_2 = \mathcal{P} \setminus \mathcal{P}_1$ as the set of noise variables.

4.3 Proposed Method and Algorithm

Follows the idea of reinforcement learning trees (Zhu et al., 2012), the proposed method fits an embedded model at each internal node when searching for splitting variable and associated cutting value. The embedded model should return the variable importance measure for each variable and the one with the largest variable importance is chosen for the split. Simultaneously, variable muting and protection processes are also implemented based on these variable importance measures and the resulting muted and protected set information is updated and passed along to the resulted two daughter nodes. Each daughter nodes repeat the same process until the number of observed failures is less than a pre-specified value. We begin this section by giving a pseudocode of the proposed method (Section 4.3.1). The details of each component are given in later subsections.

4.3.1 Reinforcement learning trees for right censored survival data

A high-level pseudocode is given in Table 4.1. We defer details of the embedded survival model (Section 4.3.2), variable importance measures (Section 4.3.3), and variable muting (Section 4.3.4) to later subsections.

Remark 4.3.1. *Bootstrapping plays an important role in random forests especially when calculating the variable importance. For the single tree fitted by a bootstrap sample, the corresponding out-of-bag data (those observations not sampled by bootstrapping) are*

Table 4.1: Reinforcement learning trees for right censored survival data

Step 1 Draw *ntrees* bootstrap samples from $\mathcal{D}_n$.

Step 2 For the m-th bootstrap sample, where $m \in \{1, ..., ntrees\}$, fit one RLT survival model, using the following rules:

a. At the root node, set both the muted set $\mathcal{P}^d = \varnothing$ and the protected set $\mathcal{P}^0 = \varnothing$, where $\varnothing$ denotes the empty set.

b. At each internal node A, including the root node, fit an embedded survival model (Section 4.3.2) using only the set of unmuted variables at the current node A, $\mathcal{P} \setminus \mathcal{P}^d_A$, and return variable importance measures $\widehat{VI}_A(j)$ (Section 4.3.3), for each variable $j \in \mathcal{P} \setminus \mathcal{P}^d_A$.

c. Split the current node using variable label $\arg\max_j \widehat{VI}_A(j)$, with the one with the largest variable importance. The splitting value c is generated uniformly from the current range of the splitting variable, however, with a restriction that each resulted daughter nodes contains at least $nmin/2$ observed failures.

d. Update the muted set and the protected set (Section 4.3.4) and pass along to both daughter nodes.

c. Stop a node when it contains no more than $nmin$ number of observed failures, and calculate the terminal node Kaplan-Meier survival function estimator.

Step 3 Average *ntrees* single tree RLT survival models to obtain the ensemble model fit.

independent from the fitted tree, hence the estimated prediction error is more reliable. However, bootstrapping is not a must-have for ensemble methods. Fitting each tree with the entire training data (Geurts et al., 2006) allows deeper and larger trees to be grown and can sometimes lead to better model fitting.

4.3.2 Embedded survival model

Why do we need an embedded model? In a traditional tree-based model, when searching for the splitting variable, only a marginal effect is evaluated in the form of an binary split $I(X < c)$. In the regression setting, weighted variance reduction is used, while in the classification setting, the Gini index is widely used (Breiman et al., 1984). However, evaluating the marginal effect using these criteria can sometimes be problematic. The checkerboard structure described in Biau et al. (2008) and Zhu et al. (2012) is one of these examples. Suppose in a regression mode, $X^{(1)}$ and $X^{(2)}$ are independent uniform random variables and the mean of response $E(Y) = I(I(X^{(1)} < 0.5) = I(X^{(2)} < 0.5))$. It is easy to verify that the variance reduction for splitting either $X^{(1)}$ or $X^{(2)}$ is always 0 regardless of the choice of the splitting point. Hence in a high-dimensional setting, it is less likely that either of them will be used as the splitting rule. In tree-based survival models, the same hidden structure can also jeopardize the accuracy of the fitted model. The commonly used logrank test statistics as used in Ishwaran et al. (2008) and Zhu and Kosorok (2012) can only assess marginal effects. In a Cox model, if the hazard ratio contains the aforementioned structure, none of the existing methods can efficiently detect it. An embedded model can be beneficial since the variable importance measure gives an overall evaluation of the contribution rather than only the marginal effect. Although the signal of $X^{(1)}$ and $X^{(2)}$ detected by the embedded model might be very weak, it is enough to push the entire model to further concentrate on these variables (for more intuition regarding the embedded model in the regression setting, please see Zhu et al. (2012)).

Now we introduce the embedded survival model in Table 4.2 , which is a simple extension of extremely randomized trees to right censored survival data. However, we defer the variable importance measure to Section 4.3.3. This model has much in common with Ishwaran et al. (2008), and can be viewed as a non-imputed version of Zhu and Kosorok (2012).

Remark 4.3.2. *Some tuning parameters are involved in both the embedded model, such as the terminal node size nmin, the number of variable sampled at each split, mtry, and the number of randomly generated splitting values, nsplit. Turning parameters can play a important role in the accuracy of tree-based methods, hence, in the simulation*

Table 4.2: Embedded survival model

Step 1	Draw *ntrees* bootstrap samples from $\mathcal{D}_n$.
Step 2	For each bootstrap sample, fit a survival tree model using the following rules:
a.	At each node, randomly select *mtry* variables from the set $\mathcal{P} = \{1, 2, ...p\}$
b.	For each randomly selected variable $X^{(j)}$, randomly generate *nsplit* random splitting value c within the range of $X^{(j)}$ in the current node. Calculate a logrank test statistic using the group label $I(X^{(j)} > c)$. Note that if there are less than $nmin/2$ observed failures in either group, that splitting rule is discarded.
c.	Compare the logrank statistics of all $mtry \times nsplit$ combinations of splitting variable and splitting value, and choose the one with the smallest p-value as the splitting rule in the current node.
d.	Split the current node into two daughter nodes according to the best splitting rule and repeat a) - d) for each daughter node until there are no more than *nmin* number of observed failures. At a terminal node, the Kaplan-Meier survival function estimator is recorded.
e.	Use corresponding out-of-bag data to calculate perdition error, and the perturbed prediction error for each variable (Section 4.3.3).
Step 3	Average all single tree survival models to obtain the ensemble model fit. Average all single tree prediction errors to obtain the variable importance measure (Section 4.3.3).

study, we try to compare different methods in several different tuning parameter settings. However, our proposed RLT survival model is less sensitive to these mentioned tuning parameters since the only information we obtain from an embedded model is the ranks of variable importance, which are not largely affected by the tuning parameters.

4.3.3 Variable importance for survival tree model

The variable importance measure is an important component of reinforcement learning trees because a within-node variable importance evaluation is the key reason why RLT performs better than random forests in the high-dimensional setting. Hence a counterpart for the variable importance measure in the survival setting must be carefully defined. In the regression model, one can simply use the increment of mean squared error caused by perturbing a variable as the variable importance. However, it is not possible to do the same in the survival setting due to censoring. Noticing that we try to obtain the prediction error by comparing a single observation (either observed or censored) with a survival function estimation, a natural choice here is the commonly used integrated Brier score (Graf et al., 1999; Hothorn et al., 2006; Zhu and Kosorok, 2012). Some other types of prediction errors are discussed in Remark 4.3.3 below.

The Brier score for censored data at a given time $t > 0$ is defined as

$$\begin{aligned} BS(t) &= \frac{1}{N}\sum_{i=1}^{N}\{(\hat{S}(t|\mathbf{X}_i))^2 I(Y_i \le t, \delta_i = 1)\hat{G}(Y_i)^{-1} \\ &\qquad +(1-\hat{S}(t|\mathbf{X}_i))^2 I(Y_i > t)\hat{G}(Y_i)^{-1}\}, \end{aligned} \tag{4.1}$$

where $\hat{G}(\cdot)$ denotes the Kaplan-Meier estimate of the censoring distribution. The integrated Brier score is further given by

$$IBS = max(Y_i)^{-1}\int_0^{max(Y_i)} BS(t)dt. \tag{4.2}$$

It is easy to see that when there is no censoring, the Brier score at time t measures the squared probabilistic error between a probabilistic prediction $S(t|X_i)$ and an observed event $I(T_i \le t)$. When censoring is present, the inverse probability weight $G(Y_i)^{-1}$ is introduced to compensate for the loss of information. New we define the variable importance based on Brier score for the tree survival model. Note that this definition applies to both the RLT survival model and its embedded model. For an embedded model, if a variable is muted at an internal node, automatically set its variable importance to 0 since that variable is not participated in the embedded model fitting.

Table 4.3: Variable importance

Step 1	For the m-th tree survival model and it's corresponding out-of-bag data, calculate the Brier score prediction errors:
a.	**Brier score**: Drop each out-of-bag data down the fitted tree and obtain the survival function estimate. Use Equation 4.2 to calculate integrated Brier score (IBS_m).
b.	**Permuted Brier score**: For each variable j, do the following:
i)	Randomly permute the values of the j^th variable $X^{(j)}$ in the out-of-bag data.
ii)	Drop permuted out-of-bag data down the fitted tree, and calculate the permuted integrated Brier score, $PIBS_m^j$.
Step 2	Average over all trees, and obtain the variable importance measure for variable j: $VI(j) = \frac{\sum_m PIBS_m^j}{\sum_m IBS_m} - 1.$

Remark 4.3.3. *Ishwaran et al. (2008) uses the concordance index (C-index, Harrell Jr et al. (1982)) as the prediction error to evaluate the overall fitting. The C-index measures the probability of concordance between predicted and observed responses, but will require a single prediction value such as the ensemble mortality used in their paper. However, this could resulting lost information since the survival function estimate is available. In Zhu and Kosorok (2012) the simulation study seems to prefer the Brier score as it is slightly more sensitive than the C-index. Other types of prediction errors such as the L_1 and L_∞ measures of the differences between the predicted and the true survival function are only available when the true data generator is know. We will give more details about these measurements (Section 4.4.3) and use them in the simulation study to compare different methods.*

4.3.4 Variable muting

Motivation: As a tree grows deeper, the sample size in the internal nodes decreases rapidly (at a rate of log_2, if we naively always split at a middle point). The search for a strong variable, with or without the embedded model, becomes increasingly difficult. The consequence is that when an internal node sample size is sufficiently small, the splitting rule behaves like a purely random search, which involves mostly noise variables. A muting procedure handles this situation nicely by preventing some variables from entering the model at deep internal nodes. We call this set of variables the muted set. At each internal node, we force p_d variables into the muted set, and we remove them from consideration as a splitting variable at any branch of the internal node. On the other hand, to prevent strong variables from being removed from the model, we set a maximum number of p_0 protected variables that we always keep in the model. Following Zhu et al. (2012), with some slight modifications, we introduce the muting process. Note that muting and protecting are only available after an embedded model is fit and the variable importance measure is returned, however, they are not available for the embedded model itself.

At an internal node A, suppose the current muted set is $\mathcal{P}_A^d$, the protected set is $\mathcal{P}_A^0$ (note that if A is the root node, we set both sets to the empty set $\emptyset$). After the splitting rule has been found through the embedded model, we denote the two daughter nodes as A_L and A_R. We first update the protected set for the two daughter nodes by adding the splitting variable(s) into the set (if the size of the current protected set is less than p_0):

$$\mathcal{P}_{A_L}^0 = \mathcal{P}_{A_R}^0 = \mathcal{P}_A^0 \cup \{\text{splitting variable at node}\, A\}, \quad \text{if}\, |\mathcal{P}_A^0| < p_0.$$

We then update the muted set for both daughter nodes. After sorting the variable importance measures $\widehat{VI}_A(j)$ returned from the embedded model, we find the p_d-th smallest value within the restricted variable set $\mathcal{P} \setminus \mathcal{P}_A^d \setminus \mathcal{P}_A^0$, i.e., in the set of variables that have not yet been muted nor protected. We denote this value as $\widehat{VI}_A^{p_d}$. Then we define the muted set for the two daughter nodes as

$$\mathcal{P}^d_{A_L} = \mathcal{P}^d_{A_R} = \mathcal{P}^d_A \cup \{j : \widehat{VI}_A(j) \le \widehat{VI}^{p_d}_A\} \setminus \mathcal{P}^0_A.$$

We then pass the updated muted and protected set to each daughter nodes and repeat the process for each of them.

Remark 4.3.4. *The number of muted variables at each split, p_d, can be set to a fixed number or it could vary depending on $|\mathcal{P}\backslash\mathcal{P}^d_A|$, which is the number of unmuted variables at the current internal node A. In Section 4.4 we will evaluate different choices for p_d such as 0 (no muting), $20\% \cdot |\mathcal{P} \setminus \mathcal{P}^d_A|$ (moderate muting), and $50\% \cdot |\mathcal{P} \setminus \mathcal{P}^d_A|$ (aggressive muting).*

Remark 4.3.5. *The protected set updating process is slightly different from Zhu et al. (2012). In their paper, the protected set is updated to full size of p_0 when the first split is done. However, in this paper, the protected set is progressively grown to p_0 by only adding the splitting variable to the protected set. This new approach makes the tuning of p_0 to have less of an impact on the model fitting. The large sample behavior under the regression setting should not be affected by this change since the protected set should contain all strong variables after finitely many splits (Theorem3.4.7).*

4.4 Simulation Studies

In this section, we use simulated data to demonstrate the performance of the proposed method and compare to competing methods such as random survival forests (Ishwaran et al., 2008) though the R package "randomForestSRC", and the LASSO for the cox model through the R package "glmnet" (Friedman et al., 2010). We also include the embedded model (the simple extension from extremely randomized trees to survival data presented Section 4.3.2, which is in fact a survival tree model without reinforcement learning) into the comparison and observe the solo effect of reinforcement learning. Note that some early single tree models are omitted in this comparison due to poor accuracy, and the survival ensembles method (Hothorn et al., 2006) is also omitted due to the fact that a key assumption, $P(T > C|\mathbf{X}) > 0$, is oftentimes violated or nearly violated.

It is known that tree-based models can be sometimes sensitive to the choice of tuning parameters. Hence in our simulation study, we compare different tree-based methods under a variety of tuning parameter settings, including *mtry*, *nmin*, bootstrapping percentage, and the splitting value generating methods. However, for the proposed method, only the default setting is used because these turning parameters do not seem to impact the variable importance measure of the embedded model, hence are meaningless to tune. The only turning parameter that we are experimenting with is the muting parameter p_d. Details of parameter settings are described in Section 4.4.2.

4.4.1 Data generator

In this simulation study, we include two typical scenarios and their variations on sample size and total dimension p. Let $X = (X^{(1)}, ..., X^{(p)})$ be drawn from a multivariate normal distribution with covariance matrix V, where $V_{ij} = \rho^{|i-j|}$, where ρ and p will be specified in the following. The first scenario is a proportional hazard model with linear link function, which will clearly favor the LASSO. The second scenario has no linear effect and will favor tree-based methods. For each setting, we use 500 test observations to calculate prediction error. Each simulation is run 100 times.

Scenario 1: A proportional hazard model with conditional independent censoring.

Let $\rho = 0.75$ and $p = 200$. The failure time T follows an exponential distribution with mean $exp(X^{(10)} + X^{(30)} + X^{(50)})$. The censoring time C follows an exponential distribution with mean $5 \times exp(X^{(20)} + X^{(40)})$. Observations exceeding 4 are forced to be censored to prevent the tail region from dominating the prediction error. The censoring rate is approximately 33%.

Scenario 1: A weibull distribution with symmetric link function. Let $\rho = 0.5$ and $P = 100$. The failure time T follows an Weibull distribution with shape parameter 2 and scale parameter $2 \times \Phi(X^{(10)} \times X^{(30)} + (X^{(50)})^2 - 1)$, where Φ is the standard normal $c.d.f$ function. The censoring time C has probability 1/3 to be 2 and probability 2/3 to be uniform (0, 2). The Censoring rate is approximately 28%.

4.4.2 Tuning parameters

For all tree-based models, we only consider bootstrapped version (Breiman, 2001) of ensembles, although a non-bootstrapped version (Geurts et al., 2006) can also be used in practice. The bootstrap sample size is set to 67% of the training sample size. We set the number of trees to be 200 for random survival forests (RSF) and the simple survival model. For the proposed RLT survival model, number of trees is set to 50. For RSF and simple survival tree, we consider all combinations of the following parameter settings: $mtry = \sqrt{p}$ or $p/3$; $nmin = 4$ or 10; the splitting value is searched to find the best or by comparing 10 random splits. In the proposed method, we only use a default setting of $nmin = 4$ and splitting value is chosen by comparing 10 random splits. Note that $mtry$ is no longer a tuning parameter in RLT since we always select the variable with the largest variable importance to split, however it can be a tuning parameter for the embedded model. We consider three muting procedures for the proposed method: no muting ($p_d = 0$), moderate muting ($p_d = 20\% \cdot |\mathcal{P} \setminus \mathcal{P}_A^d|$ at any node A), and aggressive muting ($p_d = 50\% \cdot |\mathcal{P} \setminus \mathcal{P}_A^d|$ at any node A). In glmnet, a simple lasso penalty is used, and λ is set to "*lambda.min*" or "*lambda.*1*se*" from 10 fold cross validation.

4.4.3 Prediction error

To compare with the true survival function, we use the integrated absolute error and supremum absolute error which can be viewed as the L_1 and L_∞ norm of the survival function error. To be more specific, let $S(t)$ denote the true survival function and let $\hat{S}(t)$ denote its estimate. Integrated absolute error is defined as $\frac{1}{max(Y_i)} \int_0^{max(Y_i)} |S(t) - \hat{S}(t)|dt$ and supremum absolute error is defined as $\sup\limits_{0<t\leq max(Y_i)} |S(t) - \hat{S}(t)|$. We also use the integrated Brier score (Equation 4.2) and C-index (Remark 4.3.3) in the comparison.

4.4.4 Simulation results

Among all tree-based methods, RLT survival model performs best in terms of almost all prediction errors. glmnet outperforms all tree-based method in Scenario 1, however, performs the worst in Scenario 1 when no monotone effect is involved in the true model structure. We summarize some of the key finding in the following:

- Different error measurements give almost the same conclusion when comparing RLT survival model with competing methods. $L1$ error seems to be more sensitive then other types of error measurements. In Scenario 1, all measurements favors glmnet as expected since cox model is the true model. Among tree-based models, RLT with aggressive muting performs best. It reduces $L1$ error by 13.9% from RSF and by 10.4% from simple survival tree model when $N = 200$. The differences in terms of other measurements are less significant. RLT shows greater advantage under complicated model structure. In Scenario 2, RLT with aggressive muting reduces prediction errors by 28.6%, 17.8%, 19.9% and 41.4% for $L1$, L_∞, IBS and C-index respectively.

Table 4.4: Prediction Errors: Scenario 1 (Cox model)

	$N = 200$				$N = 300$			
	L_1	L_∞	IBS	C-index	L_1	L_∞	IBS	C-index
RLT survival								
no muting	0.161	0.311	0.162	0.248	0.147	0.292	0.158	0.240
20% muting	0.159	0.308	0.161	0.247	0.143	0.286	0.156	0.235
50% muting	0.155	0.304	0.159	0.242	0.137	0.279	0.154	0.231
Competing methods								
simple survival trees	0.173	0.315	0.165	0.251	0.158	0.294	0.159	0.237
RSF	0.180	0.321	0.168	0.250	0.166	0.301	0.162	0.236
glmnet - cox	0.059	0.149	0.119	0.186	0.049	0.127	0.118	0.185

Table 4.5: Prediction Errors: Scenario 2 (symmetric and checker-board effects)

	$N = 300$				$N = 400$			
	L_1	L_∞	IBS	C-index	L_1	L_∞	IBS	C-index
RLT survival								
no muting	0.233	0.456	0.187	0.332	0.209	0.434	0.172	0.280
20% muting	0.225	0.446	0.182	0.305	0.194	0.411	0.163	0.245
50% muting	0.216	0.432	0.176	0.279	0.182	0.393	0.157	0.228
Competing methods								
simple survival trees	0.262	0.480	0.202	0.410	0.255	0.478	0.196	0.389
RSF	0.266	0.483	0.204	0.418	0.258	0.480	0.197	0.390
glmnet - cox	0.279	0.485	0.212	0.500	0.278	0.489	0.209	0.500

- Muting improves performance of RLT survival model. The $L1$ error reduction from no muting to aggressive muting ranges within 3.7% to 12.9%. In Scenario 2, aggressive muting reduces C-index error by 16.0% and 18.6%.

- RLT survival model benefits from increased sample size better than competing methods. In Scenario 2, L_1 error of RLT model decreased from 0.233, 0.225 and 0.216 to 0.209, 0.194 and 0.182 (corresponding to 10.3%, 13.8% and 15.7% reduction) when sample size is increased from 300 to 400. As contrast, the best improvement for competing methods is 3.0% (from 0.266 to 0.258).

4.5 Discussion

In this paper, we proposed a new tree-based survival model to extend the reinforcement learning trees to right censored survival data. The proposed method takes advantages of two key components in reinforcement learning trees and results in improved accuracy especially in high-dimensional sparse settings. The embedded model evaluates the contribution of each variable at the current internal node and forces splits on the most "need-to-be-split" variable. This procedure can clearly tackle down some difficult model structures such as the checker-board model (Biau et al., 2008) as we have seen in the simulation study. The variable muting process further improves the model by gradually removing noise variables from the model. In this section, we discuss some issues that we have not previously addressed in this paper.

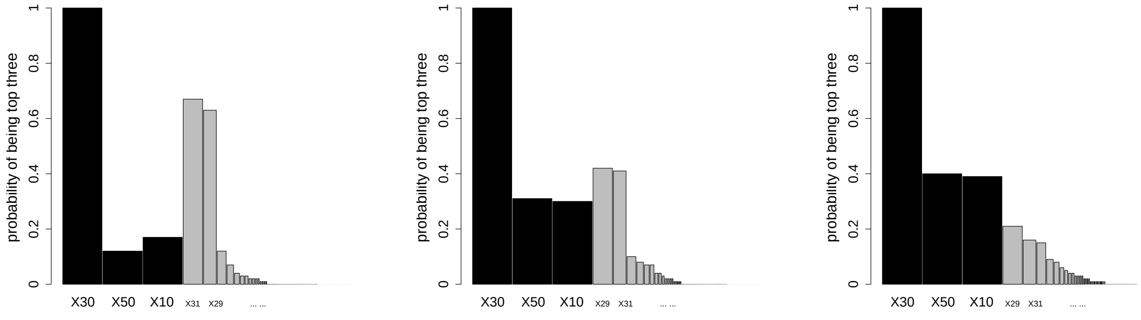
4.5.1 linear combination split

A linear combination split was proposed in Zhu et al. (2012), however, it is not implemented in our survival model. The linear combination split creates a splitting rule such as $I(\boldsymbol{\beta}'\mathbf{X} < c)$ at each internal node, where $\boldsymbol{\beta}$ is a vector of p components that defines a projection line in $p-$dimensional space. The absolute values of $\boldsymbol{\beta}$ can be practically determined using the variable importance measures to emphasize on strong variables. However, the sign of each component of $\boldsymbol{\beta}$ can be difficult to identify. Ideally, we want the variables with similar influence on the hazard function to be given the same sign, however, this can be hard to evaluate since the hazard function is changing over time. In a situation where the survival functions of two groups cross each other, creating a linear combination split might lead to a poor splitting rule. Hence, this linear combination split is not used in our model.

4.5.2 Variable importance measures of correlated variables

Correlated variables, especially those who are highly correlated with strong variables, always cause trouble in a statistical model. In tree-based models, due to the random feature selection (the tuning parameter $mtry$), some correlated variables rather than the true strong variable are often used in the splitting rule. This leads to a situation where these correlated variables may have an over-estimated variable importance measure. Some efforts have been made to correct this bias such as the conditional variable importance proposed by Strobl et al. (2008). Here, we reveal some facts that suggests reinforcement learning trees may help to reduce the bias. Consider Scenario 1 of the simulation study with $p = 100$, $\rho = 0.9$, and the mean of failure time T equals $exp(X^{(10)} + 1.5 \times X^{(30)} + X^{(50)})$. With highly correlated covariances, $X^{(29)}$ and $X^{(31)}$ usually has larger variable importance measure than $X^{(10)}$ and $X^{(50)}$, especially when $mtry$ is small. In Figure 4.1, we plot the probability of being the top three largest variable importance measures for three different measures: RSF with $mtry = P$ or $mtry = P/3$, and the proposed RLT survival model.

Figure 4.1: Comparing variable importance



RSF $mtry = p/3$ RSF $mtry = p$ RLT survival

Black: Strong variables; Gray: Noise variables

The reason behind this apparent effect for using reinforcement learning trees can be explained by the reduced chance of using correlated variables as splitting rules, which is driven by two facts. First, a strong variable has an overall larger variable importance measure than variables correlated with it. Hence, when RLT selects a splitting variable, it will most likely be a strong variable. Second, conditional on a sub-interval of a strong variable, its variable importance should still be larger than the correlated variables. Then during the entire tree construction, the RLT will tend to avoid using these correlated variables as splitting rules, and their variable importance measures will therefore tend to be small.

4.5.3 Computational issues

A R (version 3.0.1) implementation of the proposed method (including reinforcement learning trees for classification and regression) is available through author's personal website: `http://www.bios.unc.edu/~rzhu`. The algorithm is written in C, and also incorporated parallel computing using OpenMP. The computation time of a simple survival tree model is in par with existing R packages such as "randomForest" and "randomForestSRC". The proposed method (which will run simple survival tree model at each node as the embedded model) usually takes within 100 times computational time of a simple model.

Chapter 5

Conclusion and future research plan

Three methods are proposed in my dissertation: recursively imputed survival trees, reinforcement learning trees and its extension to survival data. A theoretical framework is proposed for reinforcement learning trees and showed consistency and convergence rate. This new framework is the first in its kind and can be applied to any extension of reinforcement learning trees. There are some subsequent works that I would like to explore/finish in the near future:

- The R package for reinforcement learning trees is available to public and the computational speed seems to be competitive to existing popular tree-based method packages. Some extra features need to be completed such as missing data imputation and tree plots.

- A common data structure for survival analysis contains time-varying explanatory variables, such as transplant status, or smoking status. None of the existing tree-based ensemble methods can handle this type of data. It is interesting to extend reinforcement learning trees to this data structure.

- Reinforcement learning trees can also be applied to single tree model, which is easier to interpret and visualize, and still enjoy many advantages of reinforcement learning.

- A promising application of reinforcement learning trees is on personalized medicine where high-dimensional genetic data are involved and complicated interaction model structure is needed.

Appendix

Asymptotic Results

Proof of Theorem 3.4.7.

Step 1: We first establish the asymptotic results for the variable importance measure. Without further specification, the proof of Step 1 is conditional on an internal node A with sample size n_A and number of non-muted variables equal to p_A. We denote the internal node dataset by $\mathcal{D}_A = \{(X_i, Y_i), i \in A\}$. Let $\mathbb{P}$ be the probability measure of $((X), Y)$ and let $\mathbb{P}$ be the corresponding empirical measure.

First, we observe that, $VI_A(j)$ is bounded. By Assumption 3.4.1, f is Lipschitz continuous with Lipschitz constant c_f,

$$\begin{aligned} & VI_A(j) \\ = \quad & \frac{E[E[(f(X_i^{(1)}, ..., \tilde{X}_i^{(j)}, ..., X_i^{(p)}) - f(X_i^{(1)}, ..., X_i^{(j)}, ..., X_i^{(p)}))^2 | X_i^{(j)}]|A]}{\sigma^2} \\ \leq \quad & \frac{E[E[(c_f \cdot (b_j - a_j))^2 | X_i^{(j)}]|A]}{\sigma^2} = \frac{c_f^2 \cdot (b_j - a_j)^2}{\sigma^2}. \end{aligned} \tag{5.1}$$

Hence $VI_A(j)$ is also bounded above by the interval length of $X^{(j)}$, i.e. $(b_j - a_j)$, in A. It can be further bounded above by $\frac{c_f^2}{\sigma^2}$ since $(b_j - a_j) < 1$ for any internal node A.

Now we show that $\widehat{VI}_A(j)$ converges to $VI_A(j)$ at an exponential rate. For simplicity, assume that the embedded model $\widehat{f}_A^*$ randomly selects half of $\mathcal{D}_A$ to fit the model, denoted by $\mathcal{D}_{A_1}$, and the variable importance is calculated using the other half of the data, denoted by $\mathcal{D}_{A_2}$. Noticing that this is exactly (except for the proportion of each subset) what we do for each tree in a standard random forests model. However, with the potential use of other models, this simplifies the formulation. Further, since the j-th variable importance measure is calculated by randomly permuting the values of $X_i^{(j)}$ in $\mathcal{D}_{A_2}$, which we denote by $\tilde{X}_i^{(j)}$, we assume that this permutation is done infinitely many times. Then, for the i-th observation in $\mathcal{D}_{A_2}$, the squared error after permutation is $E_{\tilde{X}_i^{(j)}}\big(\widehat{f}_A^*(X_i^{(1)}, ..., \tilde{X}_i^{(j)}, ..., X_i^{(p)}) - Y_i\big)^2$. Hence the j-th variable importance can be formulated as:

$$\begin{aligned} &\widehat{VI}_A(j) \\ &= \frac{\frac{1}{n_A/2}\sum_{\mathbf{X}_i\in\mathcal{D}_{A_2}} E_{\tilde{X}^{(j)}}\Big(\widehat{f}_A^*(X_i^{(1)}, ..., \tilde{X}^{(j)}, ..., X_i^{(p)}) - Y_i\Big)^2}{\frac{1}{n_A/2}\sum_{\mathbf{X}_i\in\mathcal{D}_{A_2}} E_{\tilde{X}^{(j)}}\Big(\widehat{f}_A^*(X_i^{(1)}, ..., X_i^{(j)}, ..., X_i^{(p)}) - Y_i\Big)^2} - 1 \\ &= \frac{\frac{1}{n}\sum_{\mathbf{X}_i\in\mathcal{D}} E_{\tilde{X}^{(j)}}\Big(\widehat{f}_A^*(X_i^{(1)}, ..., \tilde{X}^{(j)}, ..., X_i^{(p)}) - Y_i\Big)^2 I_{[\mathbf{X}_i\in A_2]}}{\frac{1}{n}\sum_{\mathbf{X}_i\in\mathcal{D}} E_{\tilde{X}^{(j)}}\Big(\widehat{f}_A^*(X_i^{(1)}, ..., X_i^{(j)}, ..., X_i^{(p)}) - Y_i\Big)^2 I_{[\mathbf{X}_i\in A_2]}} - 1, \end{aligned} \tag{5.2}$$

where $I[\mathbf{X}_i \in A_2]$ is the indicator function denoting that $\mathbf{X}_i$ falls into the internal node A, and is randomized with probability $\frac{1}{2}$ to $\mathcal{D}_{A_2}$ for calculating variable importance. Let the set $(X_i^{(1)}, ..., X_i^{(j-1)}, X_i^{(j+1)}..., X_i^{(p)})$ be $X_i^{(-j)}$. Then the numerator of the first term of (5.2) can be broken down into:

$$\frac{1}{n}\sum_{\mathbf{X}_i\in\mathcal{D}} E_{\tilde{X}^{(j)}}\Big(\widehat{f}_A^*(X_i^{(1)}, ..., \tilde{X}^{(j)}, ..., X_i^{(p)}) - Y_i\Big)^2 I_{[\mathbf{X}_i\in A_2]}$$

$$
\begin{aligned}
&= \mathbb{P}_n\Big(E_{\tilde{X}^{(j)}}\big(\widehat{f}^*_A(X^{(-j)},\tilde{X}^{(j)})-Y\big)^2 I_{[\mathbf{X}\in A_2]}\Big)\\
&= (\mathbb{P}_n-\mathbb{P})\Big(E_{\tilde{X}^{(j)}}(\widehat{f}^*_A(X^{(-j)},\tilde{X}^{(j)})-Y)^2 I_{[\mathbf{X}\in A_2]}\Big)\\
&\quad +\mathbb{P}\Big(E_{\tilde{X}^{(j)}}\big(\widehat{f}^*_A(X^{(-j)},\tilde{X}^{(j)})-f_A(X^{(-j)},\tilde{X}^{(j)})\big)^2 I_{[\mathbf{X}\in A_2]}\Big)\\
&\quad +\mathbb{P}\Big(E_{\tilde{X}^{(j)}}\big(f_A(X^{(-j)},\tilde{X}^{(j)})-f_A(X^{(-j)},X^{(j)})\big)^2 I_{[\mathbf{X}\in A_2]}\Big)\\
&\quad +\mathbb{P}\Big(E_{\tilde{X}^{(j)}}\big(f_A(X^{(-j)},X^{(j)})-Y\big)^2 I_{[\mathbf{X}\in A_2]}\Big)\\
&=: \tilde{T}_1+\tilde{T}_2+\tilde{T}_3+\tilde{T}_4. \qquad (5.3)
\end{aligned}
$$

Now we bound each of the four terms in Equation 5.3. We will first show the bound for $\tilde{T}_1$ and then for $\tilde{T}_2^*$, following the same idea. We use Theorem 8 in van de Geer and Lederer (2012) to establish the bound for $\tilde{T}_1$. The Theorem states that for any function $g(\mathbf{X})$ that lives in a collection of functions $\mathcal{G}$, if the Bernstein condition

$$
\sup_{g\in\mathcal{G}} E|g|^m \le \frac{m!}{2}K^{m-2},\ m=2,3,... \qquad (5.4)
$$

is satisfied for some constant $K\ge 1$, then $\sqrt{n}(\mathbb{P}_n-\mathbb{P})g$ has exponential tail.

By Assumption 3.4.6, $\widehat{f}^*$ has exponential tail. On the other hand, $Y=f(\mathbf{X})+\epsilon$, and $f(\mathbf{X})$ are bounded, and hence Y also satisfies the moment condition by Assumption 3.4.5. Hence, we can fine some constant K such that the following Bernstein condition is satisfied:

$$
\sup_{\widehat{f}^*} E\Big|f^*_A(X^{(-j)},\tilde{X}^{(j)})-Y\Big|^m \le \frac{m!}{2}K^{m-2},\ m=2,3,.... \qquad (5.5)
$$

Furthermore, since $\widehat{f}^*$ has finite entropy integral by Assumption 3.4.6, we can use Theorem 8 in van de Geer and Lederer (2012) and reorganize the terms to can find a constant $K_1^* > 0$ such that:

$$P\left(\sup \left| \sqrt{n}\tilde{T}_1 \right| \geq t \right) \leq e^{-t/K_1^*}. \tag{5.6}$$

For $\tilde{T}_2$, we first write it into a conditional probability $\mathbb{P}_{A_2}$ such that

$$\begin{aligned} \tilde{T}_2 &= \mathbb{P}_{A_2}\Big(E_{\tilde{X}^{(j)}} \big(\widehat{f}_A^*(X^{(-j)}, \tilde{X}^{(j)}) - f_A(X^{(-j)}, \tilde{X}^{(j)}) \big)^2 \Big) P(A_2) \\ &= \tilde{T}_2^* P(A_2). \end{aligned} \tag{5.7}$$

For $\tilde{T}_2^*$, noting Assumption 3.4.6 for the error bound of f_A^*, and following similar arguments as applied to $\tilde{T}_1$, we have for some constant $K_2^* > 0$:

$$P\left(\sup \left| \sqrt{n_A^{\eta(p_A)}} \tilde{T}_2^* \right| \geq t \right) \leq e^{-t/K_2^*}. \tag{5.8}$$

For the other two terms, it is easy to see by Definition 3.3.1 that $\tilde{T}_3 = VI_A(j)$ $\sigma^2 P(A_2)$, and $\tilde{T}_4 = \sigma^2 P(A_2)$ by Assumption 3.4.5.

Note that the denominator of the first term in (5.2) can be decomposed into four terms: T_1, T_2, T_3^* and T_4, similar to (5.3) but with $X_i^{(j)}$ in the lieu of $\tilde{X}_i^{(j)}$. The first two terms can be bounded in the same way as the above. The third term equals 0 since $\tilde{X}_i^{(j)}$ is replaced by $X_i^{(j)}$. And the fourth term $T_4 = \sigma^2 P(A_2)$.

Hence, together with (5.6), (5.8) for the numerator, and the above arguments for the denominator, we can derive that

$$
\begin{aligned}
& P\Big(\Big|\widehat{VI}_A(j) - VI_A(j)\Big| > C\Big) \\
= \;& P\Big(\Big|\frac{\tilde{T}_1 + \tilde{T}_2^* P(A_2) + \sigma^2 P(A_2) VI_A(j) + \sigma^2 P(A_2)}{T_1 + T_2^* P(A_2) + 0 + \sigma^2 P(A_2)} - 1 - VI_A(j)\Big| > C\Big) \\
\leq \;& P\Big(\Big|\frac{\tilde{T}_1}{T_1 + T_2^* P(A_2) + \sigma^2 P(A_2)}\Big| > C/3\Big) \\
& + P\Big(\Big|\frac{\tilde{T}_2^* P(A_2)}{T_1 + T_2^* P(A_2) + \sigma^2 P(A_2)}\Big| > C/3\Big) \\
& + P\Big(\Big|\frac{\sigma^2 P(A_2)(VI_A(j) + 1)}{T_1 + T_2^* P(A_2) + \sigma^2 P(A_2)} - 1 - VI_A(j)\Big| > C/3\Big) \\
= \;& P\Big(\Big|\frac{\tilde{T}_1}{T_1 + T_2^* P(A_2) + \sigma^2 P(A_2)}\Big| > C/3\Big) \\
& + P\Big(\Big|\frac{\tilde{T}_2^* P(A_2)}{T_1 + T_2^* P(A_2) + \sigma^2 P(A_2)}\Big| > C/3\Big) \\
& + P\Big(\Big|\frac{(T_1 + T_2^* P(A_2))(1 + VI_A(j))}{T_1 + T_2^* P(A_2) + \sigma^2 P(A_2)}\Big| > C/3\Big).
\end{aligned}
\tag{5.9}
$$

Noticing that all the T terms are positive, and $VI_A(j)$ is also positive and bounded above, we have:

$$
\begin{aligned}
& P\Big(\Big|\widehat{VI}_A(j) - VI_A(j)\Big| > C\Big) \\
\leq \;& P\Big(\Big|\frac{\tilde{T}_1}{\sigma^2 P(A_2)}\Big| > C/3\Big) + P\Big(\Big|\frac{\tilde{T}_2^* P(A_2)}{\sigma^2 P(A_2)}\Big| > C/3\Big) + \\
& P\Big(\Big|\frac{T_1(1 + VI_A(j))}{\sigma^2 P(A_2)}\Big| > C/6\Big) + P\Big(\Big|\frac{T_2^* P(A_2)(1 + VI_A(j))}{\sigma^2 P(A_2)}\Big| > C/6\Big) \\
\leq \;& e^{-C \cdot P(A_2) \cdot n/3K_1} + e^{-C \cdot n_A^{\eta(p_A)}/3K_2} + e^{-C \cdot P(A_2) \cdot n/3K_3} + e^{-C \cdot n_A^{\eta(p_A)}/3K_4} \\
\leq \;& e^{-C \cdot n_A^{\eta(p_A)}/K_5}.
\end{aligned}
\tag{5.10}
$$

Noting that this is the tail probability for $\widehat{VI}_A(j)$ when p_A variables are considered in the embedded model, we can easily generalize it to the situation at an internal node where only p_0 variables are considered. In this case, we replace $\eta(p)$ by $\eta(p_0)$, yielding a faster convergence rate. In the derivation, the constant K_5 can possibly depend on p_A, however, since $p_A < p$, which is finite, we can always choose a larger K_5 such that the equation holds for all values of p_A. Consequently, K_5 does not depend on the choice of internal node A.

Now, two situations can arise for $VI_A(j)$:

Situation 1: $X^{(j)}$ is a noise variable. Since changing the value of $X^{(j)}$ will not change $f(X)$, $f(X^{(1)}, ..., \tilde{X}^{(j)}, ..., X^{(p)}) \equiv f(X^{(1)}, ..., X^{(j)}, ..., X^{(p)})$, and thus $VI_A(j) \equiv 0$.

Situation 2: $X^{(j)}$ is a strong variable. According to Assumption 3.4.3, $VI_A(j)$ is bounded below by $\psi_1(\delta) \cdot \psi_2(b_j - a_j)$, where $\delta = \min_{i \in \{\mathcal{S} \setminus j\}} (b_i - a_i)$. We further note that since the internal node size is n_A, the interval length of any variable is at least $\frac{n_A}{n}$ even if all splits are made on that variable. Hence both δ and $b_j - a_j$ are larger than $\frac{n_A}{n}$. Hence $VI_A(j) \geq \psi_1(\frac{n_A}{n}) \cdot \psi_2(\frac{n_A}{n})$ for any strong variable.

Hence, to sum up situations (1) and (2), we have

$$VI_A(j) \begin{cases} \geq \psi_1(\frac{n_A}{n}) \cdot \psi_2(\frac{n_A}{n}), & \text{if } j \in \mathcal{S}. \\ = 0, & \text{if } j \in \mathcal{S}^c. \end{cases} \tag{5.11}$$

Step 2: Now we prove a) of this Theorem. Let $\widehat{j}_A$ be the selected splitting variable at internal node A, i.e. $\widehat{j}_A = \arg\max_j VI_A(j)$. Without loss of generality, we assume that at this internal node A, the true variable importance measures are in the order

$VI_A(1) \geq VI_A(2) \geq \cdots \geq VI_A(p_1) > VI_A(p_1+1) = \cdots = VI_A(p) = 0$. Then the probability that the selected splitting variable $\widehat{j}_A^*$ belongs to the set of strong variables satisfies the following inequality:

$$\begin{aligned}
&P(\widehat{j}_A \in \mathcal{S}) \\
&\quad = \; 1 - P(\widehat{j}_A \in \mathcal{S}^c) \\
&\quad = \; 1 - \sum_{i \in \mathcal{S}^c} P(\widehat{j}_A = i) \\
&\quad \geq \; 1 - \sum_{i \in \mathcal{S}^c} P\big(\widehat{VI}_A(i) > \widehat{VI}_A(j), \text{ for all } \; j \in \mathcal{S}\big) \\
&\quad \geq \; 1 - p_1 \sum_{i \in \mathcal{S}^c} P(\widehat{VI}_A(i) > \widehat{VI}_A(p_1)). \qquad (5.12)
\end{aligned}$$

Let $\widehat{\Delta}_j = \widehat{VI}_A(j) - VI_A(j)$. Using equation (5.10) and noting that $VI_A(i) = 0$ for all $i \in \mathcal{S}^c$, the above probability can be bounded below by

$$\begin{aligned}
&P(\widehat{j}_A \in \mathcal{S}) \\
&\quad \geq \; 1 - p_1 \sum_{i \in \mathcal{S}^c} P\big(\widehat{\Delta}_j + 0 > \widehat{\Delta}_{p_1} + VI_A(p_1)\big) \\
&\quad \geq \; 1 - p_1 \sum_{i \in \mathcal{S}^c} \left[P\big(|\widehat{\Delta}_{p_1}| > \frac{VI_A(p_1)}{2}) + P(\widehat{\Delta}_j > \frac{VI_A(p_1)}{2}) \right] \\
&\quad = \; 1 - p_1 \sum_{i \in \mathcal{S}^c} 4 \cdot e^{-\frac{VI_A(p_1)}{2} \cdot n_A^{\eta} / K_5} \\
&\quad = \; 1 - 4 p_1 p_2 \cdot e^{-\frac{VI_A(p_1)}{2} \cdot n_A^{\eta} / K_5}. \qquad (5.13)
\end{aligned}$$

Using Equation 5.11, we have, for any internal node A with sample size n_A, and with p_A nonmuted variables,

$$P(\widehat{j}_A \in \mathcal{S}) \geq 1 - 4p_1p_2 \cdot e^{-\psi_1(\frac{n_A}{n})\cdot\psi_2(\frac{n_A}{n})\cdot n_A^{\eta(p_A)}/(K_5\cdot 2)}. \tag{5.14}$$

Since p_1, p_2 and K_5 are all constant, the proof for a) is concluded.

Step 3: We show b) using a similar structure as the proof of a). Note that at any internal node A, the probability that the maximum true variable importance is larger than twice that of the selected splitting variable is

$$P\Big(\max_j VI_A(j) > 2VI_A(\widehat{j}_A)\Big).$$

By defining the variable with the true maximum variable importance at node A as $j_A^m = \arg\max_j VI_A(j)$, the above equation can be bounded by

$$\begin{aligned}
&P\big(VI_A(j_A^m) > 2VI_A(\widehat{j}_A)\big)\\
&\quad\leq\ P\left(VI_A(j_A^m) > VI_A(\widehat{j}_A) + \psi_1(\frac{n_A}{n})\cdot\psi_2(\frac{n_A}{n})\right)\\
&\quad=\ P\left(VI_A(j_A^m) - \widehat{VI}_A(j_A^m) > VI_A(\widehat{j}_A) - \widehat{VI}_A(j_A^m) + \psi_1(\frac{n_A}{n})\cdot\psi_2(\frac{n_A}{n})\right)\\
&\quad=\ P\Big(VI_A(j_A^m) - \widehat{VI}_A(j_A^m) > VI_A(\widehat{j}_A) - \widehat{VI}_A(\widehat{j}_A)\\
&\qquad\qquad\qquad + \widehat{VI}_A(\widehat{j}_A) - \widehat{VI}_A(j_A^m) + \psi_1(\frac{n_A}{n})\cdot\psi_2(\frac{n_A}{n})\Big).
\end{aligned}$$

Note that $\widehat{VI}_A(\widehat{j}_A) - \widehat{VI}_A(j_A^m) \geq 0$ since $\widehat{j}_A$ is the selected variable. Adapting the notation of $\widehat{\Delta}$ used in Step 2, we now have

$$\begin{aligned}
& P\Big(VI_A(j_A^m) > 2VI_A(\widehat{j}_A)\Big) \\
\leq\ & P\Big(\widehat{\Delta}_{j_A^m} > \widehat{\Delta}_{\widehat{j}_A} + 0 + \psi_1(\frac{n_A}{n}) \cdot \psi_2(\frac{n_A}{n})\Big) \\
\leq\ & P\Big(|\widehat{\Delta}_{j_A^m}| > \frac{\psi_1(\frac{n_A}{n}) \cdot \psi_2(\frac{n_A}{n})}{2}\Big) \\
& + P\Big(|\widehat{\Delta}_{\widehat{j}_A}| > \frac{\psi_1(\frac{n_A}{n}) \cdot \psi_2(\frac{n_A}{n})}{2}\Big) \\
\leq\ & 4e^{-\psi_1(\frac{n_A}{n})\cdot\psi_2(\frac{n_A}{n})\cdot n_A^{\eta(p_A)}/(K_5\cdot 2)}. \qquad (5.15)
\end{aligned}$$

Thus the proof for b) is concluded.

Step 4: We now show c), that the protected set $\mathcal{P}_A^0$ for the entire tree contains all strong variables with probability close to 1, provided the number of protect variables p_0 is greater than p_1. It is sufficient to show this property at the root node, where $A = [0,1]^p$, since the protected set will only increase after a split. Note that when $p_0 > p_1$, if a strong variable is not in the protected set, there must be at least one noise variable with larger $\widehat{VI}$. Hence we have:

$$\begin{aligned}
& P(\mathcal{S} \in \mathcal{P}_A^0) \\
\geq\ & 1 - P(\exists j \in \mathcal{S} \text{ and } i \in \mathcal{S}^c, s.t. \widehat{VI}_A(j) < \widehat{VI}_A(i)) \\
\geq\ & 1 - \sum_{j\in\mathcal{S}, i\in\mathcal{S}^c} P(\widehat{VI}_A(j) < \widehat{VI}_A(i)) \\
\geq\ & 1 - p_1 p_2 P(\widehat{VI}_A(p_1) < \widehat{VI}_A(p_1+1)).
\end{aligned}$$

By similar arguments to those used in Steps 2), and noting that $n_A = n$ at the root node, we can bound the above probability by:

$$\begin{aligned} & P(\mathcal{S} \in \mathcal{P}_A^0) \\ \geq & \ 1 - p_1 p_2 e^{VI_A(p_1) \cdot n^{\eta(p)}/(K_5 \cdot 2)}. \end{aligned} \tag{5.16}$$

Since at the root node, all the variable importance measures, including $VI_A(p_1)$, are fixed constants, The proof for c) is concluded.

□

Proof of Theorem 3.4.8. We prove this theorem in two steps. First, we show that for the entire constructed RLT, with exponential rate, only strong variables are used as splitting variables. Second, we derive consistency and error bounds by bounding the total variation using the terminal node size variable importance which converges to zero.

Step 1: In this step, we show that for the entire tree, only strong variables are used as the splitting variable, and furthermore, the variable importance measure for the splitting variable is at least half of the maximum variable importance at each split. First, it is easy to verify that, both a) and b) in Theorem3.4.7 can be satisfied simultaneously with probability bounded below by

$$1 - C \cdot e^{-\psi_1(\frac{n_A}{n}) \cdot \psi_2(\frac{n_A}{n}) \cdot n_A^{\eta(p)}/K}. \tag{5.17}$$

Define $\mathcal{A}$ as the set of all internal nodes. Recall that $\psi_1(\delta)$ and $\psi_2(b_j - a_j)$ can be approximated by δ^{ζ_1} and $(b_j - a_j)^{\zeta_2}$, respectively. Thus we can always find a $\gamma^* < 1$ such that when $n_A > n^{\gamma^*}$, $\psi_1(\frac{n_A}{n}) \cdot \psi_2(\frac{n_A}{n}) \cdot n_A^{\eta(p)} \to \infty$. We define two groups of internal nodes $\mathcal{A}_1 = \{A_i,\, s.t.\, A_i \in \mathcal{A}, n_{A_i} \geq n^{\gamma^*}\}$ and $\mathcal{A}_2 = \{A_i,\, s.t.\, A_i \in \mathcal{A}, n_{A_i} < n^{\gamma^*}\}$, where n_{A_i} is the sample size at node A_i. Then we bound the probability:

$$\begin{aligned}
&P\left(\left\{\widehat{j}_A \in \mathcal{S} \text{ and } \max_j VI_A(j) > 2VI_A\big(\widehat{j}_A\big), \text{ for all } A_i \in \mathcal{A}\right\}^c\right) \\
&\leq \sum_{A_i \in \mathcal{A}_1} P\left(\left\{\widehat{j}_{A_i} \in \mathcal{S} \text{ and } \max_j VI_{A_i}(j) > 2VI_{A_i}\big(\widehat{j}_{A_i}\big)\right\}^c\right) \\
&\quad + \sum_{A_i \in \mathcal{A}_2} P\left(\left\{\widehat{j}_{A_i} \in \mathcal{S} \text{ and } \max_j VI_{A_i}(j) > 2VI_{A_i}\big(\widehat{j}_{A_i}\big)\right\}^c\right). \qquad (5.18)
\end{aligned}$$

For all internal nodes in $\mathcal{A}_1$, the number of nonmuted variables is less than or equal to p. Hence, by the monotonicity of $\eta(\cdot)$ in Assumption 3.4.6 and Equation 5.17, the first term in Equation 5.18 can be bounded above by

$$\sum_{A_i \in \mathcal{A}_1} C \cdot e^{-\psi_1(n^{\gamma^*-1})\cdot\psi_2(n^{\gamma^*-1})\cdot n^{\gamma^*\eta(p)}/K}. \qquad (5.19)$$

Note that in $\mathcal{A}_2$, the node sample size is less than n^{γ^*}. Since we choose the splitting point uniformly between the q-th and $(1-q)$-th quintile, to reach a node in $\mathcal{A}_2$, we need to go through a minimal of $-\gamma^* \log_q(n)$ splits. Noticing that this number goes to infinity, and that we mute p_d variables after each split, all variables except the ones in the protected set should be muted in $\mathcal{A}_2$. Hence, the second term in Equation 5.18 can be bounded above by

$$\sum_{A_i \in \mathcal{A}_2} C \cdot e^{-\psi_1(n^{\gamma-1})\cdot\psi_2(n^{\gamma-1})\cdot n^{\gamma\eta(p_0)}/K}. \tag{5.20}$$

Noting that $\mathcal{A}_1 \bigcup \mathcal{A}_1 = \mathcal{A}$, and that they contain at most $n^{1-\gamma}$ elements, and combining Equations 5.19 and 5.20, we obtain:

$$\begin{aligned} & P\left(\left\{\widehat{j}_A \in \mathcal{S} \text{ and } \max_j VI_A(j) > 2VI_A\big(\widehat{j}_A\big), \text{ for all } A_i \in \mathcal{A}\right\}^c\right) \\ \leq \;& C \cdot n^{1-\gamma} e^{-\left\{\psi_1(n^{\gamma^*-1})\cdot\psi_2(n^{\gamma^*-1})\cdot n^{\gamma^*\eta(p)} + \psi_1(n^{\gamma-1})\cdot\psi_2(n^{\gamma-1})\cdot n^{\gamma\eta(p_0)}\right\}/K}, \end{aligned}$$

which goes to zero at an exponential rate. Thus the desired result in this step is established.

Step 2: Now we start by decomposing the total variation and bounding it by the variable importance:

$$\begin{aligned} \mathbb{E}[(\widehat{f} - f)^2] &= \int (\widehat{f} - f)^2 d\,\mathbb{P} \\ &= \sum_t \int_{A_t} (\widehat{f} - \bar{f}_{A_t})^2 d\,\mathbb{P} + \sum_t \int_{A_t} (\bar{f}_{A_t} - f)^2 d\,\mathbb{P}, \end{aligned} \tag{5.21}$$

where $\bar{f}_{A_t}$ is the conditional mean of f within terminal node A_t, and where t indexes the terminal node. Noting that each terminal node A_t in $\widehat{f}$ contains $n_{A_t} \geq n^\gamma$ observations, and that the value of $\widehat{f}$ at each terminal node is the average of the Ys, it must therefore have an exponential tail. Hence the first term in Equation (5.21) can be bounded by:

$$\begin{aligned}
\sum_t \int_{A_t} (\widehat{f} - \bar{f}_{A_t})^2 d\,\mathbb{P} &\le \sum_t P(A_t) \cdot (\mathbb{P}_{n_{A_t}} - \mathbb{P}_{A_t}) f \\
&= \sum_t P(A_t) \cdot O_p(n_{A_t}^{\frac{1}{2}}) \\
&\le O_p(n^{-\gamma/2}).
\end{aligned} \tag{5.22}$$

The second sum in Equation (5.21) can be further expanded as

$$\begin{aligned}
&\sum_t \int_{A_t} (\bar{f}_{A_t} - f)^2 d\,\mathbb{P} \\
&\quad = \sum_t \int_{\mathbf{X} \in A_t} (\bar{f}_{A_t} - f(\mathbf{X}))^2 d\mathbf{X} \\
&\quad = \sum_t \int_{\mathbf{X} \in A_t} \Big(\int_{\mathbf{Z} \in A_t} f(\mathbf{Z}) \frac{d\mathbf{Z}}{P(A_t)} - f(\mathbf{X}) \Big)^2 d\mathbf{X} \\
&\quad = \sum_t \int_{\mathbf{X} \in A_t} \Big(\int_{\mathbf{Z} \in A_t} \big(f(\mathbf{Z}) - f(\mathbf{X})\big) \frac{d\mathbf{Z}}{P(A_t)} \Big)^2 d\mathbf{X}.
\end{aligned} \tag{5.23}$$

The Cauchy-Schwartz inequality now implies that

$$\begin{aligned}
&\sum_t \int_{A_t} (\bar{f}_{A_t} - f)^2 d\,\mathbb{P} \\
&\quad \le \sum_t \int_{\mathbf{X} \in A_t} \int_{\mathbf{Z} \in A_t} \big(f(\mathbf{Z}) - f(\mathbf{X})\big)^2 \frac{d\mathbf{Z}}{P(A_t)} d\mathbf{X} \\
&\quad = \sum_t \int_{\mathbf{X} \in A_t} E\big[\big(f(\mathbf{Z}) - f(\mathbf{X})\big)^2 \,|\, \mathbf{Z} \in A_t\big] d\mathbf{X} \\
&\quad = \sum_t P(A_t) \cdot E\big[\big(f(\mathbf{Z}) - f(\mathbf{X})\big)^2 \,|\, \mathbf{Z} \in A_t,\ \mathbf{X} \in A_t\big].
\end{aligned} \tag{5.24}$$

For each given A_t, due to the independence of $\mathbf{Z}$ and $\mathbf{X}$, the expectation in every summand can be decomposed as

$$\begin{aligned}
&E\big[\big(f(\mathbf{Z}) - f(\mathbf{X})\big)^2 |\, \mathbf{Z} \in A_t,\ \mathbf{X} \in A_t\big] \\
&\quad = E\big[\big(f(Z^{(1)}, ..., Z^{(p)}) - f(X^{(1)}, ..., X^{(p)})\big)^2 |\, \mathbf{Z} \in A_t,\ \mathbf{X} \in A_t\big] \\
&\quad = E\Big[\big(f(Z^{(1)}, Z^{(2)}, ..., Z^{(p)}) - f(X^{(1)}, Z^{(2)}, ..., Z^{(p)})\big)^2 \\
&\qquad + \big(f(X^{(1)}, Z^{(2)}, Z^{(2)}, ..., Z^{(p)}) - f(X^{(1)}, X^{(2)}, Z^{(3)}, ..., Z^{(p)})\big)^2 \\
&\qquad + \cdots \\
&\qquad + \big(f(X^{(1)}, ..., X^{(p_1-1)}, Z^{(p_1)}, ..., Z^{(p)}) - f(X^{(1)}, ..., X^{(p)})\big)^2 |\, \mathbf{Z} \in A_t,\ \mathbf{X} \in A_t\Big].
\end{aligned} \tag{5.25}$$

Note that the variables with the labels $p_1 + 1, ..., p$ are in the set $\mathcal{S}^c$ of noise variables. Changing the values of these components will not change the value of f. Hence the last term in the expectation of (5.25) is equal to

$$\big(f(X^{(1)}, ..., X^{(p_1-1)}, Z^{(p_1)}, X^{(P_1+1)}..., X^{(p)}) - f(X^{(1)}, ..., X^{(p)})\big)^2.$$

Again, since all the components of $\mathbf{X}$ and $\mathbf{Z}$ are independent, the jth term in the expectation of (5.25) corresponds to the variable importance of the jth variable. Thus we have:

$$\begin{aligned}
&E\big[\big(f(\mathbf{Z}) - f(\mathbf{X})\big)^2 |\mathbf{Z} \in A_t,\ \mathbf{X} \in A_t\big] \\
&\quad = E\Big[\big(f(Z^{(1)}, Z^{(2)}, ..., Z^{(p)}) - f(X^{(1)}, Z^{(2)}, ..., Z^{(p)})\big)^2 |\mathbf{Z} \in A_t,\ \mathbf{X} \in A_t\Big]
\end{aligned}$$

$$
\begin{aligned}
&+E\left[\left(f(X^{(1)},Z^{(2)},Z^{(2)},...,Z^{(p)})-f(X^{(1)},X^{(2)},Z^{(3)},...,Z^{(p)})\right)^2\middle|\mathbf{Z}\in A_t,\ \mathbf{X}\in A_t\right]\\
&+\cdots\\
&+E\left[\left(f(X^{(1)},...,X^{(p_1-1)},Z^{(p_1)},...,Z^{(p)})-f(X^{(1)},...,X^{(p)})\right)^2|\mathbf{Z}\in A_t,\ \mathbf{X}\in A_t\right]\\
=&\ \sum_{j=1}^{p_1} VI_{A_t}(j)\\
\leq&\ p_1\max_j VI_{A_t}(j).
\end{aligned}
\tag{5.26}
$$

It remains to show that $\max_j VI_{A_t}(j)\rightarrow 0$ as $n\rightarrow\infty$. Using Lemma 5.0.1, we have $\max_j VI_{A_t}(j)=o(n^{-C_1})$ where C_1 depends only on γ, p_1, and q. Moreover, the definition of C_1 shows that it is a strictly decreasing function of p_1. Hence

$$
\begin{aligned}
&E\left[\left(f(\mathbf{Z})-f(\mathbf{X})\right)^2|\mathbf{Z}\in A_t,\ \mathbf{X}\in A_t\right]\\
\leq&\ C_2\times O_p(n^{-C_1}).
\end{aligned}
\tag{5.27}
$$

Combining equations (5.21), (5.22) and (5.27), we have

$$\mathbb{E}[(\widehat{f}-f)^2] \ = \ O_p(n^{-C_3}), \tag{5.28}$$

where $C_3=(\min(C_1,\gamma/2))$. Due to the monotonicity of C_1, C_3 is also monotone decreasing in p_1. Noticing that C_3 does not depend on p, the convergence rate of RLT only depends on the choice of γ, q, and the number of strong variables p_1. This concludes the proof. □

Lemma 5.0.1. *Let $\mathcal{A}_{nT}$ denote the set of the terminal hypercubes. Then it holds*

$$\max_{A \in \mathcal{A}_{nT}, j \in \mathcal{S}} VI_A(j) = O_p(n^{-C}),$$

where C is a constant depending only on γ, p_1, and q.

Proof of Lemma 5.0.1. For any terminal hypercube $A \in \mathcal{A}_{nT}$, let $A_1 \to A_2 \to \ldots \to A_N = A$ be the constructed chain of the nodes leading to A, where A_{k+1} is the daughter node of A_k. Since at each node, the splitting point is chosen uniformly between the $100q$ and $100(1-q)$ quantiles of the current range of the splitting variable for some $q \in (0, \frac{1}{2})$, and since the terminal node is the last node having $\geq n^\gamma$ observations, it is easy to see that $-\gamma \log_q(n) \leq N \leq -\gamma \log_{(1-q)}(n)$. Let $j_k = \text{argmax}_{j \in \mathcal{S}} \widehat{VI}_{A_k}(j)$ be the index of the variable selected for splitting at node A_k and, moreover, define $m_j = \sum_{k=1}^N I(j_k = j)$, the number of times the jth variable is used for splitting. Let $N_j = \max\{k :, k = 1, ..., N, j_k = j\}$, the index of the last node split with the jth variable.

Before presenting the main proof, we state two simple properties:

Property 1. For $j \in \mathcal{S}$, $VI_{A_{N_j}}(j) \leq c_1(1-q)^{m_j}$. This is because after node A_{N_j}, the interval of the jth variable has been split m_j times so its length is at most $(1-q)^{m_j - 1}$. Therefore, according to the proof of Theorem 3.4.7, $VI_{A_{N_j}}(j) \leq c_1(1-q)^{2m_j}$.

Property 2. For $k = 1, ..., N-1$ and any $j \in \mathcal{S}$, $V_{A_{k+1}}(j) \leq 2VI_{A_k}(j_k)/q^2$. That is, the importance of any variable in the daughter node is no larger than the importance of the selected variable at the current node by a factor of $2/q^2$. This follows from

Theorem 3.4.7 (b): $2VI_{A_k}(j_k) \geq \max_j VI_{A_k}(j)$. On other hand, for any $j \in \mathcal{S}$, since $A_{k+1} \subset A_k$ and $|A_{k+1}| \geq |A_k|/q$, we have

$$\begin{aligned}
VI_{A_k}(j) &= \frac{E\Big[\big(f(X^{(-j)}, X^{(j)}) - f(X^{(-j)}, \tilde{X}^{(j)})\big)^2 I(X \in A_k, \tilde{X} \in A_k)\Big]}{\sigma^2 P(X \in A_k)} \\
&\geq \frac{E\Big[\big(f(X^{(-j)}, X^{(j)}) - f(X^{(-j)}, \tilde{X}^{(j)})\big)^2 I(X \in A_{k+1}, \tilde{X} \in A_{k+1})\Big]/q}{\sigma^2 P(X \in A_{k+1})q} \\
&= VI_{A_{k+1}}(j)/q^2.
\end{aligned}$$

Thus, $V_{A_{k+1}}(j) \leq VI_{A_k}(j)/q^2 \leq 2VI_{A_k}(j_k)/q^2$. With these two properties, we now proceed to prove the lemma. First, we define the following sequence:

$$N > \frac{N}{(rp_1)^1} > \cdots > \frac{N}{(rp_1)^{p_1}} > 0, \tag{5.29}$$

where r is a constant satisfying $r > 1$ and $2(1-q)^{2r}/q^2 = c \leq 1$. Since $0 < q < 1/2$, r can always be properly chosen. Correspondingly, we obtain intervals $W_k = [N/(rp_1)^k, N/(rp_1)^{k-1})$ for $k = 1, ..., p_1$ and $W_{p_1+1} = [0, N/(rp_1)^{p_1})$. Recall the definition of m_j, the number of times the jth variable is selected for splitting. Since $\sum_{k=1}^{p_1} m_j = N$, there must be at least one j such that $m_j \geq N/(rp_1)$ and $m_j \in W_1$. Furthermore, since there are $(p_1 + 1)$ intervals, there exists an integer $p_1 + 1 \geq k_0 \geq 2$ such that $m_j \notin W_{k_0}$ for any $j = 1, ..., p_1$. Hence, we can define two sets:

$$\mathcal{S}_1 = \{j : m_j \geq N/(rp_1)^{k_0 - 1}\},$$

and

$$\mathcal{S}_2 = \{j : m_j < N/(rp_1)^{k_0}\},$$

so that $\mathcal{S}_1 \neq \emptyset$ and $\mathcal{S}_1 \cup \mathcal{S}_2 = \{1, ..., p_1\}$.

Let j^* be the variable in $\mathcal{S}_1$ and split last among all the variables in $\mathcal{S}_1$ and let N^* be the node index where this variable is split last. In other words, the variables selected in the nodes A_k for $k > N^*$ are all from $\mathcal{S}_2$. Then using Property 1, we have $VI_{A_{N^*}}(j^*) \leq c_1(1-q)^{2m_{j^*}}$. Using the fact that $j^* \in \mathcal{S}_1$, we obtain

$$V_{A_{N^*}}(j^*) \leq c_2(1-q)^{2N/(rp_1)^{k_0-1}}.$$

Since all splitting variables after node A_{N^*} are from $\mathcal{S}_2$, and the number of the distinct variables is at most $(p_1 - 1)$, and the number of possible splits after $A_{N^*} = N - N^*$, is no larger than $(p_1 - 1)N/(rp_1)^{k_0}$. Hence we conclude: (a) if $N^* = N$, then

$$\begin{aligned} VI_A(j) &= VI_{A_N}(j) \leq 2VI_{A_N}(j_N) = 2VI_{A_{N^*}}(j^*) \\ &\leq 2c_1(1-q)^{2N/(rp_1)^{k_0-1}} 2c_1 \leq (1-q)^{2N/(rp_1)^{p_1}}. \end{aligned}$$

(b) if $N^* < N$, then according to Property 2,

$$VI_A(j) = VI_{A_N}(j) \leq (\frac{2}{q^2})^{N-N^*} VI_{A_N^*}(j^*) \leq (\frac{2}{q^2})^{(p_1-1)N/(rp_1)^{k_0}} VI_{A_N^*}(j^*).$$

Thus,

$$
\begin{aligned}
VI_A(j) &\leq \frac{2c_3}{(1-q)^2q^2}\left(\frac{2(1-q)^{2r}}{q^2}\right)^{(p_1-1)N/(rp_1)^{k_0}} (1-q)^{2rN/(rp_1)^{k_0}} \\
&\leq c_4(1-q)^{2rN/(rp_1)^{k_0}} \\
&\leq c_4(1-q)^{2rN/(rp_1)^{p_1+1}}, \qquad (5.30)
\end{aligned}
$$

where c_4 is a constant depending on p_1 and q, and where we used the fact that $2(1-q)^{2r}/q^2 < 1$.

Finally, since $-\gamma\log_q(n) \leq N \leq -\gamma\log_{(1-q)}(n)$, we obtain

$$\max_{j\in\mathcal{S}} VI_A(j) \leq c_5(1-q)^{-2r\gamma\log_q(n)/(rp_1)^{p_1+1}},$$

where c_5 is a constant depending only on p_1, q and R. The lemma holds.

□

Bibliography

Ali, K. M., and Pazzani, M. J. (1996), "Error reduction through learning multiple descriptions," *Machine Learning*, 24(3), 173–202.

Amit, Y., and Geman, D. (1997), "Shape quantization and recognition with randomized trees," *Neural computation*, 9(7), 1545–1588.

Bauer, E., and Kohavi, R. (1999), "An empirical comparison of voting classification algorithms: Bagging, boosting, and variants," *Machine learning*, 36(1-2), 105–139.

Biau, G. (2012), "Analysis of a random forests model," *The Journal of Machine Learning Research*, 98888, 1063–1095.

Biau, G., Devroye, L., and Lugosi, G. (2008), "Consistency of random forests and other averaging classifiers," *The Journal of Machine Learning Research*, 9, 2015–2033.

Breiman, L. (1996), "Bagging predictors," *Machine learning*, 24(2), 123–140.

Breiman, L. (1998), "Arcing classifier (with discussion and a rejoinder by the author)," *The annals of statistics*, 26(3), 801–849.

Breiman, L. (2001), "Random forests," *Machine learning*, 45(1), 5–32.

Breiman, L. (2004), "Consistency for a simple model of random forests," *Technical Report 670*, .

Breiman, L., Friedman, J., Ohlsen, R., and Stone, C. (1984), "Classification and regression trees," , .

Caruana, R., Niculescu-Mizil, A., Crew, G., and Ksikes, A. (2004), Ensemble selection from libraries of models,, in *Proceedings of the twenty-first international conference on Machine learning*, ACM, p. 18.

Chipman, H. A., George, E. I., and McCulloch, R. E. (2010), "BART: Bayesian additive regression trees," *The Annals of Applied Statistics*, 4(1), 266–298.

Ciampi, A., Hogg, S. A., McKinney, S., and Thiffault, J. (1988), "RECPAM: a computer program for recursive partition and amalgamation for censored survival data and other situations frequently occurring in biostatistics. I. Methods and program features," *Computer Methods and Programs in Biomedicine*, 26(3), 239–256.

Cutler, A., and Zhao, G. (2001), "PERT-perfect random tree ensembles," *Computing Science and Statistics*, 33, 490–497.

Dietterich, T. G. (2000), "An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization," *Machine learning*, 40(2), 139–157.

Efron, B., and Tibshirani, R. (1993), *An introduction to the bootstrap*, Vol. 57 CRC press.

Fleming, T. R., and Harrington, D. P. (2011), *Counting processes and survival analysis*, Vol. 169 Wiley. com.

Friedman, J. H. (2001), "Greedy function approximation: a gradient boosting machine," *Annals of Statistics*, pp. 1189–1232.

Friedman, J., Hastie, T., and Tibshirani, R. (2010), "Regularization paths for generalized linear models via coordinate descent," *Journal of statistical software*, 33(1), 1.

Geurts, P., Ernst, D., and Wehenkel, L. (2006), "Extremely randomized trees," *Machine learning*, 63(1), 3–42.

Graf, E., Schmoor, C., Sauerbrei, W., and Schumacher, M. (1999), "Assessment and comparison of prognostic classification schemes for survival data," *Statistics in medicine*, 18(17-18), 2529–2545.

Harrell Jr, F. E., Califf, R. M., Pryor, D. B., Lee, K. L., and Rosati, R. A. (1982), "Evaluating the yield of medical tests," *JAMA: the journal of the American Medical Association*, 247(18), 2543–2546.

Ho, T. K. (1998), "The random subspace method for constructing decision forests," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8), 832–844.

Hothorn, T., Bühlmann, P., Dudoit, S., Molinaro, A., and Van Der Laan, M. J. (2006), "Survival ensembles," *Biostatistics*, 7(3), 355–373.

Hothorn, T., Lausen, B., Benner, A., and Radespiel-Tröger, M. (2004), "Bagging survival trees," *Statistics in medicine*, 23(1), 77–91.

Hsieh, K.-L. (2007), "Applying neural networks approach to achieve the parameter optimization for censored data.," *WSEAS Transactions on Computers*, 6(5), 858–863.

Hunt, E. B., Marin, J., and Stone, P. J. (1966), "Experiments in induction.," , .

Ishwaran, H. (2007), "Variable importance in binary regression trees and forests," *Electronic Journal of Statistics*, 1, 519–537.

Ishwaran, H., Kogalur, U. B., Blackstone, E. H., and Lauer, M. S. (2008), "Random survival forests," *The Annals of Applied Statistics*, pp. 841–860.

Kim, H., and Loh, W.-Y. (2001), "Classification trees with unbiased multiway splits," *Journal of the American Statistical Association*, 96(454).

Kohavi, R., and Quinlan, J. R. (2002), Data mining tasks and methods: Classification: decision-tree discovery,, in *Handbook of data mining and knowledge discovery*, Oxford University Press, Inc., pp. 267–276.

Kosorok, M. R., and Lin, C.-Y. (1999), "The versatility of function-indexed weighted log-rank statistics," *Journal of the American Statistical Association*, 94(445), 320–332.

Kwok, S. W., and Carter, C. (1990), Multiple decision trees,, in *Proceedings of the Fourth Annual Conference on Uncertainty in Artificial Intelligence*, North-Holland Publishing Co., pp. 327–338.

LeBlanc, M., and Crowley, J. (1992), "Relative risk trees for censored survival data," *Biometrics*, pp. 411–425.

LeBlanc, M., and Tibshirani, R. (1998), "Monotone shrinkage of trees," *Journal of Computational and Graphical Statistics*, 7(4), 417–433.

Lin, Y., and Jeon, Y. (2006), "Random forests and adaptive nearest neighbors," *Journal of the American Statistical Association*, 101(474), 578–590.

Martínez-Muñoz, G., and Suárez, A. (2006), Pruning in ordered bagging ensembles,, in *Proceedings of the 23rd international conference on Machine learning*, ACM, pp. 609–616.

McLachlan, G., and Krishnan, T. (2007), *The EM algorithm and extensions*, Vol. 382 John Wiley & Sons.

Mehta, M., Rissanen, J., Agrawal, R. et al. (1995), MDL-Based Decision Tree Pruning.,, in *KDD*, Vol. 95, pp. 216–221.

Mingers, J. (1987), "Expert systems-rule induction with statistical data," *Journal of the operational research society*, pp. 39–47.

Mingers, J. (1989), "An empirical comparison of pruning methods for decision tree induction," *Machine learning*, 4(2), 227–243.

Niblett, T. (1987), "Constructing decision trees in noisy domains," , .

Quinlan, J. R. (1983), "Learning efficient classification procedures and their application to chess end games," in *Machine learning* Springer, pp. 463–482.

Quinlan, J. R. (1986), "Induction of decision trees," *Machine learning*, 1(1), 81–106.

Quinlan, J. R. (1987), "Simplifying decision trees," *International journal of man-machine studies*, 27(3), 221–234.

Quinlan, J. R. (1993), *C4. 5: programs for machine learning*, Vol. 1 Morgan kaufmann.

Rissanen, J. J. (1996), "Fisher information and stochastic complexity," *Information Theory, IEEE Transactions on*, 42(1), 40–47.

Sauerbrei, W., and Royston, P. (1999), "Building multivariable prognostic and diagnostic models: transformation of the predictors by using fractional polynomials," *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 162(1), 71–94.

Schumacher, M., Bastert, G., Bojar, H., Huebner, K., Olschewski, M., Sauerbrei, W., Schmoor, C., Beyerle, C., Neumann, R., and Rauschecker, H. (1994), "Randomized 2 x 2 trial evaluating hormonal treatment and the duration of chemotherapy in node-positive breast cancer patients. German Breast Cancer Study Group.," *Journal of Clinical Oncology*, 12(10), 2086–2093.

Segal, M. R. (1988), "Regression trees for censored data," *Biometrics*, pp. 35–47.

Simon, N., Friedman, J., Hastie, T., and Tibshirani, R. (2011), "Regularization paths for Coxs proportional hazards model via coordinate descent," *Journal of Statistical Software*, 39(5), 1–13.

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., and Zeileis, A. (2008), "Conditional variable importance for random forests," *BMC bioinformatics*, 9(1), 307.

Sutton, R. S., and Barto, A. G. (1998), *Reinforcement learning: An introduction*, Vol. 1 Cambridge Univ Press.

Tong, L.-I., Wang, C.-H., and Hsiao, L.-C. (2006), "Optimizing processes based on censored data obtained in repetitious experiments using grey prediction," *The International Journal of Advanced Manufacturing Technology*, 27(9-10), 990–998.

van de Geer, S., and Lederer, J. (2012), "The Bernstein–Orlicz norm and deviation inequalities," *Probability Theory and Related Fields*, pp. 1–26.

Van der Laan, M. J., and Robins, J. M. (2003), *Unified methods for censored longitudinal data and causality* Springer.

Van Der Vaart, A. W., and Wellner, J. A. (1996), *Weak Convergence* Springer.

Wei, G. C., and Tanner, M. A. (1990), "A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms," *Journal of the American Statistical Association*, 85(411), 699–704.

Zhao, Y., Kosorok, M. R., and Zeng, D. (2009), "Reinforcement learning design for cancer clinical trials," , .

Zhu, R., and Kosorok, M. R. (2012), "Recursively imputed survival trees," *Journal of the American Statistical Association*, 107(497), 331–340.

Zhu, R., Zeng, D., and Kosorok, M. R. (2012), Reinforcement Learning Trees,, Technical Report 33, The University of North Carolina at Chapel Hill Department of Biostatistics Technical Report Series.